# Affordable Reflectance Transformation Imaging Dome Manual

*Release 0.1*

Pawlowicz

Jan 18, 2019

# Contents:

CHAPTER 1

# Design History

This is just a quick explanation of how I came up with the basic LED lighting and control system I use in my RTI system. You might find it useful in understanding how things work, and I hope it's not too technical; but you don't have to understand this to build the system.

I first started working on this project early in 2013. I needed a control system that could turn on a large series of individual LEDs in sequence. I knew the control system would be driven by an Arduino, since the control requirements were pretty basic - turn lights on and off, fire a camera shutter in sync with the lights. A post on the Hackaday blog linked to a previous blog post on my project, and one comment in particular struck me. The author said that my project was "not much more than some LEDs mounted to a camera", and that he could build a comparable system for less than $100. I had to smile - I used virtually the same words to a friend of mine three-and-a-half years ago when I started working on the idea, including that exact dollar amount. Turns out that there's a whole lot of complications that you don't think about until you actually start working on a project like this.

The main idea the author had was that you could use inexpensive addressable RGB LED strips like this one to simplify the whole process of controlling the lighting, as well as cutting the cost. Nice idea, but three big problems that aren't immediately obvious.

## 1.1 The three big problems

### 1.1.1 LED strips

The LEDs on these 1-meter strips use 60 mA of current each (3.5 A strip total / 60 LEDs). I spent a lot of time figuring out how powerful the LEDs needed to be in order to light up a system up to a meter in diameter. My first thought was to power LEDs individually from Arduino Mega outputs, which limited me to 30 mA current (spec max is 40 mA, but it's recommended you stay away from that if possible). I tried "superbright" LEDs, in both straw-hat variety (wide uniform light dispersion), and focused (bright in the center, but narrow non-uniform beam). The straw-hats simply didn't have enough light intensity for reasonable camera exposure times even with smaller domes, a minimum of several seconds. The focused LEDs resulted in reasonable exposure times, but the small size of the lit area and its non-uniformity made them an unacceptable choice. Based on my initial experiments, I calculated that I needed at least 100 mA of current for reasonable exposure times for large domes with wide dispersion LEDs, and bought 0.5 W straw-hat LEDs for that.

Turns out I was optimistic, and the exposure times for the first dome I built (18" diameter) were on the order of a second or more, which translates to more than 4 seconds for larger domes. And this was with the lowest f-stop I had; dropping the f-stop to improve depth of field could increase exposure time by a factor of 4 or more. An upgrade to 350 mA / 1W LEDs brought that exposure time to about 1/2 a second, which would translate to 2 seconds for a larger dome, but still not good enough. That's why I ultimately went to my current design, which supports 1 A / 3W LEDs, and could in principle be modified to double that. 60 mA LEDs just don't produce enough light for reasonable exposure times.

A recent paper by Tom Kinsman had similar results. He built his own independent RTI system, using an Adafruit LED matrix driver controlled by an Arduino Uno that supplies a maximum of 50 mA of current to his LEDs. With a 12" diameter dome, and a macro lens stopped down to f/8.0, his exposure times were 4 seconds, which translates out to 40 seconds for a one-meter dome. OK for the small dome, albeit a bit slow, but way too long for a big dome.

Why are big domes important? Because a rough rule of thumb is that the largest object you can accurately image with Reflectance Transformation Imaging is half the distance to the light source you're using. So for a 12" dome, 6" radius, that's 3". I think this is conservative, and you can do a bit bigger, but there's still a limit. Larger domes allow for accurate imaging of larger objects - a one-meter-diameter dome can image an object about 10" in size.

### 1.1.2 Light intensity

When I was testing the 30 mA LEDs, I found that the variation in light intensity between different LEDs was as much as 20%. Reflectance Transformation Imaging requires that the light sources be as close to the same intensity as possible, otherwise the final result may be an inaccurate fitting of the light curve. I wasn't sure about the reason, but a bit of research brought up the most logical answer. While LEDs are sold with a quoted forward voltage for a specific current, what you actually get from a bunch of LEDs is a spread of forward voltages centered around the specification. Given how steep an LED I vs. V curve can be, a small change in Vf can result in big changes in the current, and big changes in the light intensity (see Grumpy Mike's excellent discussion of these issues). The current-limiting resistor commonly used with low-current LEDs is there to drop the voltage across the LED so that you don't burn it out, and not to specifically control the exact current. Given the low cost of the LED strip, I have to believe that it uses resistors to limit current to the LEDs, which could translate into substantial differences in output power between different LEDs. Not acceptable. My systems have always used adjustable constant current regulators to get around this problem.

### 1.1.3 Spectrum

Finally, RGB LED strips create "white" light by turning on the red, green and blue LEDs simultaneously. While it looks "white" based on appearance, it's not a very good white. Here's a "white" spectrum from an RGB LED:

The LEDs I wound up using for my RTI systems are Cree 3W/1A neutral white star LEDs. Here's a spectrum from a neutral white Cree LED (the green line), like the ones I spec for my system:

Much more balanced and uniform over the visible spectrum, with a peak near the blue end. This is easier to color correct than the RGB "white" spectrum. What's more, Cree LEDs are "binned" to have color rendering indices (CRI) that are close to each other; cheap super-bright LEDs can have large differences in CRI in the same batch

## 1.2 Control a swarm of LEDs

Next issue is how to control a large number of high-current LEDs from an Arduino Mega. While in principle there are likely enough outputs from a Mega to control a fair number of LEDs individually, in practice that would make the circuit design and the wiring a nightmare. The obvious solutions was to use a LED matrix design, with 8 rows x 8 columns for a maximum of 64 LEDs.

To turn on a single LED, you need to activate the appropriate electrical connections for the matching row and column, high-side voltage for the columns, low-side for the rows. For example, activating Column 4 and Row 5 lights up only the 4th LED in the 5th row.

Running sequentially through rows and columns, every light can be turned on just once. This requires the use of only 16 Arduino output pins to control all the LEDs.

Solving the next problem, getting accurate control of LED currents up to 1A, requires circuitry on both the high and low sides of the LED. Here's the solution I came up with:

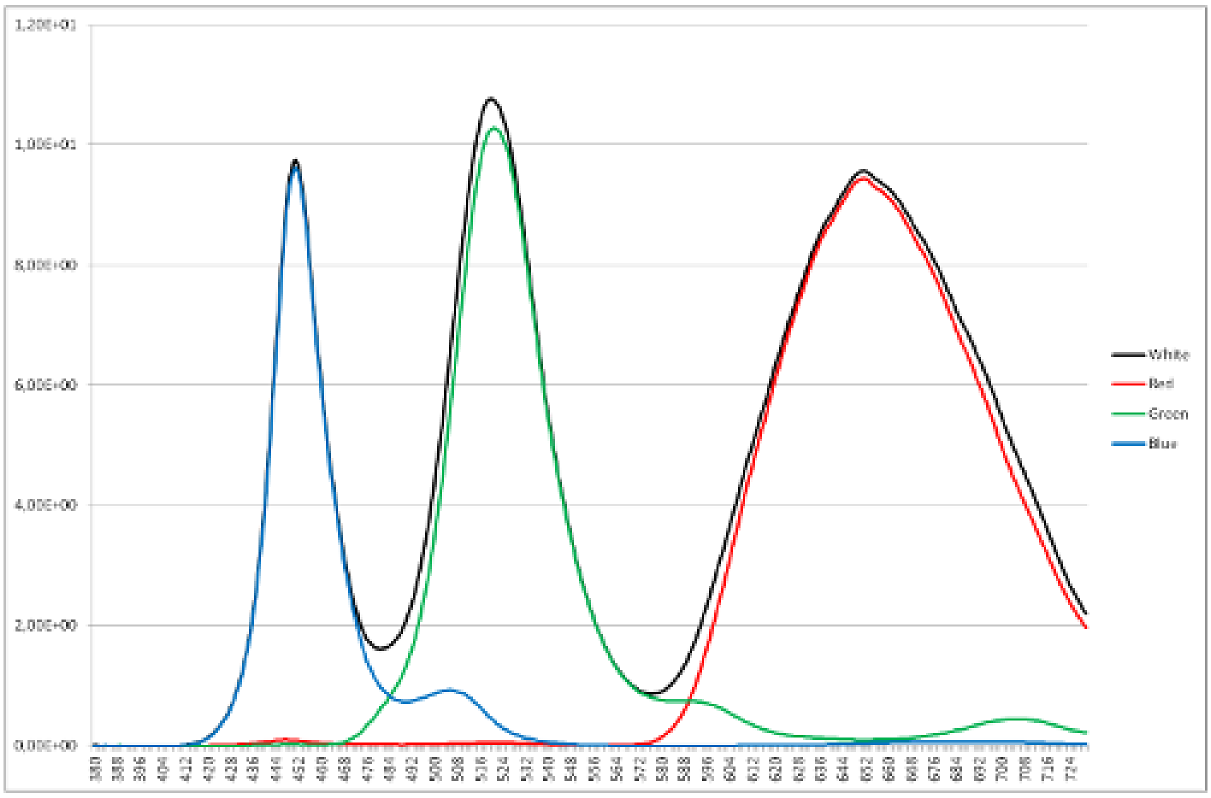


Fig. 1: White spectrum from an RGB LED, not very uniform.

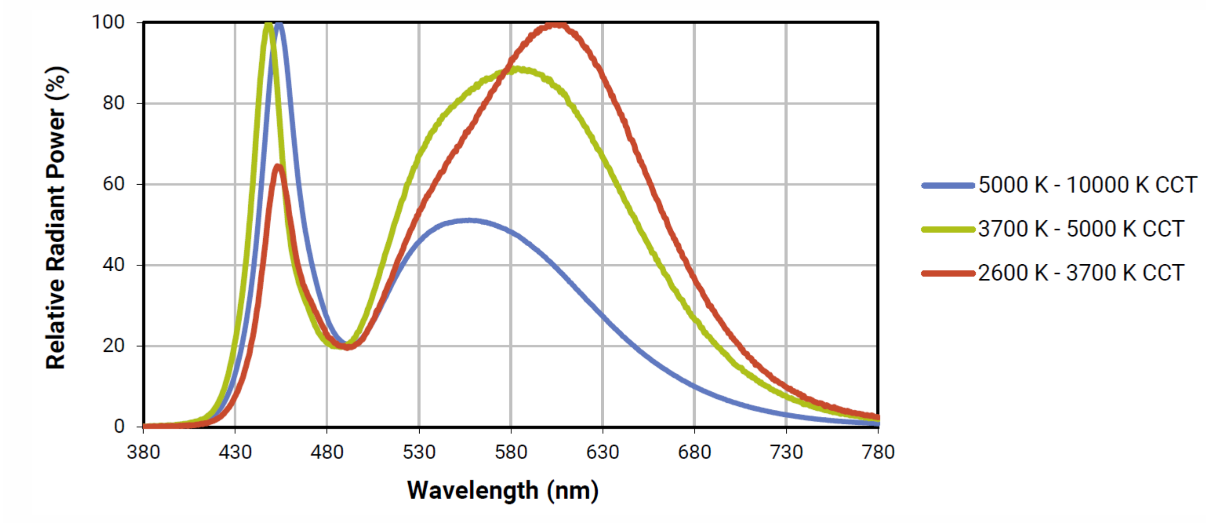


Fig. 2: Spectrum from a neutral white Cree LED (the green line).

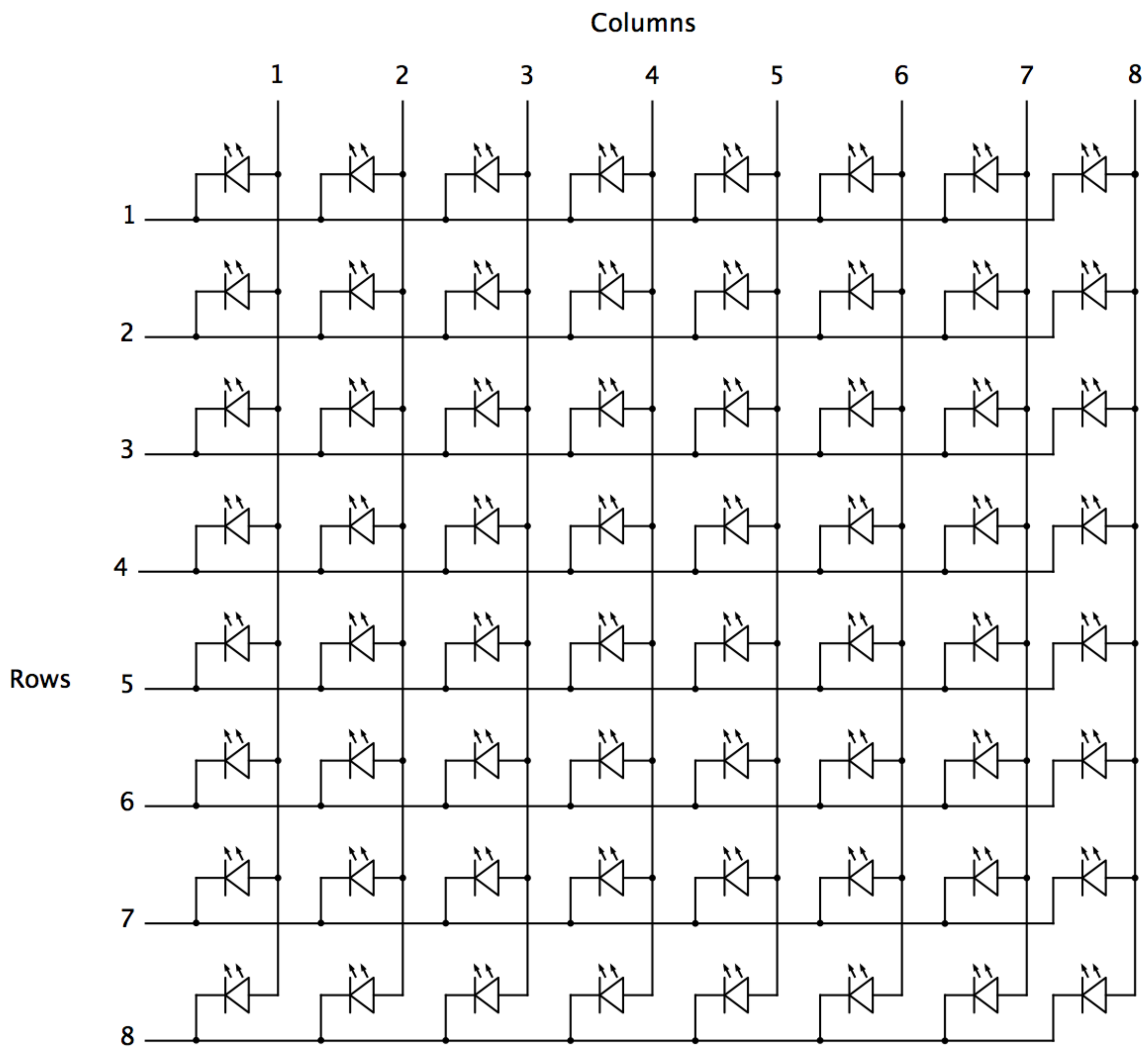


Fig. 3: LED matrix design with 8 rows x 8 columns.

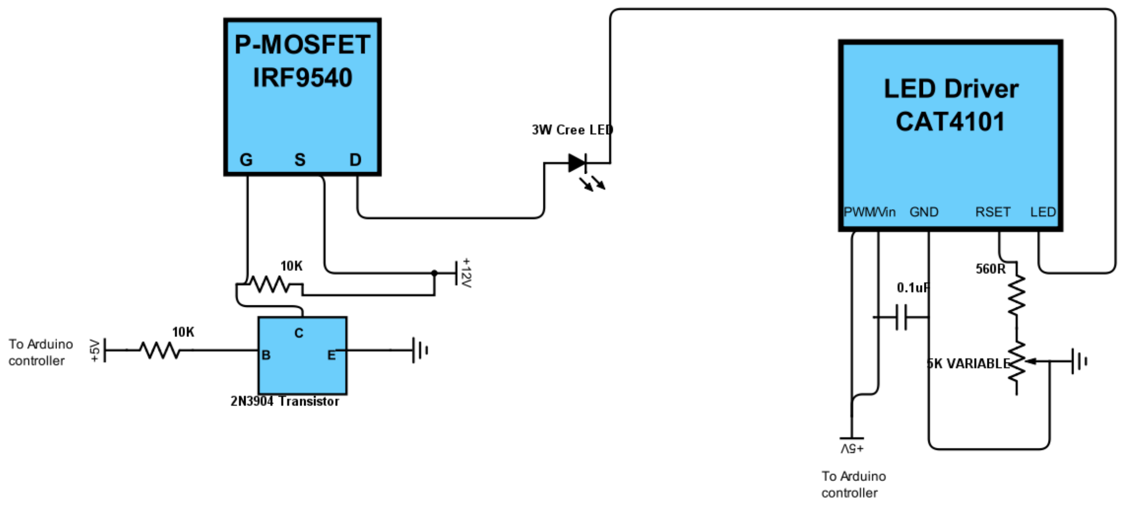


Fig. 4: LED circuitry

Sorry about the crummy schematic - most of mine are hand-drawn. On the left is a standard high-side P-channel MOSFET driver layout, which is easily found on the Internet (e.g. here and here); these control the columns of the LED matrix. I originally used a single-chip high-side Darlington array with 8 channels, but the maximum current was only 350 mA, resulting in LED light intensities that were still a bit dimmer than what I wanted; P-Channel MOSFETs can handle much higher currents, even though I had to now use 8 of them.

The P-Channel MOSFET controls on/off for the LED power supply, shown as +12V, but which normally runs at +9V, and can run +8-12V. The connection from +12V through the 10K resistor to the Gate is required because P-channel MOSFETs are on when the Gate is unbiased, so a voltage equal to the Source-Drain bias voltage needs to be applied to keep the MOSFET normally turned off. When +5V is applied from an Arduino controller output pin to the base of the 2N3904 transistor, it turns that transistor on, which shunts the +12V voltage away from the gate and turns on the P-MOSFET supply voltage to the LED. Check out the links above for a more coherent explanation.

Below is a better schematic of the high-side driver from the first site linked above; the “Load” in this case is an LED, and there’s no need for a flyback protection diode (commonly used with inductive loads like motors or relays). I use IRF9540 P-Channel MOSFETs because they’re cheap, and work fine, but an FQP27P06, FQ47P06, or NDP6020P P-Channel MOSFET would work just as well. You just need a P-channel MOSFET that can handle high currents, and has a low Rds.

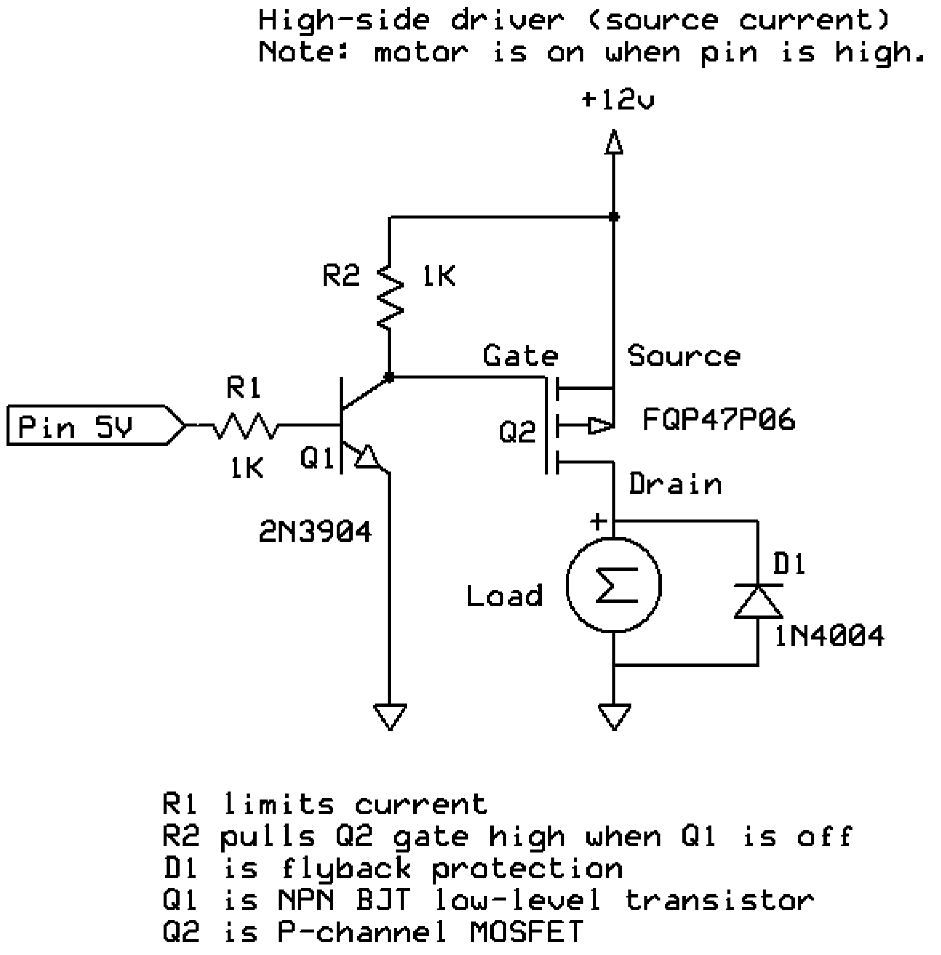

High-side driver (source current)
Note: motor is on when pin is high.
+12v
R2
1K
Gate
Source
R1
Pin 5V
Q2
FQP47P06
1K
Q1
Drain
2N3904
+
D1
Load
1N4004
R1 limits current
R2 pulls Q2 gate high when Q1 is off
D1 is flyback protection
Q1 is NPN BJT low-level transistor
Q2 is P-channel MOSFET

Fig. 5: Schematic of the high-side driver.

On the right-side of my schematic is the low-side driver/control (handling the LED matrix rows), a CAT4101 LED driver that not only switches on and off the low (ground) side, but allows you to set a desired current between 150 mA and 1 A. This is a 5-pin chip. The first pin, EN/PWM, switches on the LED and can control intensity through Pulse Width Modulation; the 2nd pin, Vin, supplies +5V of power to the chip. I have these two pins bridged in my control system, so that the same Arduino controller pin both switches the chip on and supplies power. I did this because leaving all of the CAT4101 chips powered on continuously led to some weird issues with multiple LEDs turning on even when only one should have. The GND pin goes to ground. RSET connects to resistors that set the output current, with 560R giving you about 1 A, the maximum.

The 5K variable resistor lets you fine-tune the current to allow for variations between different chips, and also allows you to set currents lower than the max 1A. This is useful for extending the life of the LEDs by keeping them from overheating, and also in certain use cases. For example, I sometimes use USB microscopes for micro-RTI, and full LED intensity is usually too bright for those; if I turn the current down to 150 mA, then there's no problem with sensor saturation. Finally, the LED pin connects to the ground side of the LED.

The schematic shows one high-side and one low-side driver, connected to one LED. But in the control system, there are eight high-side drivers, each connected to a column in the LED matrix, and 8 low-side drivers/constant current controls, each connected to a row in the matrix. Each driver is connected to an Arduino controller output pin that can turn it on and off as needed in software. This setup can drive up to 64 Cree 3W LEDs to their full power limit, at a current of 1 A.

Just for laughs, here's the inside of my original controller box from 2013, using a Darlington array:

Fig. 6: Original controller box from 2013.

And here's the most recent version I've built, the prototype for the system I'm describing on these project pages:

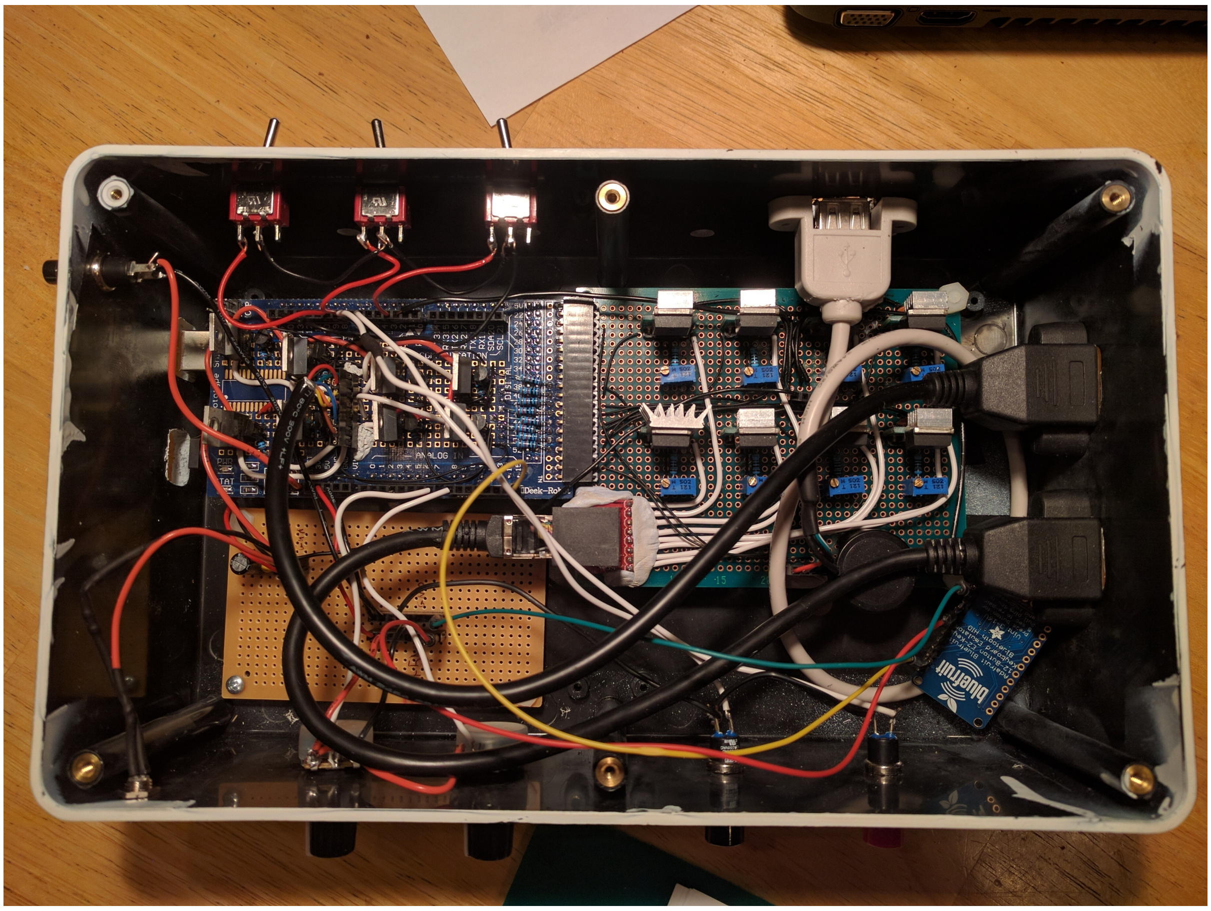


Fig. 7: P-Channel MOSFETs on the left, CAT4101 chips on the right side.

# CHAPTER 2

# Parts And Tools

This section will discuss how to choose the size of the dome for your RTI system, as well as what kind of cameras will work with the system. There's also a rundown of what the individual parts in the part list do. Finally, there's a brief discussion of what tools you'll need, with a special emphasis on those specialized tools that are either absolutely necessary, or which will make construction much easier.

## 2.1 Choosing a Dome

RTI requires you to take photographs from a fixed position above the object of interest, but at multiple lighting angles all the same distance from the center point of the object. All points at the same distance from a single center point => sphere. But you're only photographing one side of the object => dome. Attaching lights to the inside of the dome, pointing inwards, is probably the easiest way to handle the lights, and certainly the most common. There are alternate ways to get lights in an equiradial pattern without a dome, but those usually involve quite a bit more work and expense. Feel free to explore alternatives if you like, but from this point on, I'll only deal with domes.

The most important initial question you have to ask yourself is, how big a dome do I need? To answer that question, you have to think about what an ideal light source would be for RTI. It would be at a constant incident intensity across the entire object you were imaging, and also all the light rays would be parallel. But unless your light source is infinitely far away from the object, that's virtually impossible to achieve. For a perfect point light source, the intensity will drop off inversely proportional to the square of the distance (the classic inverse-square law, describing how a spherical intensity changes with distance). But LEDs are not a perfect light source in that their output light varies with angle. For the LEDs I'm using, maximum intensity occurs at an angle of 0 degrees, and drops off essentially with the cosine of the angle (the Lambertian distribution).

The further away the light is from an object, the less important both of these effects become. The radius of the sphere of light becomes larger and flatter, so that the variation in light across the object decrease; the angular width relative to the light position also decreases, so you're up at the flatter part of the Lambertian curve centered around zero degrees. So, ideally, you'd want a dome of nearly infinite radius. Of course this is logistically impractical, and you'd you have to increase the power of the light source so that some reasonable amount of it is still available to hit the object and light it up - also not practical. So you're going to have to live with some non-uniformity of lighting, but the bigger the dome, the relatively more uniform the light will be over a larger area.

Cultural Heritage Imaging says, based on their experience, that the maximum object size you can get good results from is roughly half the radius of the dome, e.g. a 12" diameter dome would have a radius of 6" and a maximum object size of about 3"; 18" dome has a 9" radius and a 4.5" max object size; a one-meter dome would have a radius of 50 cm, and a maximum object size of 25 cm (just shy of 10"). My personal opinion is that these are

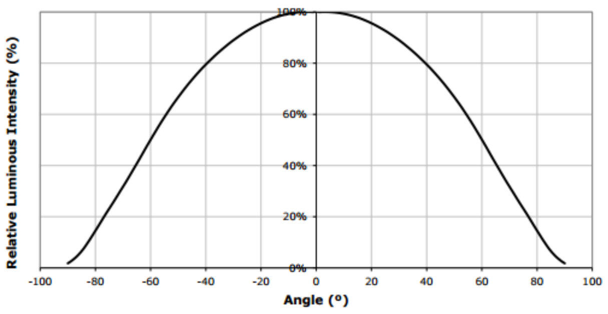


Fig. 1: Lambertian distribution of light intensity with angle in one dimension (Source: ledsupply.com)

conservative numbers, and you can get useful results from objects that are up to 25% larger than these, but YMMV. I'm working on some ideas to extend these size limitations, but they're not ready yet.

So bigger is better? In terms of light uniformity over a larger area, yes. But there are trade-offs:

1. Larger domes mean that the light intensity will be lower at the object, since it's further away from the light source. For my 12" dome, exposure times are typically on the order of 1/15th second, while they're about 1/5th second for my 18" dome. Extrapolate out to a meter-size dome, and you get a one second exposure. This is with a compact point-and-shoot at 3.1x zoom (f/5.6) at ISO 100, and you would get better results with a better camera using a faster lens, larger aperture, higher ISO, etc.. The camera is in a fixed, rigid position so you don't have to worry about camera shake limiting shutter speeds. Still, at some size, the exposure times may become longer than you're willing to live with.
2. Larger domes will cost more. I've bought all my domes from EZ Tops, and been very happy with them. If you check their website and price domes, the listed price of a 24" dome is only about $20 more than a 12" dome. But it's the shipping costs that will kill you - they're about $35-40 for a 12" dome, and over $100 for a 24" dome. Plus, a 12" dome only has room for 48 LEDs (15" minimum for 56 LEDs, 18" minimum for the maximum 64 LEDs supported by my controller design), so you'll save some money by not needing as many LEDs.
3. Larger domes require larger stands and more space, and are less portable.
4. If you want to do high-magnification RTI, working either with a USB microscope or a DSLR with a macro lens, you'll need to have a smaller dome for maximum magnification. Typical working distances for macro lenses are on the order of 6", as are typical lengths for USB microscopes, so a 12" diameter dome would probably be the optimum size for that application. Note 1: My controller can work with any size dome, so budget allowing, you could build multiple domes of different sizes, and run them with the same controller (not all at the same time). Note 2: Distances for macro lenses are often given as the distance from the object to the camera's focal plane, which is not the same as the working distance, the distance from the front of the lens to the object you're photographing.

All a long-winded explanation leading to some basic rules:

- If the most important factor to you is the size of the objects you want to image, measure the largest dimension of the largest object you need to image, multiply by 4 or a bit less, and that's the minimum dome diameter you'll need.
- If you need to do high magnification imaging with a macro lens or USB microscope, I'd recommend 12" diameter as the optimum size. I wouldn't go smaller, as you rapidly run out of room to install LEDs inside with smaller domes than that.

Here are a few more details to keep in mind when ordering a dome.

- While you can use any dome-shaped object you like, like a styrofoam dome, I've always used acrylic domes. They're sturdy, and come in a wide range of sizes. My source for domes in the US has been EZ Tops Worldwide; they have a wide variety of sizes and styles, and prices are reasonable compared to other sources I've seen online. Outside of North America, you're best off finding a local supplier if at all possible; shipping costs for domes are high. If you're from outside North America, and find a good dome supplier, please let me know in the comments and I'll add them to a list.

- Speaking of which: the prices at EZ Tops are for the domes alone. Shipping costs can easily increase the cost by a factor of 2 or more - the larger the dome, the higher the factor. I'm guessing this will be true for any acrylic dome vendor.
- When spec'ing the dome, make sure to specify the following dimensions
  - Flange dimension. This is a rim on the outside of the dome, useful for handling the dome as well as attaching it to a stand. 3/4" flange size is fine for domes up to about 15" diameter, above that I'd spec 1". No need to go larger than 1", even on a big dome.
  - Dome diameter. This is the size you choose based on the previous project log entry. This diameter does not include the flange.
  - Dome height. This should be half the dome diameter, so that the shape of the dome is essentially a uniform half-sphere.
  - Material thickness. Thinnest available thickness is fine - the dome isn't going to bear any significant amount of weight, and the material is quite rigid. Thinnest is also cheapest.
- EZ Tops domes come in several acrylic colors: clear, white, tinted, and black. You're going to be painting the interior of the dome to minimize intrusive lights, and light scatter, so any of these would be usable. Clear is the cheapest option, and will have a cool shiny black appearance after you paint the inside of the dome; this is best for any domes you will only be using indoors. For a dome intended for outdoor portable use, I'd recommend white, since that color will help keep it cooler in the sun. I'd stay away from tinted or black; not only are they more expensive, but they make marking the positions to mount LEDs more difficult (you'll see why in the first instructions section, coming soon).
- Domes need to have a hole in the top for the camera to look through. I have manually cut holes in acrylic domes several times for this purpose, but I will never do it again. Acrylic is very tough to drill through - melts when you cut, the bits catch on the edges, and it cracks/breaks all too easily. When you choose a dome supplier, make sure they offer the option to cut a hole in the top. Costs a few extra bucks, but it's definitely worth it in ease and peace of mind. The size of the hole will depend on the size of the camera lens/aperture you plan to use with the system. 2.5" is a good size for most point-and-shoots, and also for DSLR macro lenses that won't need to be lowered into the dome (e.g. for a 12" diameter dome, 6" hole to specimen distance, 6" lens working distance). If you do need to lower a macro lens into the dome a bit to get closer to the object, you'll need to specify a hole diameter that will allow the entire lens to fit inside. I think 3" should work for most macro lenses, but you'll need to check your lens size specs to make sure.
- Don't throw out the box they ship the dome in - it makes a handy carrying container.

## 2.2 Choosing a camera

The RTI system I've designed and built has full control over turning lights on and off inside the dome. It can also turn the camera on and off in sequence with the lights, and do both in an automated, sequential fashion but only if there's a way to fire the camera automatically using the Arduino-based controller. There is a manual option, where you can turn a light on, press the shutter manually, then go on to the next light, but that requires you to sit at the system and push buttons for a few minutes at a time; automatic is much better.

The following camera remote modes are currently supported by the control system:

1. The control software uses Sebastian Setz's Multi Camera IR Control library, which currently supports cameras with IR remote capability from Canon, Nikon, Olympus, Pentax, Sony, and Minolta. Double-check that your camera has IR capability; for example, Canon DSLRs do but most Canon point-and-shoots don't.
2. I've also built hardwired remote cables that work with Canon/Nikon cameras, using the pin out specs at this page. Basically, you use an optoisolator as an electronic relay to close a circuit and fire the camera. If your camera supports a hardwired remote, there should be a way to hack it to work this way as well.
3. Most Canon point-and-shoots do not support a remote by default. But the Canon Hackers Development Kit (CHDK) allows many older Canon point-and-shoot camera models to be fired remotely through the USB connector, and this is the system I normally use for most of my RTI photography. Not all Canon point-and-shoots are supported, though; in particular, support for the most recently-released Canon point-and-shoots

is spotty to non-existent. Check the list of supported cameras on the main CHDK page to make sure your camera is supported. I use the Canon S110 Powershot (not the Elph); not only does it support CHDK, it also has support for the RAW format. It's also very small and lightweight, perfect for a portable system. The slightly older Canon S100 Powershot also supports RAW, and is somewhat less expensive. eBay is a good place to find older Canon models in good condition for reasonable prices.

4. If your camera can be fired via computer software, the control code supports firing the shutter using the Adafruit Bluetooth HID module. I use this for automated control of microscopic RTI using USB microscope imagers, but it should work fine with Canon/Nikon PC software for controlling cameras, or any other computer-based camera controller.
5. I'm planning to design and build a mechanical automatic shutter control using a simple servo mechanism – stay tuned. If you see any such designs on the web, let me know, as I'm perfectly happy to use someone else's working design rather than having to come up with my own.

## 2.3 Parts, Parts, Parts

I'll talk about elements of the components list here with a description of what every part is supposed to be used for; hopefully, that will make the design and assembly process more understandable. Check the original components list for required quantities; there's an Excel spreadsheet version of the components list that should be kept up-to-date at all times.

A few comments on parts suppliers. For many of the discrete components, I link to a Tayda Electronics page. These guys have great prices on discrete components even in small quantities, they ship fast, and the shipping costs are very reasonable. For parts/items they don't have, I use Digikey - great service and fast shipping, but prices tend to be somewhat higher for many components. I also link to pages on eBay, AliExpress, Amazon and other vendors for various items not found on Tayda or Digikey. But feel free to use your own favorite vendor.

Three general categories:

- controller parts for the main control unit
- dome parts for the RTI dome itself
- camera parts for the odds and ends that you'll need to connect your camera to the controller for automatic shutter control.

### 2.3.1 Controller parts

**Arduino Mega 2560 R3 Controller** Main control unit. I've used several brands of less-expensive clones without problems. R3 is the most recent model, with the fastest processor, but if you have an older non-R3 unit, that will work as well.

**Power supply** Any Arduino-compatible power supply should work, as long as it's rated for at least 2A current. 9-12V OK but 9V recommended, 2.1mm x 5.5mm barrel jack, center-positive. While the max current draw is only 1 A, I've had problems with 1 amp rated power supplies being noisy near their maximum current spec.

**Battery power supply (optional)** Required for portable operation. Uses 6 AA alkaline batteries. The linked item already has a standard Arduino jack (2.1mm x 5.5mm barrel jack, center-positive), but you can certainly hack together your own as well. If you plan to use rechargeable NiMH batteries, you'll need a holder that takes 8 AA batteries, since NiMH batteries have lower voltages than AA (1.2V vs. 1.5V each).

**Arduino Mega protoshield** A handy protoboard that plugs into the Arduino Mega, and has room on top to solder in the high-side driver components (albeit just barely). Make sure the one you buy looks like the one I link to.

**11 x 8.5 cm protoboard** I highly recommend the one I link to, as it's cheap, the right size, and the metal goes all the way through the holes producing a secure solder hold. On cheaper protoboards, the metal is often just a little disk on top, and has a tendency to fall off. If you buy one of a different size, you may have to modify the component layout during the build, and you should also check to make sure it will fit in your enclosure of choice.

**10K linear taper potentiometers and knobs** Control the duration of the light, and the delay to allow the camera time to process and save the image before taking the next picture. 5K will work fine as well, and feel free to choose any style knobs you want.

**Momentary push buttons ('red and black)** Activate key functions, and one red button also resets the Arduino.

**Toggle switches** Set options: sound on/off, manual/auto mode, shutter hardwire or IR/Bluetooth mode. The specified switches are single pole/double throw, but single pole/single throw (i.e. basic on/off) are fine for this application (e.g. slide switches).

**Red LED, panel LED holder, 560R resistor** For power indicator that hooks up to 8-12V power supply. I use a 560R resistor because 8 of those are needed for another purpose, and minimum resistor order from Tayda is 10. If you have a 470R resistor, you can use that instead.

**USB Female A Panel Connector** Main jack for hardwired camera shutter connections, and also for the IR LED if you use a camera with LED remote capabilities.

**Ethernet RJ-45 panel jack** The controller connects to the dome through Cat5E cables, which plug into these panel jacks.

**RJ-45 Jack and breakout board** One RJ-45 jack is mounted internally, and the breakout board is needed to mount it to the protoboard.

**CAT4101 LED Driver and breakout board** CAT4101s are the main switch/drive/current controller on the low side. They are surface-mount devices, so breakout boards are needed to use them on the through-hole protoboard.

**0.1 uF disc capacitor** Use to suppress noise on input of CAT4101.

**560R resistors and 5K trim potentiometers** CAT4101 use a resistor to set the desired current, with lower resistances yielding higher currents, up to 1 A max. The 560R resistor sets the maximum allowed current of 1 A; the trim pot is in series with that resistor, allowing you to set lower output currents if desired, as well as allowing you to fine-tune the output of all eight CAT4101s so that they match.

**10K-ohm resistors** Used in several locations to limit currents.

**IRF9540 P-Channel Power MOSFET** High-side switch to LED matrix; controls columns.

**2N3904 NPN transistor** Controls IRF9540 MOSFETs.

**Power strip protoboard** There are a lot of ground and +5V/+9V connections required, more than are available on the Arduino itself. The power strip protoboard adds extra female pin headers to accommodate those connections.

**100 uF capacitor** Smooths out any major variations in the 9V input.

**Piezoelectric buzzer** For audio feedback of several functions. Can be turned off in hardware or software.

**Heat sinks (7mm x 7mm to 10mm x 10mm)** These probably aren't necessary. The MOSFETs are rated to operate continuously at up to 13A at 100C, and they will never see more than an amp for a few seconds at a time at much lower temps; while I had heat sinks for these on the component list initially, I've removed them as unnecessary. The CAT4101 drivers will need to burn off more voltage, but even they won't be on for long periods, and have a thermal shutdown at 150C. Still, just to be safe, I use small heat sinks on the Cat4101s. I recommend getting heat sinks that come with tape adhesive in place - makes installation a lot easier.

**Various headers** Used for wiring connections.

**Dupont pins (male and female)** Used primarily for the LED wiring connections, but also in a few other locations. Housings needed for the female pins. If you live near the sea and corrosion is an issue, consider getting gold-plated Dupont pins, available on eBay; just slightly more expensive.

**Black and red 22 AWG solid hookup wire** used for making connections inside the controller. Red and black to keep track of hot and ground connections, respectively. I say 10 ft. of each, which is probably more than you'll need, but it's cheap.

**Black Kynar insulated 24 AWG solid wire** The main wire for connecting LEDs inside the dome, and protoboard interconnections. One 100-foot roll is probably enough for domes up to 24" in diameter; larger domes may require a second roll.

The next set of parts is for the enclosure that will hold the electronics, and into which will be installed switches, pots, LEDs, panel jacks, etc. Feel free to design and construct your own, using my model as an example.

**Plastic enclosure box** This was the only pre-made one I could find that was big enough to hold everything. It's a nice box, but kind of expensive- shipping almost doubles the price.

**Rubber feet** To put on the bottom of the enclosure box, to keep it from sliding around,

**Mixed nuts, washers, bolts, screws, spacers** Attach the main control electronics to the box.

### 2.3.2 RTI Dome Parts

What you need to build the RTI dome, to light up the object from multiple angles and take photographs through the top.

**Dome** I've already posted here and here on what's required for your dome. Deviate from these recommendations at your own peril. I've been happy with the domes I've gotten from EZTops, but any reliable source should be fine. Amazon has a 12" dome that looks good, for a slightly lower price than EZTops.

**Flat/matte black spray paint for plastic** Used to cover the inside of the dome, to minimize scattered light from the interior. Make sure whatever black paint you choose dries with a "flat" or "matte" non-reflective finish. Avoid "satin" finishes, "glossy" is totally unacceptable. Also make sure the paint is specified as working with plastics. I found Rustoleum works best; Krylon also works, but requires more coats. For larger domes (>24" diameter), chalkboard paint is likely to be an easier and less expensive option.

**Black reflective sphere** This will be used to calibrate the dome LED positions. I reference a high-quality silicon nitride bearing, but any decent black reflective sphere will work. This includes black marbles, stainless steel ball bearings dipped in black ink, or obsidian spheres for larger systems. For proper calibration, the sphere must have a diameter of at least 250 pixels when photographed from the top of the dome.

**LEDs** The system was designed to handle up to 3W LEDs mounted on star bases for heat sinking. I highly recommend Cree LEDs, and I also highly recommend buying from a reputable source. I made the mistake of buying "Cree" LEDs at bargain prices from Chinese vendors, which suffered from non-uniform intensities and wildly varying color balances. Official Cree LEDs are binned to match fairly closely in intensity and color balance. My vendor of choice is LED Supply, decent prices, volume discounts, and they ship quickly. Buy a few more than the minimum you need, in case you have problems in assembly, or wind up with a bum LED or two. New models of LED come out on a regular basis, my current choice is the 3W neutral white XP-E2 variety, 1up star. For domes 12-14" diameter, I recommend no more than 48 LEDs; 15-17", 56 LEDs; 18" and up, 64 LEDs (the maximum number the controller can handle).

**2mm heat shrink tubing** To insulate LED wiring connections.

**Cat5e 24 AWG Ethernet patch cables** These are used to connect the dome to the controller. I spec 24 AWG to maximize the current capacity and reduce resistive heating I've run 1 A through the individual wire strands continuously for up to an hour without any significant heating. Monoprice is an awesome source for these, as well as any additional USB cables you might need. I spec 5 ft., but choose a longer length if you want more flexibility; you'll be cutting one end off of these, so you can always shorten them if necessary during the system build. I strongly recommend that one of the cables you buy be red, to be designated as the positive/high connection to the dome. For the other cable, pick any other color. I usually choose a color to match the dome (black or white for all the ones I've built to date).

**4" cable/zip ties** Used to tidy up cables inside the dome. Available everywhere; I usually pick a color that matches the exterior of my dome.

### 2.3.3 Camera cables

If you haven't already, read my log on camera considerations for additional background on these parts. I highly recommend Monoprice for all your USB cable needs.

#### Canon camera with CHDK:

All you need is the standard USB cable that plugs into the camera on one end, a female USB jack on the other. Depending on your configuration, you may want to get a longer cable, or a USB extension cable.

#### Camera with IR remote capability:

- 940 nm IR LED
- 47R 1/2-W resistor
- USB cable

Assembly is easy, just wire the resistor in series with the LED, connect to the USB cable; see the assembly instructions for more details. Software supports cameras made by Canon, Nikon, Olympus, Pentax, Sony, and Minolta that have IR remote capability. I recommend this option heartily for any camera that supports IR remote; cheapest, easiest and most flexible option.

Camera with hardwired remote capability:

- 4N35 Optoisolator
- 220R resistor
- Remote connector plug compatible with your camera. You may have to buy a cheap hardwired remote on eBay or Amazon, and then cut off the connector plug.
- USB cable

More explicit instructions in the assembly instructions.

### 2.3.4 Dome stand

Check the assembly instructions - lots of options here, as well as the freedom to set it up anyway you want.

## 2.4 Tool time

Apologies to experienced makers - you probably have/know all of this. This project is potentially of use to many people that might not be makers (like archaeologists and paleontologists), so I'm throwing in a bit of background material on the toolset that's needed. The links are to show you examples of what you'll need (I don't usually own the ones I've linked to) - any equivalent to these tools will be fine.

### 2.4.1 Absolutely required

If you're going to build this RTI system, you'll definitely need the following tools: buy, beg, borrow or steal.

#### Soldering tools

The electronics are put together using soldering for the most part, nothing fancy. If you've never soldered before, it's not that hard. Check YouTube for training videos on how to solder, some are very good. This Adafruit guide is worth a look.

**Soldering iron** Don't need a super-fancy one, just make sure it has a point tip on it; I'd recommend 40W or higher, with controllable temperature. You don't want a soldering gun, or an iron with a large chisel tip, as these won't work well for the small-scale through-hole soldering you'll be doing. The linked soldering iron includes a few extras, including some items listed below. I own this one, a bit more control.

**Solder** Get the thin stuff, 1 mm (.039") or smaller in diameter; easier to handle, and melts faster. Also make sure it's rosin core.

**Brass tip cleaner / sponge tip cleaner for solder** Cleans oxide and unneeded solder of your soldering iron. I prefer the brass cleaner, but the wet sponges work as well.

### Wiring tools

**Wire stripper** No, the cheap one with the screw adjusted won't work. Get one with fixed holes for AWG wire diameters 22 to at least 26. I use the linked one, but there's a slightly cheaper one by a reputable maker at Digikey.

**Flush cutter or diagonal cutter** Used for trimming electrical wires and part leads. The Hakko flush cutter works well and is cheaper, but the diagonal cutter works nearly as well, and you're likely to find it handier for other tasks in the future than the Hakko.

**Needle-nose pliers** For inserting parts, bending wires, holding nuts.

**Multimeter** Critical for testing the circuits, and measuring/calibrating the output current. Pick one that can handle currents up to at least 1 A (most do up to 10 A).

**Alligator leads** Useful for connecting the multimeter to wires; also double as clamps for soldering.

### General assembly tools

**Crimper** Even with a small dome, you'll be crimping hundreds of Dupont pins, male and female. Technically you could do this with a pair of pliers, but after 50 or so crimps with pliers (and with multiple failures), you will start praying for death. These take a bit of practice, but once you get the hang of them, they're pretty easy to use.

**Electric drill, drill bits** No link for this one, hopefully you have one of these or know someone with one. Largest hole you may have to drill is 1/2" diameter.

**1/8" acrylic drill** Acrylic plastic is a pain to drill with regular bits; you'll get lots of chipping, grabbing, and even breaking. This bit has a special design to help eliminate these issues. For larger holes, you can use regular bits after creating a started with this one; just gradually use bits of increasing size until you get to the final size.

**Random tools (screwdriver, file, saw, etc.)** A whole bunch of random tools that you may need, but are pretty common.

**Silicone adhesive** Used for attaching LEDs to the dome. You won't need a lot of it.

**'Safety glasses <>'_** 'Nuff said.

## 2.4.2 Optional, but helpful

**Head magnifier** If you have young, good eyes, you may not need this. My eyes are old and decrepit, so this magnifier makes soldering and other fine work possible.

**Solder sucker / solder wick <http://amzn.to/28TjPRT>'_** If you're a master solderer and never make mistakes, you may not need these <>'_ they help fix soldering problems by removing solder. I have both :). Check this video for more info.

**Flux** Love this stuff; makes soldering problem connections a lot easier. Cleans off with rubbing alcohol.

**Dremel or other rotary tool** Primarily useful for cutting holes in the controller enclosure, as well as general cutting/sanding/filing/milling. You can do all this stuff with other tools as well, it may just take longer.

**Heat gun** Used for heat-shrink tubing in this build, special flexible tubing that shrinks when heated to insulate/protect/reinforce wire connections. I don't have a heat gun, but use a hair dryer on the hottest setting put right next to the heat shrink tubing; works, but a bit slow. A lighter or candle flame works on heat shrink tubing as well, as long as you're careful not to melt the tubing or set it on fire.

**Anti-static wrist strap** Static discharge is bad in general for electronics, and the P-channel MOSFETs in particular are very static-sensitive. This wrist strap offers a level of protection from static discharge.

**Step drill (aka Unibit)** The 1/8" acrylic drill listed above will be good enough for most of the holes you need to drill. However, there are a couple of holes in the dome that will need to be a bit larger. You can drill these by starting with the acrylic drill, and then gradually increasing the size of the drill bit you use until you reach the final size. If you skip too quickly to a larger size, you risk cracking or breaking the dome. A step drill has a conical shape with gradually increasing bit sizes in a single bit, you start with a smaller size, and gradually work your way up to a larger size. I've used these often on acrylic plastic, and as long as you start with a pre-drilled hole, you shouldn't have problems with cracking/breakage as long as you don't apply too much pressure while drilling.

I may think of more tools as I write up the build, watch this space.

CHAPTER 3

# Assembly Logistics

I've had a few emails asking about how difficult this project will be to make, and how long it will take. Not exactly sure how to answer the first question, as it depends on your level of experience and willingness to learn. Having said that, if you know or can pick up simple soldering, and know how to use tools, there's nothing that you should find particularly challenging. Hopefully that's especially true because, for better or for worse, I'm trying to document every single step as thoroughly as I can.

As far as time, I think that if I were pushed, and had all the parts, I could probably put together a system in a long weekend plus a few extra weeknights (given the benefit of experience plus now having a set of written instructions). However, if you can grab a bunch of friends to help, much of the project could be broken down into subsections that people can work on independently of others at various stages.

Even some of these can be broken into separate concurrent processes that several people can work on at the same time:

1. Dome prep, marking and painting (do this one day ahead of all other steps)
2. MOSFET driver shield assembly
3. CAT4101 driver board assembly
4. Power board and Arduino power wiring
5. Control box preparation (drilling/cutting holes)
6. Individual LED wiring
7. LED Matrix / Dome wiring (allow one day for adhesive to dry)
8. Necessary hardware (stand, camera control, etc.)

With multiple assemblers at work, this can easily be a one-weekend project. Beer is a useful bribe.

I've embedded an assembly flow chart below that shows all of the above subsections numbered:

Fig. 1: RTI-Mage System Build Flowchart

CHAPTER 4

# Prepping the dome

This first section will deal with prepping the dome for installing the LED lights on the inside. In a general sense, this boils down to two operations:

1. Marking the positions where the LEDs will be mounted evenly on the inside, so that they'll still be visible after you paint the inside of the dome.
2. Painting the inside of the dome matte/flat black, to minimize scattered light.

By now, you've hopefully read the previous section on how to select a dome, have ordered it, and now have it sitting before you, like this one :

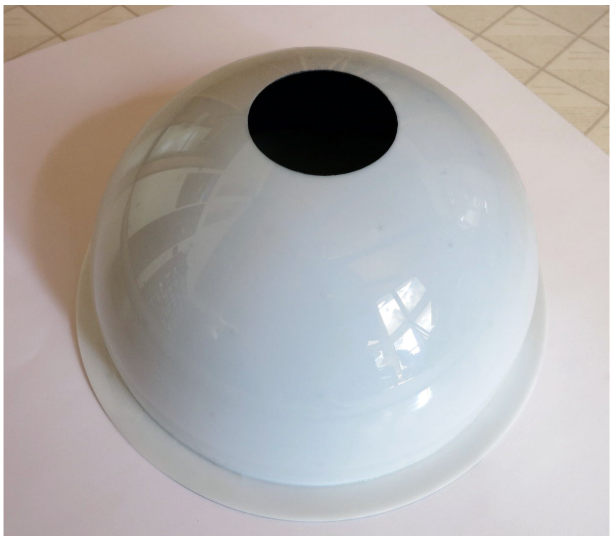


Fig. 1: Not exactly like this one - I've already marked the LED positions and painted the inside. But I'll use it as a model for this section.

When construction is completed, and all the LEDs installed, the inside of the dome will look something like one of these :

The LEDs are mounted in rows with shifted columns, with even rows lining up with even rows, and odd with odd; this is to maximize the variation in angular position. The 12" dome has 48 LEDs in 6 rows/8 columns, while the 18" dome has the maximum of 64 LEDs, 8 rows and 8 columns. 8 rows and 8 columns are the maximum, but you can have fewer than those.

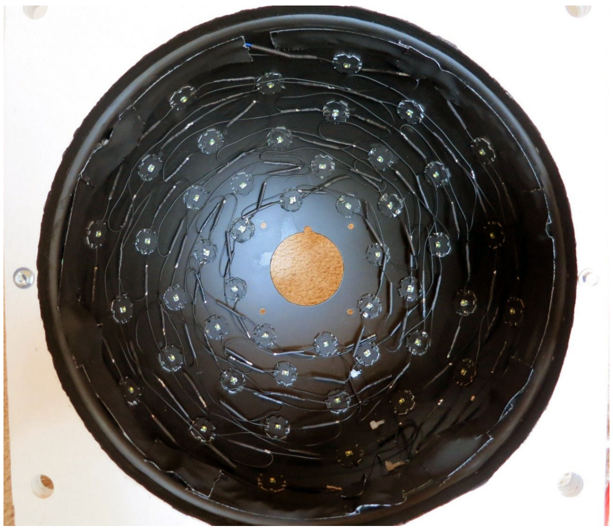


Fig. 2: Interior of 12” diameter dome, with 48 LEDs

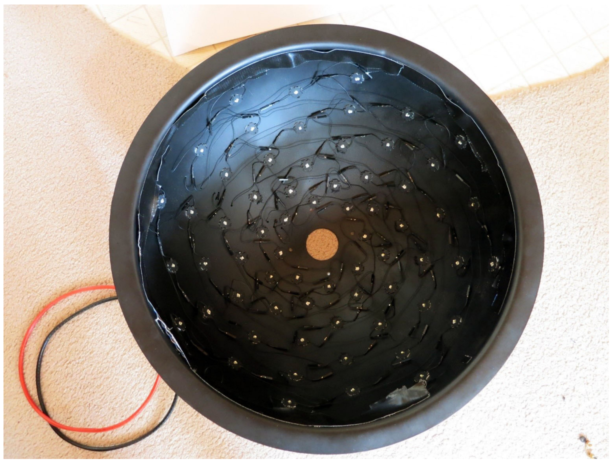


Fig. 3: Interior of 18” diameter dome, with 64 LEDs

It's possible to have rows with varying numbers of columns, but I don't recommend it - it makes wiring the system and managing the software a real pain. Plus, you want to have the LEDs spaced uniformly in angle for best results, and that can be tricky with varying column counts. So the first thing to do is figure out where the LEDs will need to be mounted inside the dome, and mark those positions. It's best to do this before you paint the inside of the dome black.

Before proceeding, a quick comment about accuracy. While getting the rows and columns laid out evenly will make assembly easier, don't worry too much about accuracy, as small errors won't make that big a difference in the results you get from the system. While the software needs to know the positions of the lights in order to do its calculations, it gets those positions from a calibration file that's generated from the actual light positions. So even if one or more lights are a bit "out of line" with others in their row or column, it won't make a significant difference in the results, since the calibration file will take that into account.

## 4.1 Determine and mark the angular positions of the columns

To find the angular spacing, just divide the number of columns into 360 degrees. So for 8 columns, the spacing would be 45 degrees; 60 degrees for 6. I wouldn't use fewer than 6 columns, and 8 is preferable to 6.

Place the dome on a sheet of paper large enough to allow you to trace a circle around the edge of the dome flange, then trace the circle.

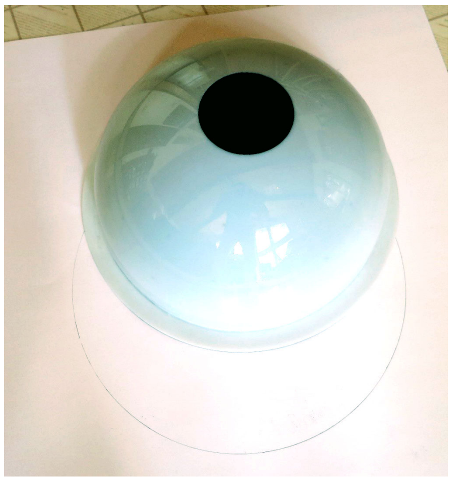


Fig. 4: Trace circle; dome obscures about half of it

Find the center of the circle you've just drawn. There are several methods you can use (ask Google), but this one is probably the simplest and cleverest.

Erase all the lines you've drawn except one, which you'll use as a reference. Using a protractor, and the reference line, mark the angles at the spacings you determined in A.1, i.e. 45 degrees or 60 degrees; 45 degrees for the picture below.

dome_angles.jpg Angles marked at 0/45/90/135 degrees

Draw lines through these marks extending out to the edges of the circle; make an accentuating mark (x, dot, whatever) where the line and circle intersect.

Place the dome back on the paper so that it is centered inside the traced circle. Mark the flange clearly at every position where the angle lines intersected the circle. You may find it tough to find something that can temporarily mark acrylic plastic. Sharpie permanent markers work quite well, and the mark can be removed by rubbing with a paper towel soaked in WD-40; other solvents may work, but test them first to make sure they don't dissolve the plastic. Bits of masking tape are another option.

Now take the dome and rotate it in the traced circle until one of the initial marks lines up with the mark created in step 6 above. Repeat the process of marking the dome flange at all the positions where the angle lines intersected the circle; try and differentiate this mark somehow from the first set (different color, put a slash through it, etc.).

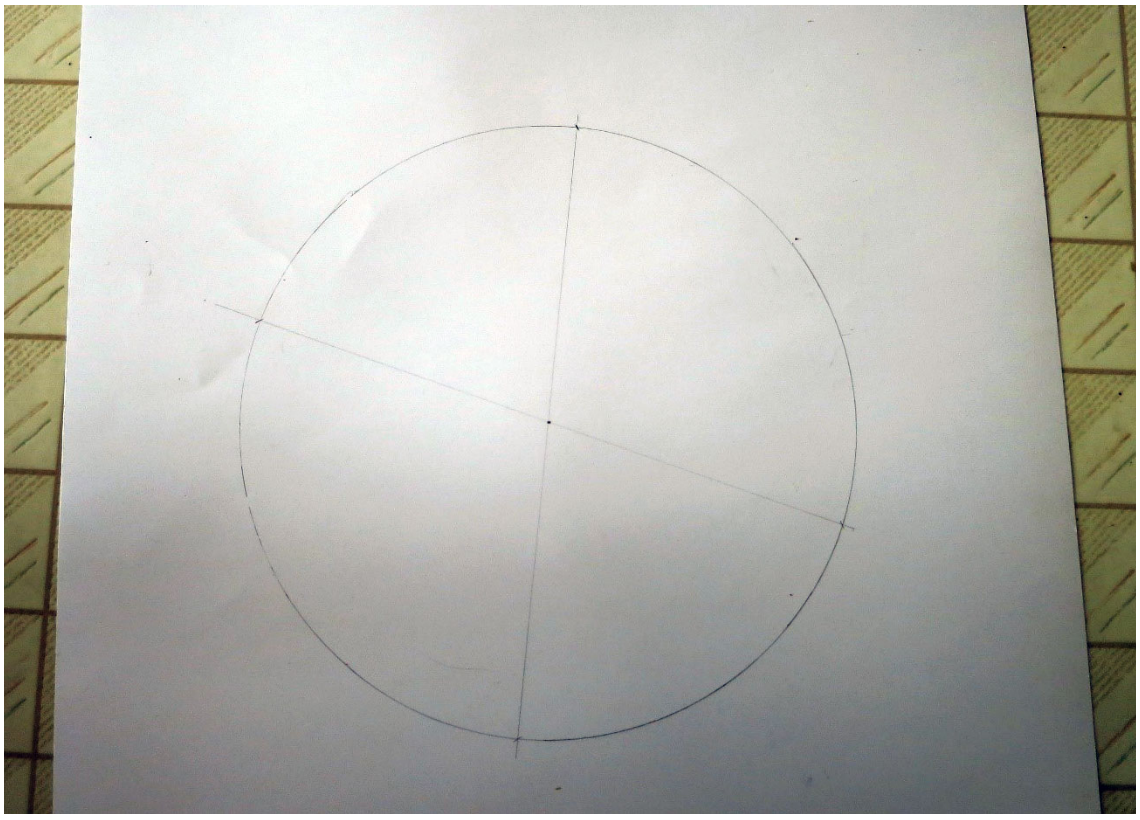


Fig. 5: Center found!

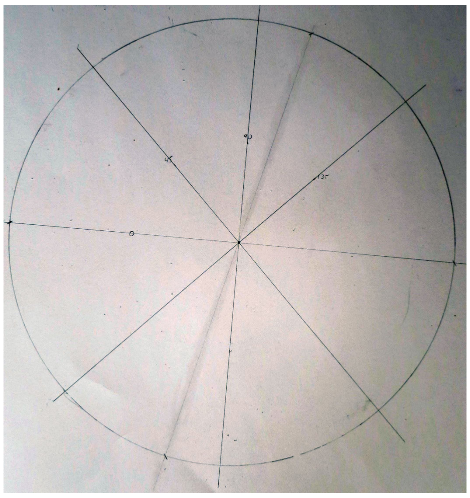


Fig. 6: Lines intercept edge of circle at 8 points (45-degree spacing)

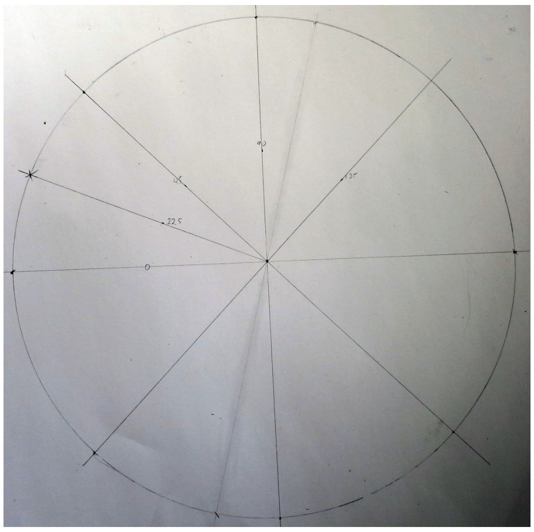


Fig. 7: Extra mark at 22.5 degree angle

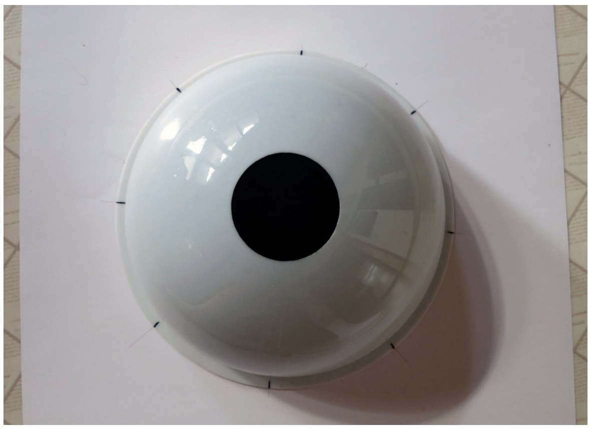


Fig. 8: Dome flange marked at 45-degree increments

You have now marked the angular positions for all the columns. Half the rows will have their LED columns lined up with the first set of marks, the other half will be lined up with the second set of marks.

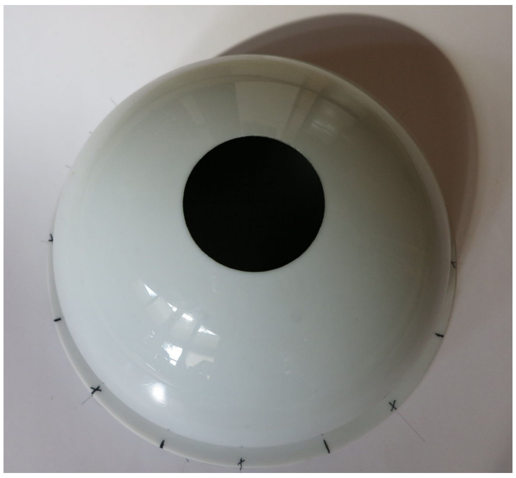


Fig. 9: Second set of marks spaced at 45 degrees, located between first set of marks

## 4.2 Determine the heights of the rows, and mark the LED positions on the dome

Using standard RTI guidelines, the lowest angle the LEDs should be mounted is at 15 degrees above the plane of the ground; below this lighting angle, the object you're photographing may be too dim. The highest angle should be somewhere around 65-75 degrees; above this, the LED lighting angle may make the object too bright. With your desired top and bottom angles, the number of LED rows you'll be installing, and the size of the dome, you can figure out the proper spacing on the rows.

### 4.2.1 Example 1

My big dome (18" dome diameter, 8 rows of 8 LEDs, 16 through 72 degrees). With 8 rows, there will be 7 angular gaps between the rows (8-1). The total angular distance between the low and high rows will be 72-16 = 56 degrees. The angular distance between individual rows will be 56/7=8 degrees. So there will be rows at the following angular positions (where bottom = 0 degrees):

- 16
- 24
- 32
- 40
- 48
- 56
- 64
- 72

To figure out the true arc distance up the side of the dome these angles correspond to, divide the angle by 90, then multiply by 1/4 the circumference of the dome. For an 18" diameter, the circumference is 18" x pi, or about 56.4"; 1/4 of that is 14.1". Divide the table above by 90 and multiply by 14.1", and you get (after rounding):

- 2.5"
- 3.8"
- 5.0"
- 6.3"
- 7.5"
- 8.8"
- 10.0"

- 11.3”

So these are the row distances above the base of the dome/flange junction. But the columns in adjacent rows will be shifted by half the angular spacing, so you should group the heights into two staggered sets of rows (each set has LED columns aligned with each other):

| Set 1 | Set 2 |
|---|---|
| 2.5” | 3.8” |
| 5.0” | 6.3” |
| 7.5” | 8.8” |
| 10.0” | 11.3” |

### 4.2.2 Example 2

The small dome (12” dome diameter (the size of the dome I’m building here), 6 rows of 8 LEDs, 15 through 65 degrees).

- Angular distance between high & low = 65 - 15 = 50.
- Angular distance between individual rows = High & Low / (# rows -1) = 50 / (6-1) = 10 degrees between rows.
- Row angles = 15, 25, 35, 45, 55, 65.
- 1/4 circumference = 12” x pi/4 = 9.4”
- Row distances (from bottom) = 1.6”, 2.6”, 3.7”, 4.7”, 5.7”, 6.8”
- Row distances paired by column alignment = 1.6”, 3.7”, 5.7” and 2.6”, 4.7”, 6.8”

**Warning:** If this isn’t 100% clear, just keep reading - hopefully after you see the actual marking operation, you’ll understand.

Now you need some way to mark these distances on the dome. Take a piece of string, make one mark on it to indicate the zero reference point, and then measure and mark positions for the first set of row distances on that piece of string (1.6”, 3.7”, 5.7” in this case). Create a similar second string for the other set of row distances (2.6”, 4.7”, 6.8”). Taping the ends of the strings down while marking them helps keep them straight.

Tape the first string to the dome at the base with the flange at one of the marked angles, so that the zero distance mark is at that base. Tape or hold the other end of the string on the top edge of the hole, making the string as perpendicular to the base as you can. Make marks on the dome next to the marks on the string. Repeat this for all matching marks with the same angular spacing as in A.1.

Repeat step B.3 with the second piece of string, on the other set of marked angles (the ones offset by half the spacing of the first set; these should lie exactly halfway between the first set of marks you made).

And this is how the dome should look when you’re done when viewed from above:

## 4.3 Transfer the marked positions to the interior of the dome

The marks on the outside of the dome correspond to the positions of the LEDs on the inside of the dome. Thing is, you’re about to paint the inside of the dome black, which will make it difficult to figure out what the corresponding position is. So, before you paint the dome, you’ll want to put some kind of mark on the inside that will remain visible even after you paint it.

There are several options:

1. Put a blob of paint or nail polish on the inside of the dome at the same position as the mark on the other side.

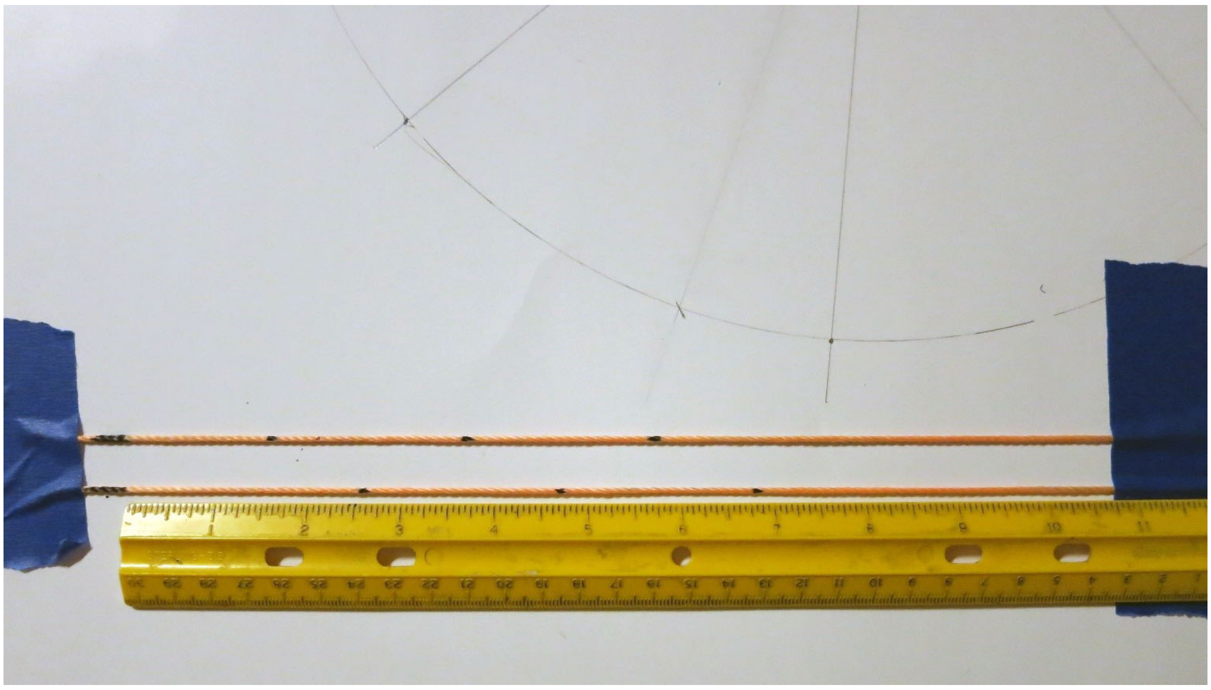


Fig. 10: Marking sets of distances on two string segments. The long marks at left mark the zero point on the right edge of the mark. These are distances for the 12" dome, Example 2.

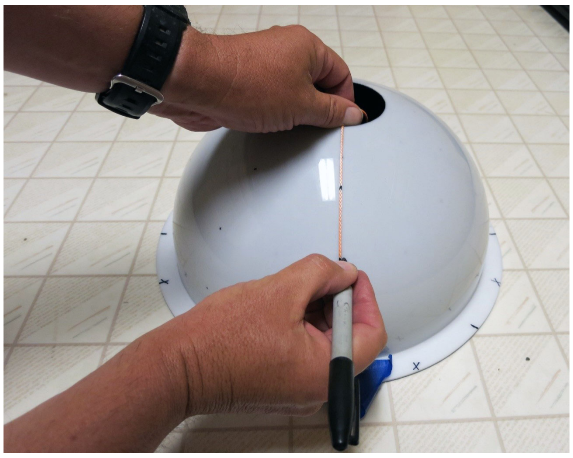


Fig. 11: Marking the first set of columns

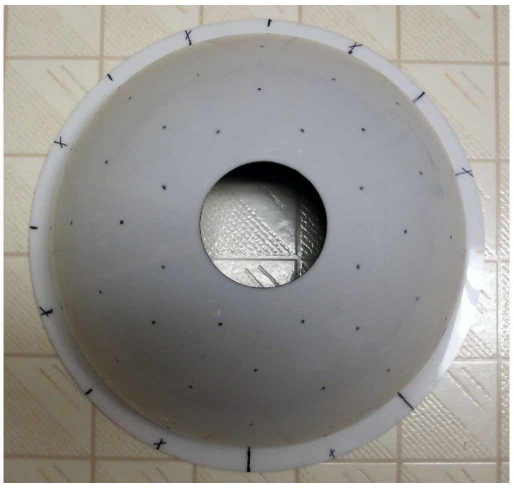


Fig. 12: 48 marks on this dome for 8 columns x 6 rows of LEDs

2. Drill or scratch a small dimple at the same position as the mark on the other side. Be careful with this approach - I used an electric drill with the dome picture above, and came close to punching through the dome in several spots. Do it by hand and you should be OK.
3. Put a tiny piece of masking tape in the matching position, and peel it off after you've painted the inside.
4. Got a better idea? Let me know!

This is why I recommend either a clear or white dome; both of those make it easy to see where the marks are on the outside of the dome when you're looking at the inside. A dark-colored dome makes this a lot tougher.

## 4.4 Paint the inside of the dome

To minimize scattered light, the inside of the dome should be painted flat/matte black. Up to now, all the domes I've painted have been done with spray paint. You can use a brand specifically labeled as being appropriate for plastic, with the color marked as "flat black" or "matte black" to minimize reflections and light scattering. Avoid "satin" or "glossy" finishes. My preferred brand is Rustoleum Flat Black Ultra Cover Paint + Primer ; Krylon will work, but seems to require more coats.

Just recently, I stumbled across another option for painting the interiors of the domes. This involves using a deep black chalkboard paint, sometimes mixed 50/50 with a roughening agent like flour, finely-sifted sand or poppy seed, to create a dark light-absorbing surface. I think this is probably a superior option to spray paint, especially for larger domes.

1. Make sure interior of dome is clean and dust-free; use soap and water if necessary, then dry completely. Instructions suggest lightly sanding the inside; that's not really necessary for spray painting, as long as the surface is clean, but is probably a good idea for chalkboard paint.
2. Cover the outside of dome with masking tape to protect it from stray sprayed or brushed paint. Pay special attention to the hole at the top, as that's the most likely place for paint to leak through.
3. Follow instructions on paint can.
   (a) For spray paint, apply paint to inside of dome in steady back-and-forth motion, overlapping strokes. Not too heavy, or you will get drips/runs inside the dome. You will almost certainly have to apply more than one coat to achieve full light blockage. Allow the first coat to dry for a few minutes, then apply a second coat. I'd even recommend a third coat if you have enough paint left. Let the final coat dry to the touch, then remove the masking tape from the outside. Hold the dome up to a bright light source (the sun works great for this), and check to see if you've put on enough paint to block all the light. If not, re-mask and put on another coat. The Rustoleum instructions say you should put successive coats down less than one hour before applying the previous coat, or after 48 hours.

   (b) For chalkboard paint, follow can instructions. If one coat isn't sufficient, you can apply a second coat 4 hours later.

4. Set the dome aside to dry. The instructions on the Rustoleum can say that it takes 5-7 days for the paint to fully bond with the plastic, but don't worry too much about that - you can work with the dome in a day or two if you need to, since you won't be putting any stress on the paint. Same thing with the chalkboard paint. But I'm putting these instructions first so that you can let the paint dry longer while you're working on the rest of the project.

That's it. Hopefully, you will see the LED position marks you created in Step C, like these drilled dimples in my dome, visible after painting:

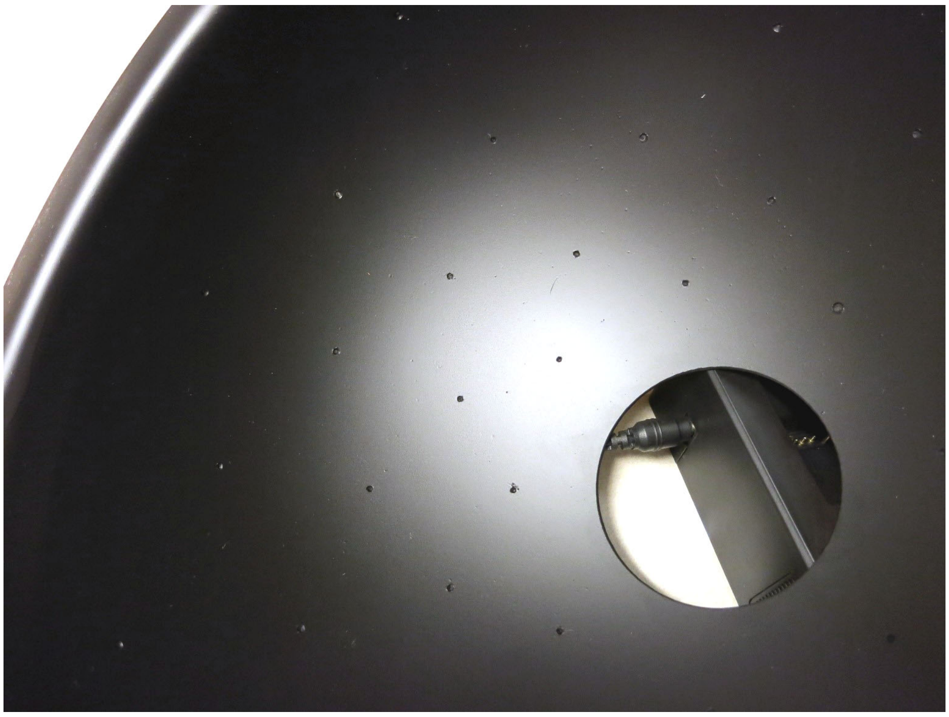


Fig. 13: Drilled "dimples" that mark LED positions in painted interior of dome. I made these too deep, to the point that some of them almost punched through the outside of the dome.

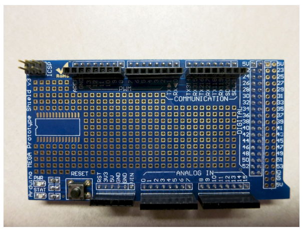
If some of the marks are indistinct or missing, use the marks on the outside as a guide to adding them on the inside.

Once you have clearly-marked LED positions on the inside of the painted dome, you can remove all the marks from the outside, including the flange marks. If you use WD-40, take care not to get any on the inside of the dome, and wipe off the dome exterior with a damp soapy cloth to remove the WD-40.

CHAPTER 5

# Building the MOSFET driver shield

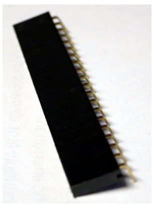
This instruction step documents the building of the high-side driver controller. This has the P-channel MOSFETs that control the plus-side voltage to the LEDs, through the columns of the LED matrix. These will be installed on the Arduino Mega Shield, which plugs directly into the Arduino Mega control board:



On the bottom are the pins that go into the female headers on the Arduino. This shield makes connections with the Arduino so easy and convenient, I had to use it. But there is one drawback: the limited amount of space you have for components. There is just barely enough room for every component needed for the MOSFET drivers, as long as you place them correctly on the board. I will show you where everything needs to go; modify these positions at your own risk.

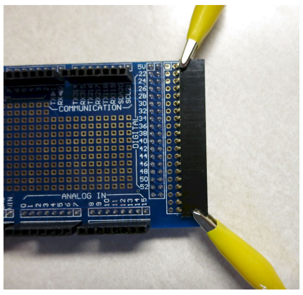
The first set of parts is easy, and is good warmup practice if you haven't done any soldering recently. You'll be soldering a female pin header to the right side of the board - this is where the protoboard with the minus-side current controllers will plug in and connect to the Arduino. Then, you'll be soldering eight resistors in that will eventually connect to transistors that control the P-channel MOSFETs.

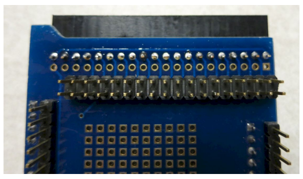
First, grab the 20-pin right-angle female header:



And stick it into the 20 holes on the far right side of the shield. Don't do what I almost did, and stick it in the holes to the left of the correct ones! If you've got something you can use to clamp it in place (binder clips, alligator clips), use those to hold it in place:

Now flip the shield over, and solder the pins at opposite ends. Once done, you can remove the clamps/clips and solder the remaining pins to the holes:

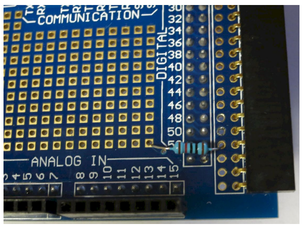




Next, you'll be soldering 8 10K resistors to the shield; these will eventually be connected to the bases of the 2N3904 transistors, which will in turn control the P-channel MOSFETs. The resistors are there to limit the current to the transistors – Arduino recommends no more than 40 mA of current from any of its output pins. One lead of each resistor will go into a hole immediately to the left of the header you just installed, the resistor will straddle another set of soldered pins on the shield (the ones marked 36-52), then most of them will have their other lead go into the holes directly to the left of the number labels.

When you solder the resistors, give them enough height so that they clear the soldered pins underneath, or you might have problems with shorting. If you like, you can put a thin strip of electrical tape on top of the pins, or paint the pins with nail polish or rubber paint, to make sure they don't short out the resistors.

Start by putting the leads of the first resistor into the shield, as in the picture below. Note the hole positions - on the left, the hole at bottom right on the shield, on the left, the 4th hole from the bottom:



Flip the shield over, and you'll see the two leads sticking out:

Solder the resistor leads to the shield holes they're in, and trim the outside lead, the one next to the connector. Put the next resistor into the next set of holes (above the first set), and repeat the process until you've done all 8 resistors. If you have trouble getting the resistors to stay in place when you flip the shield, you can bend one or both of the leads down to hold it more securely:

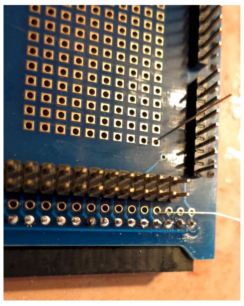
When you're done, the top of the shield should look like this:

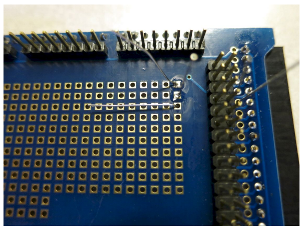
On the opposite side, leave the leads for the first and fifth resistors from the top uncut (you can bend them down to get them out of the way if you like), and trim the leads for the rest to a length of about 1/4":

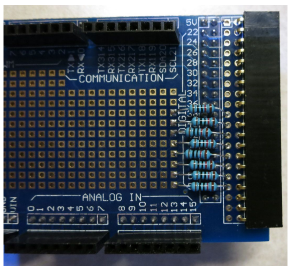
Now it's time to solder in the high-side driver channel components - P-Channel MOSFET, npn transistor, and 10K resistor. Here's a rough schematic for one driver. The reason for the complicated circuit is that the P-MOSFET is normally on when the gate is unbiased. This circuit biases the gate with +9-12V when power is turned on, turning the P-MOSFET off. When +5V is applied to the 2N3904 transistor, it shunts the voltage away from the MOSFET gate to ground, allowing the P-MOSFET to turn on.

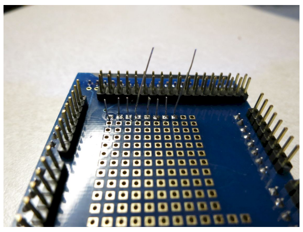
You've already installed the 10K resistor at left, though it still has to be connected to the transistor base (a future step). So this section will deal only with the MOSFET, the npn transistor, and the other 10K resistor. Here's a pic of the MOSFET; note that the position of drain and source in my schematic are swapped compared to the actual MOSFET (gate is correct):

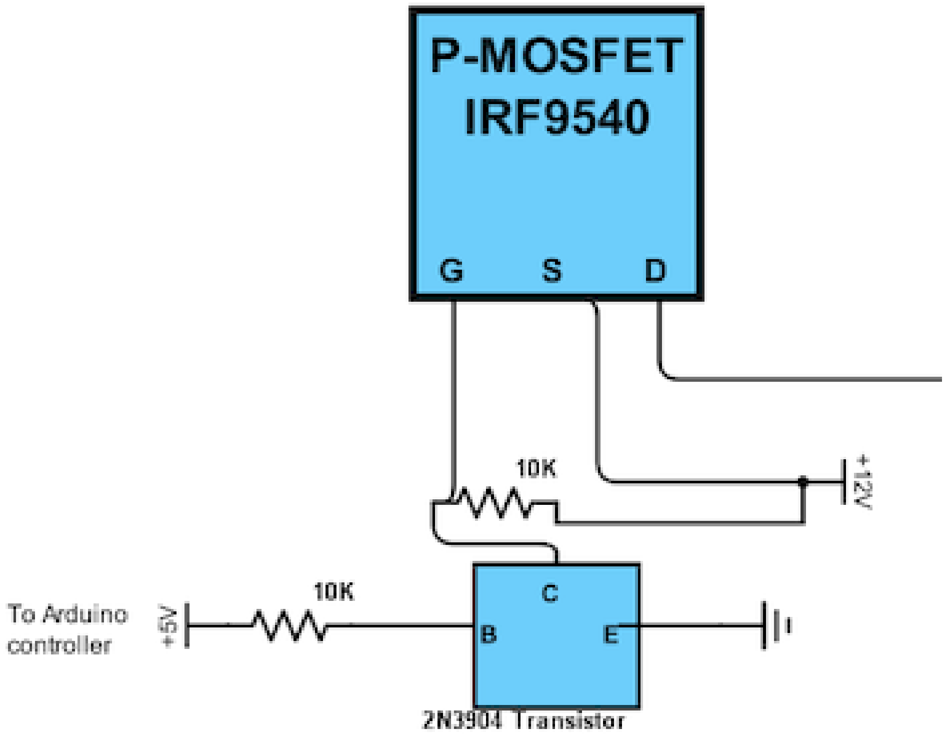
And here's a pic of the npn transistor, showing collector, base and emitter leads:

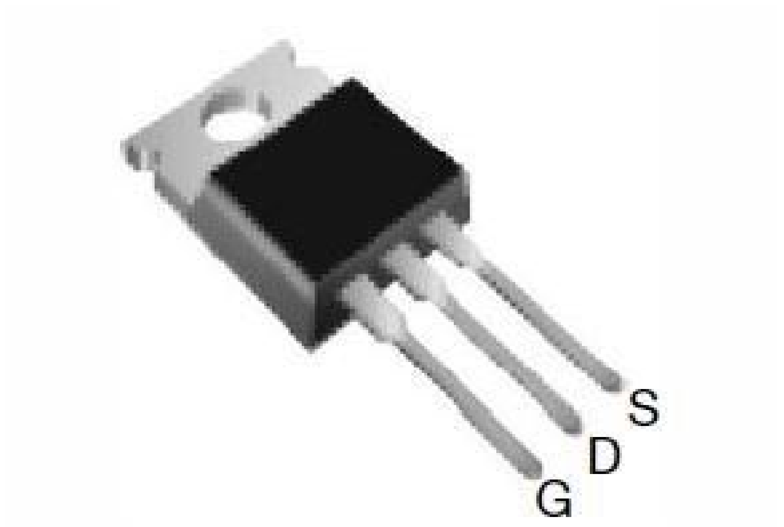


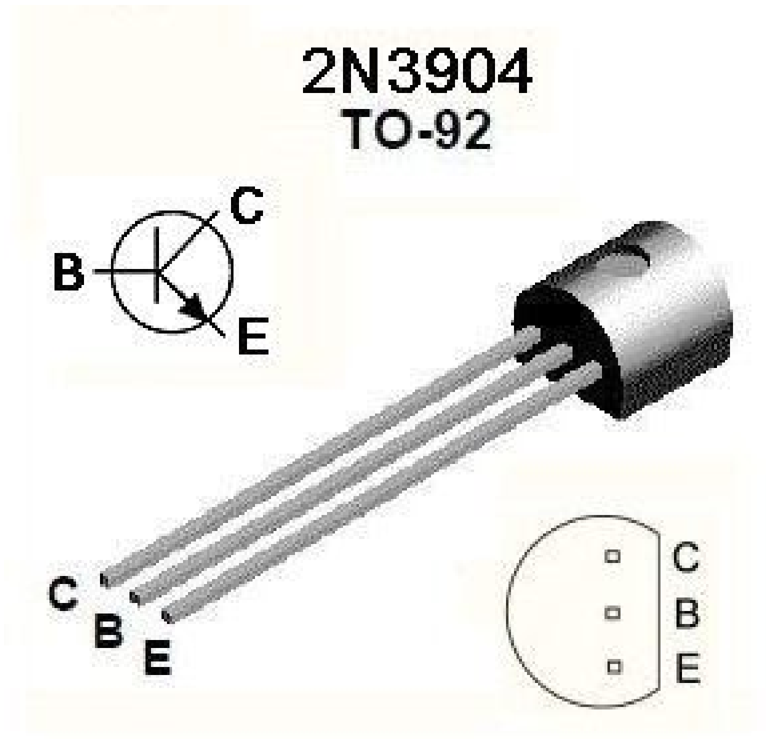



COMMUNICATION
DIGITAL
ANALOG IN




P-MOSFET
IRF9540
G
S
D
10K
+12V
10K
To Arduino controller
+5V
C
B
E
2N3904 Transistor

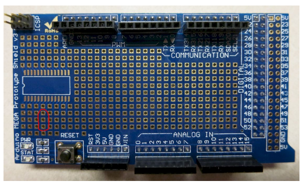

S
D
G



**Warning:** A few cautions before you begin. Some of these may seem obvious, but I'm including them because I've either come close to making all these mistakes, or have actually made them and had to go through the pain of fixing them.

1. MOSFETs are notoriously sensitive to static discharge. Make sure you've discharged any static buildup before handling MOSFETs. If you have a grounding wrist strap, now would be a good time to put it on.
2. Make sure you install both transistors in the correct orientation.
3. Remember that when you flip the shield over, what was on the left is now on the right, and vice-versa; make sure you're making the right solder connections.
4. When trimming leads, take care to make sure the trimmed lead doesn't wind up wedged someplace where it can cause a short. This has happened to me several times, and can drive you crazy as you try to figure out why things aren't working.

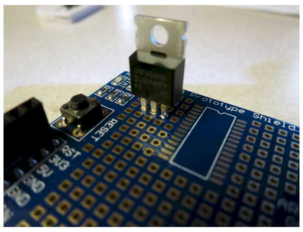
Here's a picture of the shield board, with the location of the first MOSFET circled in red:

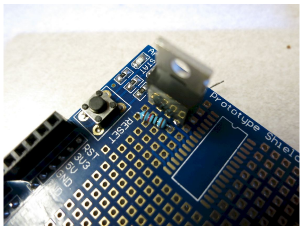


Insert the MOSFET into the marked pins, with the black labeled side facing to the right of the shield, and the metal backside facing left; solder one pin in place to fasten it to the shield:

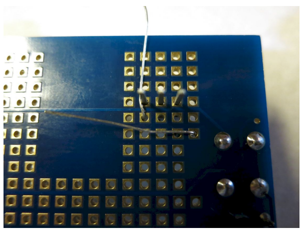
It can be tough to solder it upright - what I usually do is solder it in place at any angle, then re-melt the solder while pushing he MOSFET until it's perpendicular. Careful - it can get hot.

Next, the 10K resistor is inserted as shown in the picture below, with one lead in a hole right next to the source pin, and the other lead two rows up and one row to the left of the gate pin. Pull the leads tight from the underside so that the resistor is flush with the shield board, using needle nose pliers if necessary.





Flip the shield over, and bend the resistor leads so that the one closest to the source pin is flat and next to the source pin, while the other lead is bent parallel to the MOSFET pins:



Now it's time for the 2N3904 npn transistor. The lead spacing on this resistor is smaller than the hole spacing on the shield, so you'll have to first gently spread the leads further apart until they will fit into the shield. Then insert the transistor with the flat part facing the MOSFET, and the curved part facing toward the right side of the shield:

Note that the three npn transistor leads are in the same rows as the MOSFET leads.

Flip the shield over, and bend the collector pin down over the resistor lead and next to the MOSFET gate pin (better than I did in this pic):

Now solder the collector pin on the npn transistor to the gate pin on the MOSFET; the resistor lead under the collector pin to the collector pin; and the other resistor lead to the source pin:

Trim off the excess lead on the resistor leads, and on the collector lead:

And that's it - you've created a high-side p-channel MOSFET driver circuit! Hope you enjoyed that, because you'll now have to repeat the process 7 more times for the remaining high-side driver circuits.

Here's a picture showing the locations of all the MOSFETs, including the one you just did:

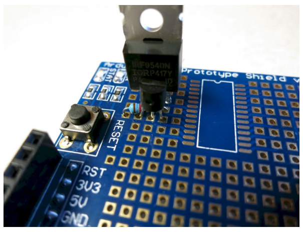
Just follow the exact same steps as the first driver circuit for all the successive ones. It's best to work left to right on the shield when installing the driver circuits, as it gives you the easiest access to the board when installing parts. When it's all done, the shield should look like this on top:

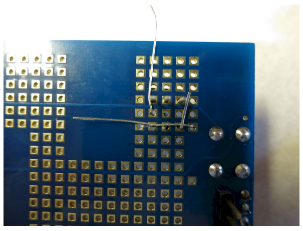
And like this underneath:

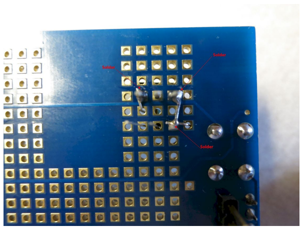
Next is a simple step - adding several female headers to the shield, for wiring power and ground connections. But I screwed up the first time through. I soldered two 2-pin female headers to the board for power connections in the positions indicated below (circled in red):

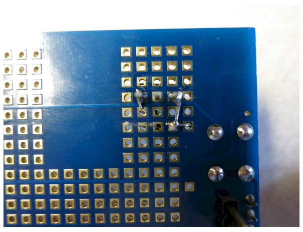
But I screwed up here – the top header keeps the shield from properly fitting into the Arduino Mega. So I removed it at a later step, but too late to fix most of these pictures. I've crossed out the errant header and associated soldering steps in a few upcoming pictures - just don't do them. Some later pics may not have that header crossed out – just ignore it.

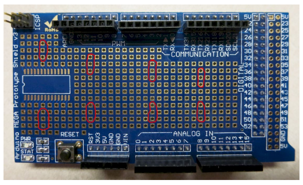
So here's the modified pic, with the header to be omitted left out:

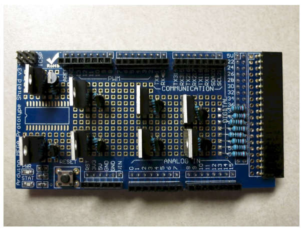


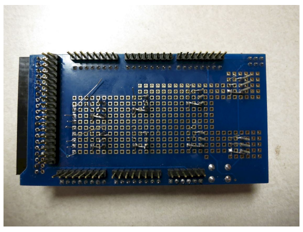


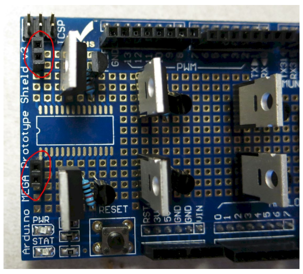


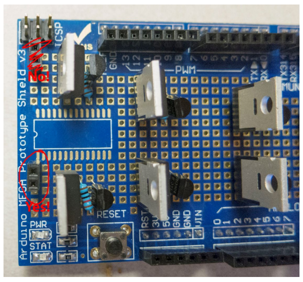






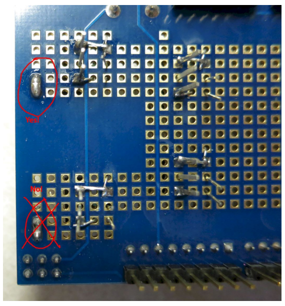






It's tough to hold these in place for soldering - I usually use masking tape on the top to hold the header in place when I flip the board over.

You want the two sockets on the header to be connected electrically - the simplest way to is put a blob of solder between the pins on the bottom of the board:



This header will be used for connecting "high" DC voltage (8-12V) to the shield, for powering the 3W LEDs.

Next, solder a 2-pin female header in the location indicated below, for a ground connection:

As with the previous headers, you also want the two pin sockets to be connected electrically with a solder blob:

This connector will be used to add some extra ground connections; the ones already present on the Arduino aren't sufficient, and also aren't convenient for several future connections.

Finally, an 8-pin male header strip needs to be soldered in place; this will be used to connect the high-side drivers to the columns in the LED matrix. IMPORTANT: On this one, you will NOT be bridging pins with solder blobs, but keeping all the pins electrically separate from each other. Also, don't solder all the pins at once – first do only two pins to hold it in place, one row in from each edge, then do the rest.

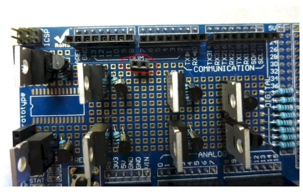
From the 40-pin male header strip listed in the components section, break off an 8-pin section (they're notched to break easily). The photo below shows it in place, but I put it in one row higher than I probably should have. It will work in this position, but you will have a bit more room to work with if you move it down one row, closer to the "Vin" label at the bottom.

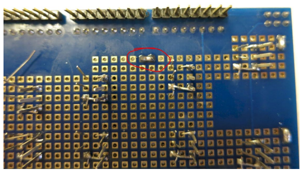
Here's a side shot:

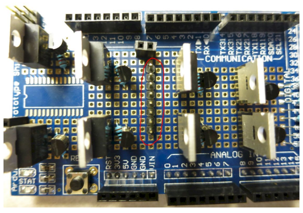
And the view from below, showing the two pins soldered to the shield, to hold the pin strip in place. Continue soldering all the pins to the holes they're in.

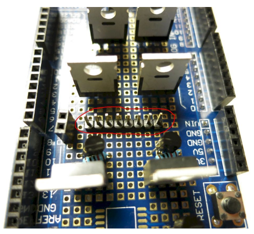
Now comes one of the toughest sections of work on the whole project - wiring up the connections on the MOSFET driver shield. It's not too hard, but it is a bit tedious. A few pieces of advice:

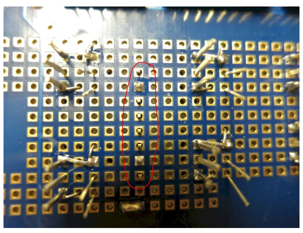
1. Work slowly and carefully.







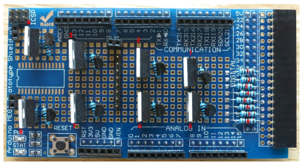




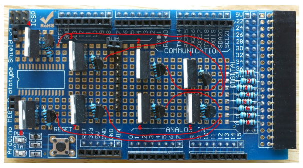
2. Make sure you're wiring up the right connections, i.e. connecting the wires to the proper component parts. A mistake here shouldn't be fatal, but it would be a huge pain to repair (believe me, I know :-/).
3. Make solid solder joints that only connect the right parts. Watch out for solder "bridges", where a bit of solder connects two leads/components that shouldn't be connected. Also watch out for loose trimmed leads that could lead to shorts.
4. Inspect your solder joints when done with a magnifier to make sure they're good.
5. Don't make your wires "just fit" - add some slack so that you can re-arrange them later on. As you'll see in a few steps, I didn't do that, and I wound up with a couple of spots where the wires blocked some holes where I wanted to solder a wire, and I didn't have enough slack to move the wires to free up the holes. I had to modify my wiring configuration on the fly to make up for that, and given the limited space I had to work with, that was not fun.

**Note:** In schematic wiring drawings, the wire lines are drawn from one hole on the shield to another. The wires connecting these holes should have the insulation stripped from their ends, and have the bare lead inserted into the hole. The bare lead should then be connected to the specified nearest component/connector lead.

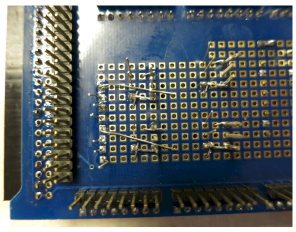
First set of wiring will be to connect the base of the npn transistor (the central lead of the D-shaped thingies) to the 10K resistors on the side of the board; these will control turning on/off the main p-channel MOSFETs that run to the positive (high) leads of the LEDs. The numbers next to the resistors are the LED column they control, and need to be connected via wires to the npn transistor next to the correspondingly-numbered MOSFETs:



Here's my recommended wiring layout, with one end of the wire to be soldered to the resistor, and the other soldered to the central lead of the transistor:



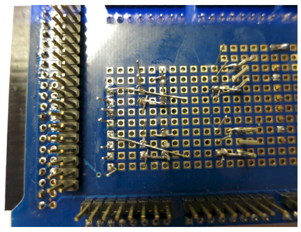
You'll notice that there are no wires coming from resistors 1 and 5 to their corresponding transistors. That's because in step 2, we left the resistor leads untrimmed for those two resistors, and we can now attach them directly to the nearby transistors using those leads on the underside of the board. Just bend them down to make contact with the central npn transistor lead, as shown below:

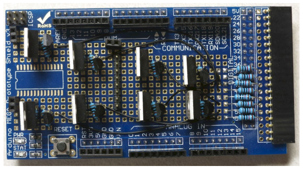


Now solder the connections; trim off the excess lead when done:

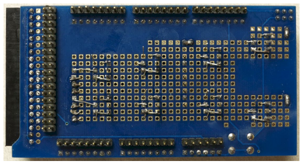
I usually play it safe now by blocking the hole next to resistors 1 and 5 with a bit of solder, to make sure I don't accidentally put a wire into those holes (which I have done; multiple times :-/).

Now follow the wiring diagram above, and connect the remaining resistors to the matching transistors with the 24 AWG Kynar wire. When the connection is soldered, trim off the excess leads. Once you're done, it should look something like this from above:





And this from below:



Next step is to connect the drains of the MOSFETs (their center leads) to the central 8-pin connector; there will ultimately be a header that plugs into this connector, connecting the output of the P-Channel MOSFETs to the LEDs.

As you'll see in succeeding steps, I didn't do a very good job on the wiring. Two big fails on my part: I didn't give the wires enough slack, and I kept bending them flat against the board to take up less space and look neater. As a result, access to some of the holes on the shield became very difficult, and I wound up having to use a number of workarounds to finish the wiring. And all for no good reason - wire is cheap, and since all the electronics will sit inside an enclosure with lots of room, it doesn't matter how neat it looks (no one will see it), or how much space I saved (there'll be plenty of space to spare). So use the following as a general guide, but be generous with the wire, and don't bend the wires flat to the board like I did unless you absolutely have to. And don't think you have to follow the wiring layout exactly as I give it - as long as the wires are soldered to the correct leads, they will work (you'll see an example of this shortly).

So here's a pic with the MOSFETs numbered, and the matching pins on the central connector numbered:

You want to connect the central drain lead on the MOSFETs to the corresponding pin on the central connector:

Here's how it looks after the drain wiring was completed:

Most of the wires were connected per the wiring plan above, but you will notice that the wire for MOSFET 8 (lower left) does not go into the hole in the wiring plan; there was another wire from a previous step that interfered with it. So I had to move that wire to the hole on the opposite side of the connector, but since I soldered it to the correct pin, it will work fine.

Here's a view of the bottom of the board after this wiring step:

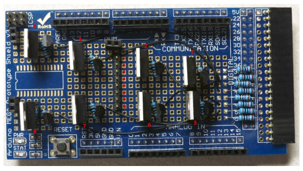
You can clip off the central drain lead of the MOSFET now to free up some room (that's the MOSFET leads you soldered the wires to in this step). You may need that room to maneuver in - there are still two more sets of wires to solder in place.

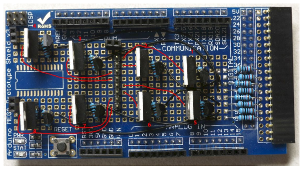
Next step is to wire all the source leads on the MOSFETs together (the right lead when you're looking straight on at it), and also connect them electrically to the 2-pin female header on the lower left side of the board (which will be connected to the positive lead of the power supply):

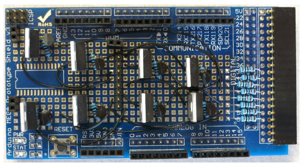
The easiest way to make the connection between the MOSFET at lower left and the female header is to bend the source leads of the lower left MOSFET:

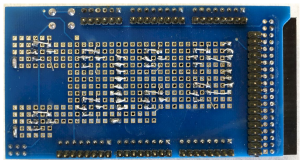
So that they come close to the soldered female headers (on the back side of the shield board):

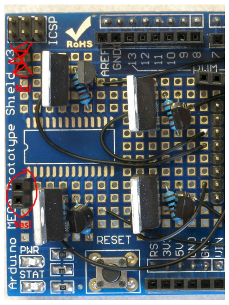
Then glob solder between that bent lead and the solder blob for the headers to connect them electrically:

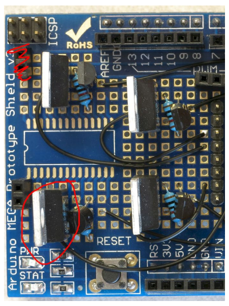


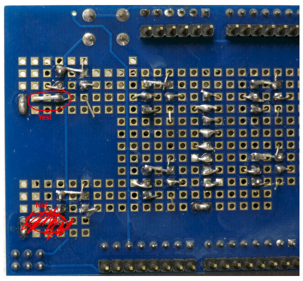


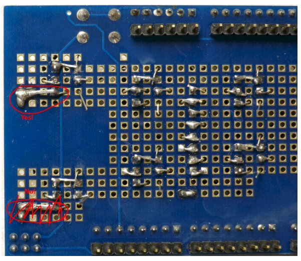




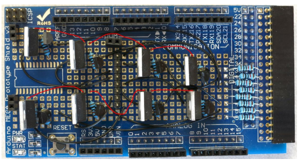




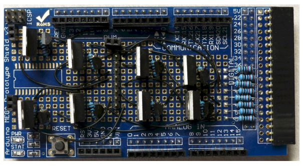




For the rest of the source connections, just wire jumpers between MOSFET connections in the same row:



So here's how my board looked after I wired the source jumpers:



Notice anything missing? Yup, there's two jumpers missing from the shield, on the bottom row. As I mentioned earlier, I pressed the wires flat, and didn't leave enough slack, so I wound up with a situation where I didn't have enough room to get access to the holes from the top. I wound up soldering jumper wires between the MOSFET source leads on the bottom of the board (circled below):

This works, but it's not very neat. What's more, I made the jumper connection on the right in the above picture too short, and blocked one of the last solder connections I'll need to make in the next step. Once again, I emphasize - leave yourself lots of slack in the wires so that you can move them out of the way if necessary for subsequent wiring/soldering steps.

Because I removed one of the two-pin female headers, I also had to add a jumper from the remaining two-pin header on the left to the source pin of the top left MOSFET; the wiring connection is paralleled with the red line:

And here's how those connections look like on the bottom (solder points for the jumper are circled in red):

Last step in the MOSFET driver shield board construction is to connect all the emitter lead on the npn transistors to ground. The npn transistors are the small D-shaped transistors, and the emitter is the lead furthest to the right when looking head on:

The best way to minimize the amount of wiring on the board (which is already crazy with wiring) is to connect the emitter lead closest to the ground connection header to that header:

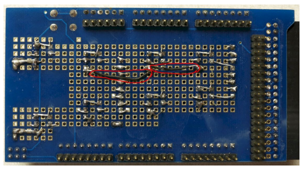
Then use jumper wires to connect all the remaining emitter leads to each other. Because of the limited space on top (thanks to my poor wire management skills), I didn't think there was room on top to do this, so I did the emitter ground wiring on the bottom. First I bent the emitter lead to make contact with the ground header solder blob, and then attached a jumper wire to that junction:

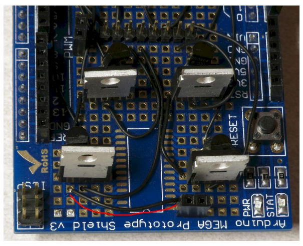
I then soldered that connection so that the emitter lead, jumper wire end, and ground header solder blob were all electrically connected, and starting adding successive jumper wires to connect additional emitter leads (taking care to make sure they didn't short out any other connections):

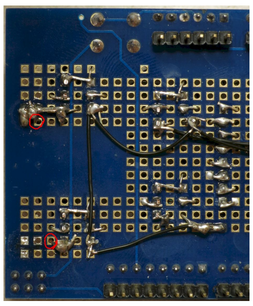
When all the jumper wires were in place, they were soldered to connect them to the emitters (though you can just as well solder them one at a time):

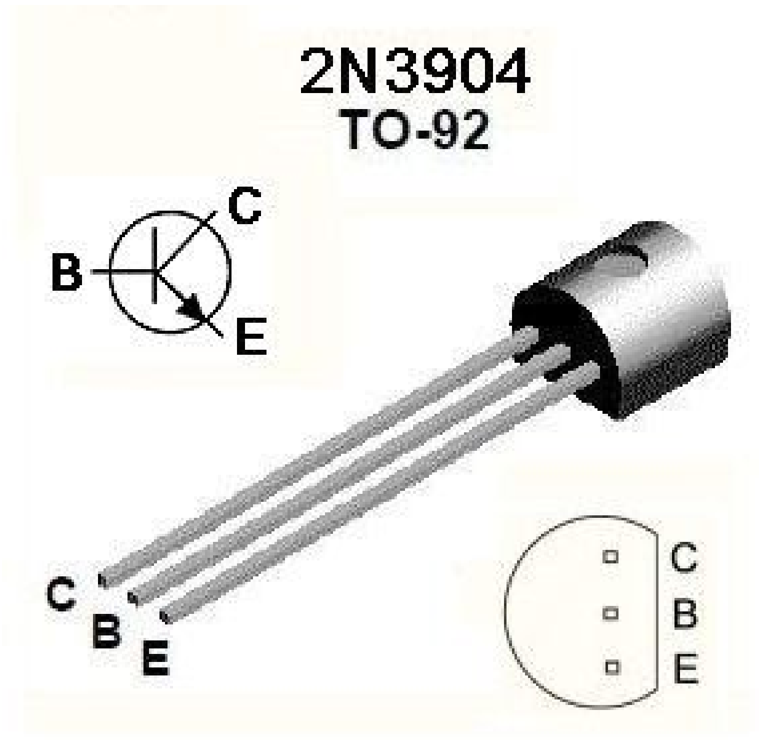
I've highlighted all the jumper wires with an adjacent red line, so that you can see exactly where they go.

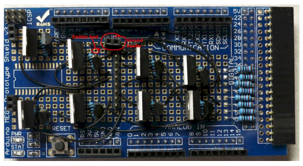
Once you're done with soldering the emitters, you should cut all the protruding leads (transistors, wires, whatever) as flush as you can without affecting the soldered connections.

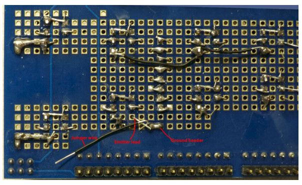


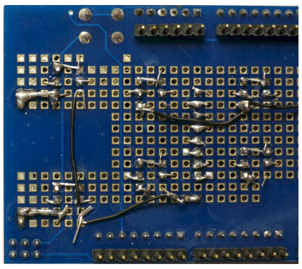

RESET
PWR
STAT
Arduino MEGA Prototype Shield v3




2N3904
TO-92
C
B
E
C
B
E
C
B
E







You're now done with the toughest system assembly steps, tough because of the tight space limitations. There's more wiring /soldering coming up, but it should be easier, as you'll have a lot more room to work with.

In the past, I've always assumed that I've wired the MOSFET driver shield correctly, and that's usually been the case (with a few minor shorts due to stray clipped leads). But for these instructions, I came up with a fairly simple way to test all 8 driver circuits on the driver board, to make sure they work correctly.

You will need:

1. Arduino Mega and wall wart power supply
2. Multimeter with either alligator clip leads, or separate alligator clip leads
3. One female Dupont pin
4. Jumper wires. I use breadboard connectors, but you can cut a few wire segments from the 22 AWG wire on the parts list (just save it when you're done).
5. The driver shield you've just finished assembling.

Here are all the parts in a neat assemblage:

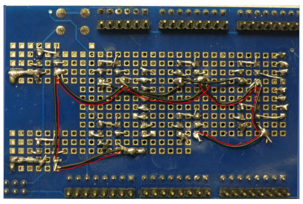
The Arduino will only be used for supplying +5V and ground, nothing more. So you'll connect one wire to a +5V connection on the Arduino, and the other to a ground connection:

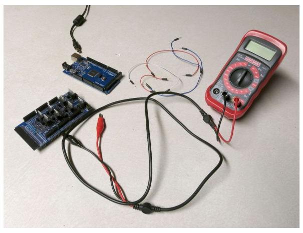
Note that the power supply isn't connected yet - I save that until all the wiring connections are done. I'm using standard breadboard jumper wires here, red for +5V and blue for ground, but you can use sections of 22 AWG wire. These wires should now be connected to the driver board, along with the remaining jumpers.

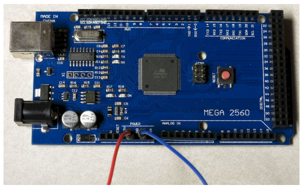
Because I removed the top left header, there's no red jumper between the two headers as seen in the photo, and you should plug the white wire equivalent into the open socket in the lower left header:

Here's an annotated connection pic:

The idea is that when the white wire (at +5V) isn't touching any of the 10K resistor connections on the right, there's no voltage across the multimeter. When you touch the white wire to one of the resistors, you switch on the npn transistor, which "effectively" turns on the MOSFET transistor (technically, you're unbiasing the p-channel MOSFET gate, which turns it on). You should then see 5V on the multimeter, which is measuring the output voltage from the MOSFET driver. In the pic above, the red (+V) multimeter lead is connected to the #1 MOSFET output driver connection, while the blue multimeter lead is connected to ground.







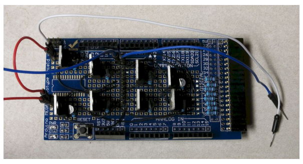
I tried connecting the multimeter lead directly to the output pins in the center, but the spacing is too close. So I took a female Dupont header pin, spread the clamps apart, then clipped it to an alligator clip:

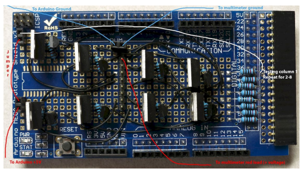
The female Dupont header slides easily onto each of the 8 pins on the output connector, so that you can test all 8 circuits sequentially.

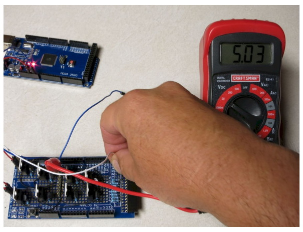
Plug in the Arduino to the power supply now. Here's a pic of me testing MOSFET drive transistor 1 by holding the white bias wire to the end of the top 10K resistor; the red multimeter lead is connected to pin 1 (the top pin) of the 8-pin male connector in the center using the Dupont female header. You'll see that the multimeter, set at the 20V DC scale, shows about 5V when the connection is made, and should show zero when there's no connection:

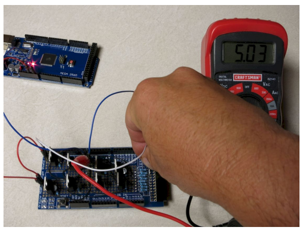
Now I connect the multimeter red lead to the 2nd pin down, corresponding to MOSFET driver 2, and do the same test by touching the second resistor down:

You should only see 5V on the multimeter when the white wire is touching the resistor that corresponds to the output pin. It's worth testing this by touching every resistor when you have the red multimeter lead connected to a single output pin - only when the resistor that corresponds to that output pin is touched should you see 5V on the multimeter. If you see voltage on more than one resistor, you have a wiring problem (electrical short, wrong connection, etc.), and you'll have to double-check your work to find out where you went wrong. But hopefully, everything will work fine (as it did for me in this case). Repeat the process for all 8 output pins. Once you're done testing, set the driver shield assembly aside somewhere - you won't need it for a while.




To Arduino Ground
To multimeter ground
Jumper
Testing column 1
Repeat for 2-8
To Arduino +5V
To multimeter red lead (+ voltage)


5.03


5.03

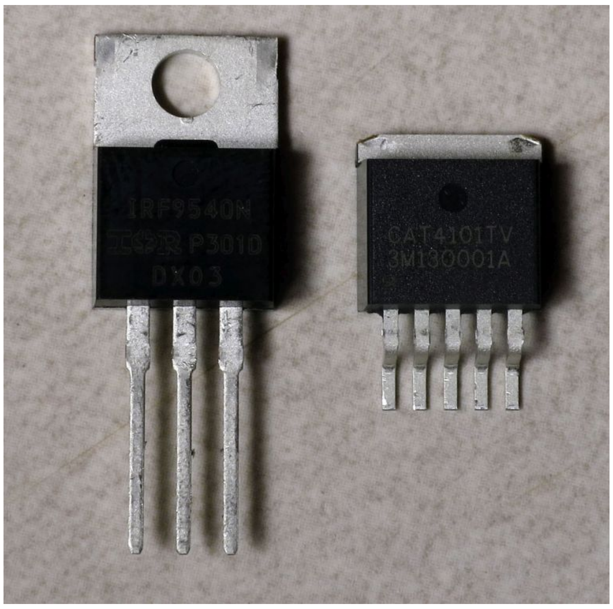
CHAPTER 6

# CAT4101 board

This board controls the “low side” (ground connection) for the LED matrix.

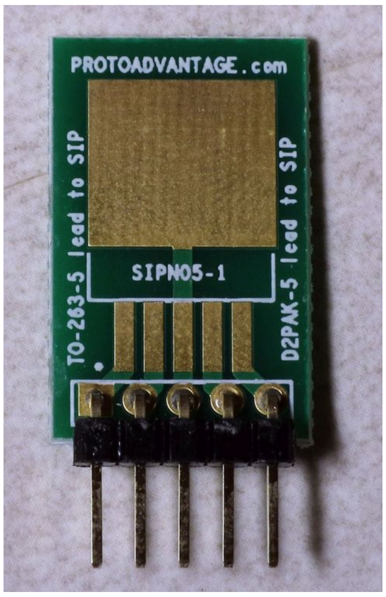
Most of the components used for building the RTI controller are “through-the-hole”, which means that they can be installed and soldered onto a standard protoboard/shield with 0.1” spacing between holes (2.54 mm). These are a lot easier to install and solder than the other common option, “surface mount devices” (SMD). The latter are designed for easier mechanical placement and bulk automated soldering. You supposedly can hand-solder SMD devices, but I am apparently too incompetent to do this well.

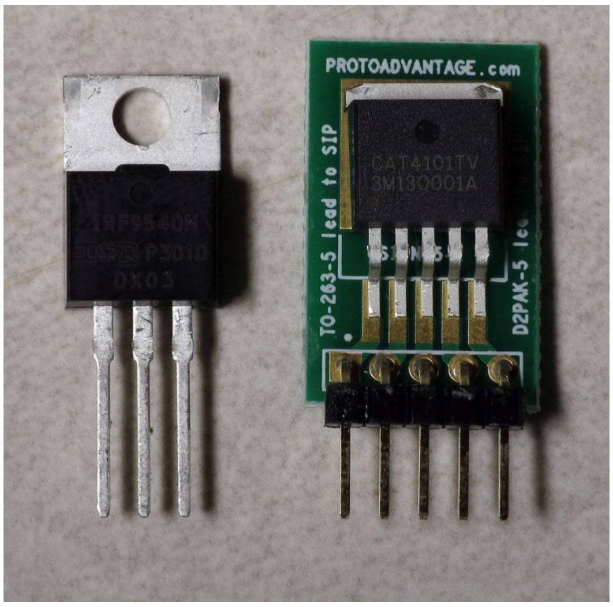
Unfortunately, there is one set of RTI controller components that is only available in SMD format - the CAT4101 LED drivers. Here’s a pic of a standard MOSFET transistor on the left with 0.1” lead spacing, and a CAT4101 on the right; note the difference in lead spacing:



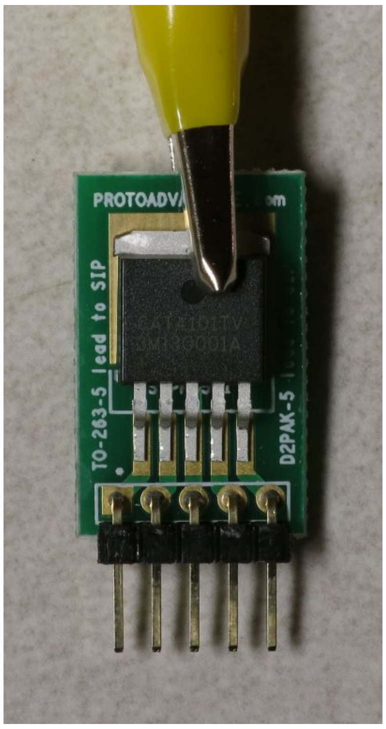
Fortunately, there’s something called a “breakout board” for the style of chip package used in the CAT4101 (technically, a TO-263 5 lead package). The breakout board has leads with the standard 0.1” spacing, and you solder the chip to pads that connect electrically to the leads. Here’s one example of such a breakout board:

And here’s the two chips again, this time with the CAT4101 sitting on the breakout board:

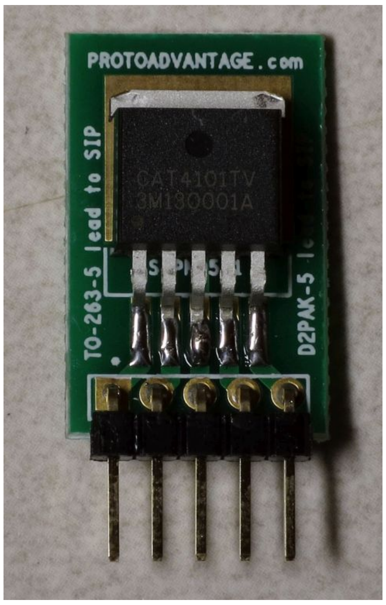

PROTOADVANTAGE.com
TO-263-5 lead to SIP
D2PAK-5 lead to SIP
SIPN05-1


PROTOADVANTAGE.com
CAT4101TV
3M130001A
IRF9540N
TO-263-5 lead to SIP
D2PAK-5 lead to SIP

You still have to solder the CAT4101 chip to the breakout board, though. I use an alligator clip to hold the CAT4101 on the breakout board while soldering the first one or two connections:



To solder the leads, the technique that works best for me is to hold the tip of the soldering iron touching both the gold pad on the breakout board and the end of the chip lead. Then apply the solder carefully to the end of the lead, and it should flow smoothly along the length where the lead touches the pad. Don't apply too much solder, or it may blob up and connect adjacent pads/leads, requiring cleanup and a re-do. When you're done, it should look something like this:



Double-check to make sure there are no shorts between leads. Do this for all 8 CAT4101 chips.

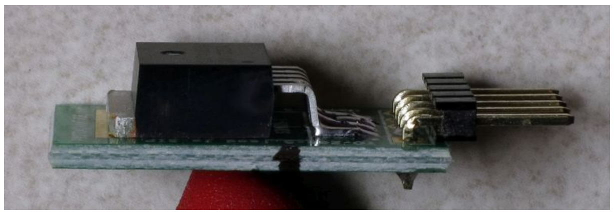
**Note:** You'll get the CAT4101 chips in a hermetically-sealed envelope with a desiccant inside, including a note that tells you to keep them sealed until use, with limits on temperature when assembling. While you should certainly keep them in the sealed package until you're ready to solder them, these precautions are primarily for

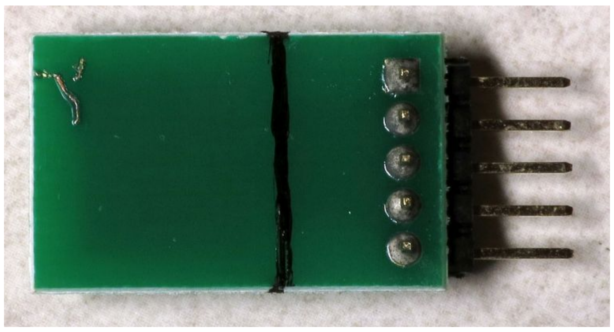
SMD assembly. In SMD assembly, the entire part is stuck into an oven that melts a solder paste to attach the CAT4101 chip to a PCB board / heat sink, so the entire part gets hot.

There's an encapsulant inside the chip that can absorb moisture from the air, and if it absorbs too much before heating it can "popcorn" from the steam, swelling up and breaking the chip package. Unless you hold the soldering iron on the chip for a prolonged period of time, it's almost impossible to "popcorn" the chip with hand-soldering.

---

Now that you've got your 8 CAT4101s neatly soldered on the breakout boards, it's time to do some cutting. While my experience has been that the CAT4101s never get warm enough to justify heatsinks, I prefer to play it safe and put heatsinks on them just in case. But to attached the heatsinks, you'll need to cut off part of the breakout boards to access the bottom of the chip. Mark each breakout board as shown below, above the chip leads and below where the bulk of the chip lays on the breakout board:





Now saw/cut the breakout board on the back to remove the upper part. I've used both a hacksaw blade and a Dremel with a cutoff wheel to do this; any fine saw blade or cutter will likely do just as well (e.g. jeweler's saw). You don't have to cut all the way through - if you get about 2/3rds through, you can usually bend the board to break it off. There will be one metal connection running to the upper part of the board, but don't worry about that - bend/cut it off to detach it.

**Warning:** Be careful in this step not to cut through either the chip leads or the chip itself. While you're at it, please take care to not injure yourself. Safety glasses are mandatory, and I use a dusk mask to keep from breathing in powdered PCB.

When you're done, you should have a nice pile of CAT4101s on trimmed breakout boards, and the trimmed part of the boards:

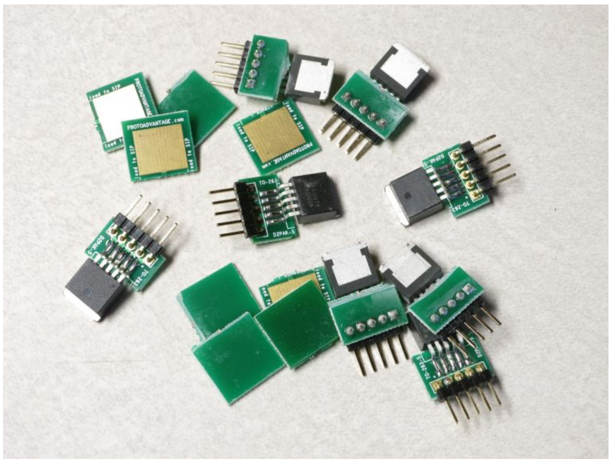
Throw out the trimmed board pieces, and put the CAT4101/breakout boards someplace safe for now. Try not to put any sideways pressure on the chips relative to the boards, as the leads can bend. I've bent many of these, and bent them right back without problems, but I'm sure they can break if bent far enough, or enough times.

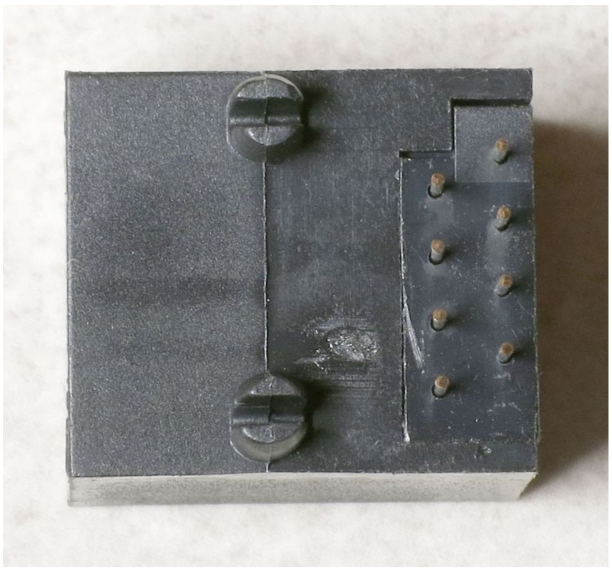
One more component requires a breakout board, and that's an RJ-45 (Ethernet) jack. Here's a bottom view:

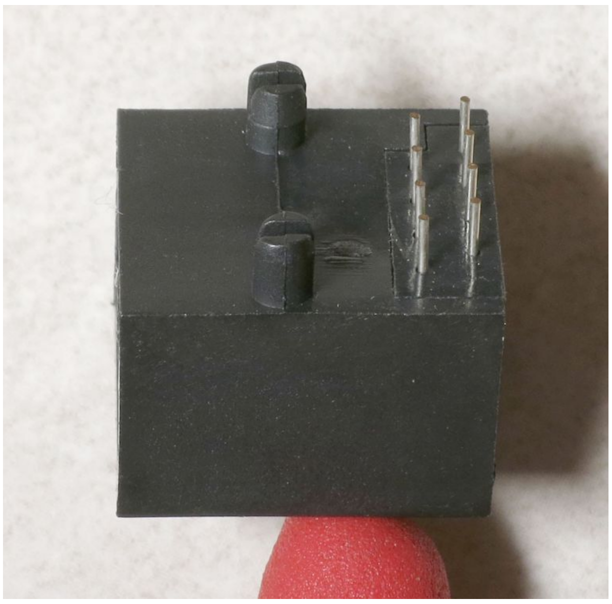
And an oblique view (the red thingie at bottom is a plier handle used to prop up the jack):

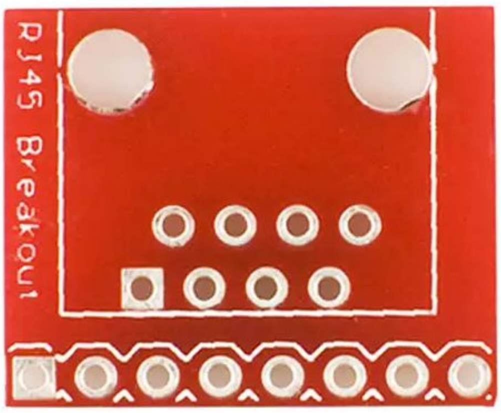
You can actually solder the leads on the right to a standard PCB board if it has a set of holes that have matching spaces, but in this case we have evenly-spaced holes 0.1" apart, so that won't work. Breakout board to the rescue. I didn't have any unsoldered RJ-45 breakout boards available, so I borrowed this pic from Digikey:

You will need to break off an 8-pin segment of one of your 40-pin 2.54mm single row pin header strips, and solder it to the underside of the breakout board, so that the short pins poke up from the lower pins of the board as seen

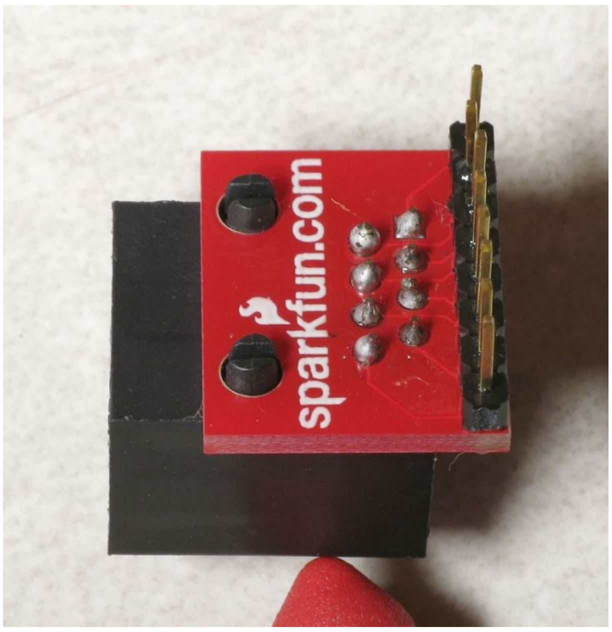

PROTOADVANTAGE.com
TO-263
D2PAK-5







above. Solder one pin first, then straighten the set of pins if necessary by remelting the solder on that one pin and re-positioning the 8-pin header strip so that it's perpendicular to the board. Then place on the RJ-45 jack on the top side of the board so that the 8 staggered pins go through the 8 staggered holes, pressing down so that the two large pins on the RJ-45 jack go through the two large holes as seen above. Solder the 8 staggered pins to the holds, and you're done. It should look like this:



Time to start putting together the low-side driver board with the CAT4101s. First step is to install the connector on the 11 x 8.5 cm protoboard. This is the spec'ed component, but if you can't find one that size, you can use another one up to 11.5 cm in the long (horizontal) dimension, and 11 cm in the short (vertical dimension), which will fit in the enclosure I've been using. Bit smaller in horizontal dimension will work, but if it's too small you may have problems getting all the components and wiring to fit; probably won't work if it's a bit smaller in the vertical dimension. If you use a larger enclosure, then you can use a larger protoboard; don't go smaller with the enclosure, or the controller boards may not fit.

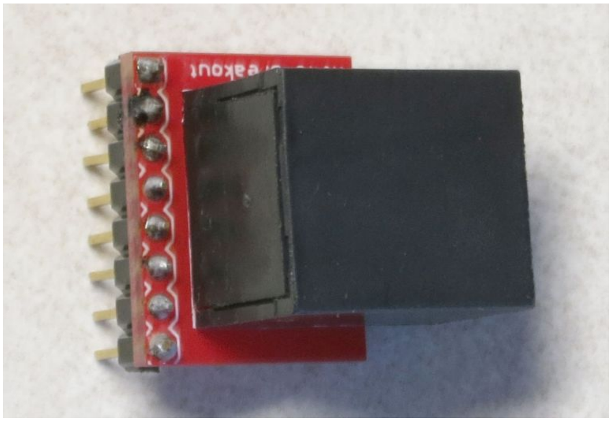
The connector is a right-angle male header, which comes in a 40-pin length:

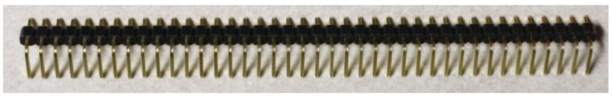
We need a 20-pin connector, but it's easy to break the 40-pin in half:

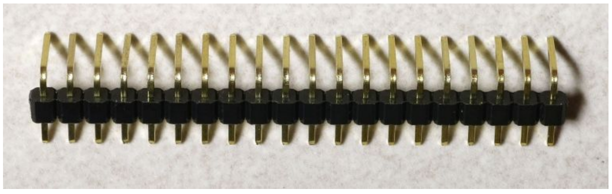
You're going to be soldering the connector to the protoboard as far to the upper left as you can get:

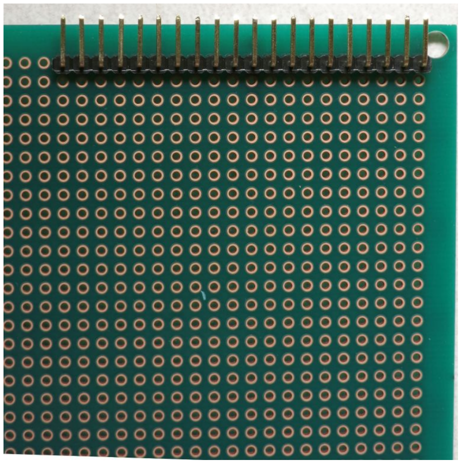
The short pins go into the protoboard for soldering. The plastic base of the pins needs to be flush with the protoboard, so that the long pins are parallel to the protoboard. It can be tricky to do this. I approach this by

sparkfun.com

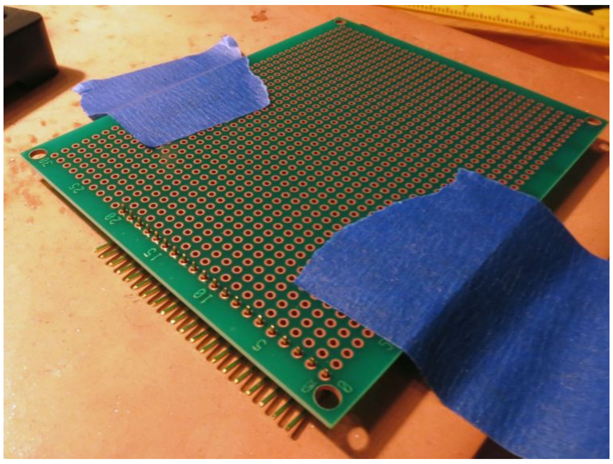




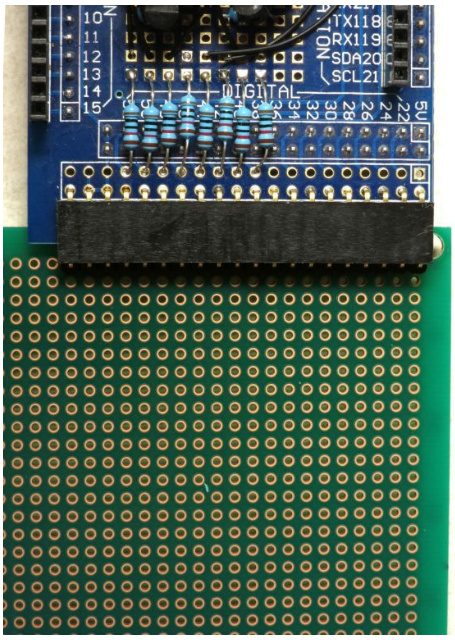




inserting the connector into the board, flipping it upside down, placing it on the soldering work area, and taping it down:



Solder a pin at one end, then remove the board and see if the connector is flush with the board. If not, reheat the solder connection and twist the connector to make it flush with the board. Caution: Make sure you don't do what I've done a couple of times, and press on the same pin that's touching the soldering iron ;-). Check again to see if it's flush. If yes, solder the pin at the opposite end, and check to make sure that it's flush as well. When both ends are flush, solder one more pin somewhere near the middle of the connector, and make sure that's flush, too. If you've done it right, the male connector on this board should plug cleanly into the female connector on the MOSFET shield you assembled in the previous steps (no bending required):

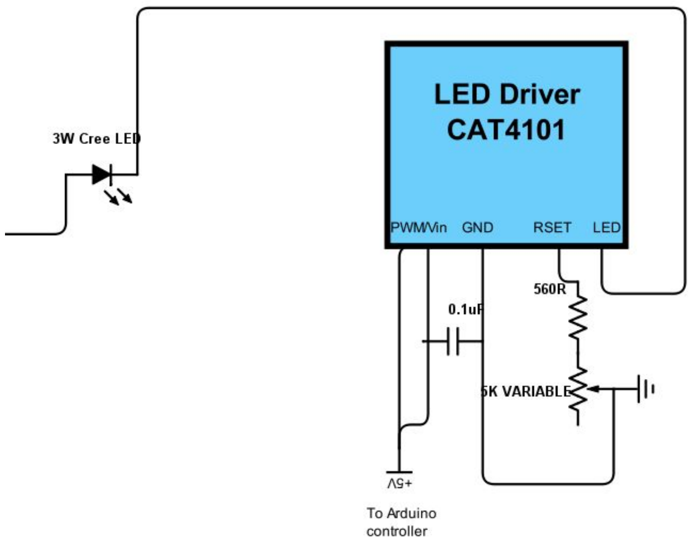


**Note:** In schematic wiring drawings, the wire lines are drawn from one hole on the shield or protoboard to another. The wires connecting these holes should have the insulation stripped from their ends, and have the bare lead inserted into the hole. The bare lead should then be connected to the specified nearest component/connector lead.

Time to install the CAT4101 LED driver breakouts and associated components onto the protoboard. Here's a crude circuit schematic:

There are 5 leads on the CAT4101. From left to right:

1. EN/PWM: Enables the current from the LED driver. PWM is pulse width modulation, a way to control



apparent light intensity by varying the input signal voltage - not used in this system.

2. Vin: Powers the chip with +5V.
3. GND: connects to ground.
4. RSET: Connects to a resistor that sets the output current per this curve (from the datasheet).
5. LED: Connects to the ground lead of the 3W LED you're lighting up.

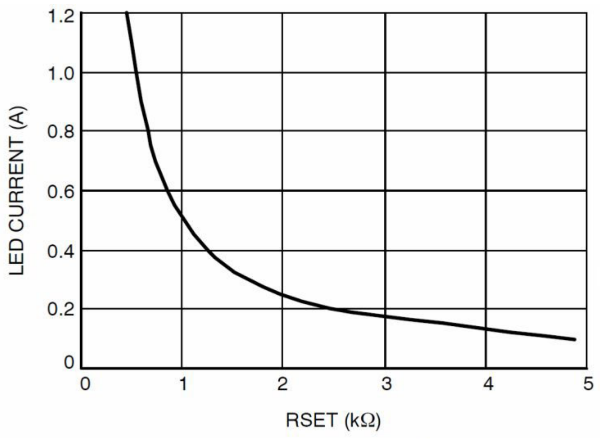


Fig. 1: LED current vs. RSET Resistor

In my circuit, EN/PWM and Vin are shorted together, and both are connected to an Arduino output that supplies +5V. I tried a constant +5V connection to all the CAT4101 driver Vin pin 2s, and a separate Arduino connector to all the EN/PWM pin 1s, and wound up with weird issues like multiple LEDs lighting up when only one should have. Shorting 1 and 2 together and hooking them up to the Arduino pin output fixed that problem, as both the power supply and enabling voltage are turned on and off at the same time.

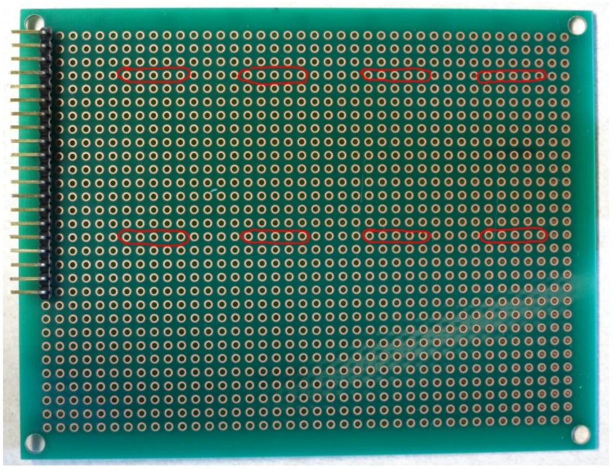
The 0.1 uF capacitor between the input voltage and ground is recommended by the manufacturer to reduce input noise. I'm guessing it's probably not necessary in this use case, but it's cheap to play it safe. Plus, it makes some upcoming wiring connections a bit easier.

The RSET graph above shows the current going as high as 1.2 A, but the recommended max current for both the CAT4101 and the 3W LEDs is 1 A, which the datasheet says corresponds to a resistance of 549R. The 560R resistor in the current is the closest higher standard resistor value, and is there to make sure you don't accidentally send too much current through the CAT4101 and LED. The 5K variable resistor is there to let you adjust the current to a specified value.

Three reasons for this:

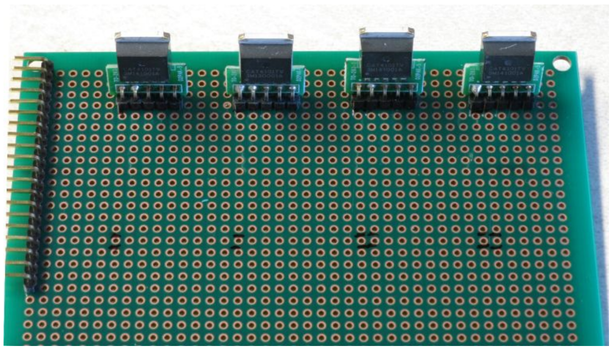
1. The current v. RSET curve is steep at lower resistances, so even small circuit variances between different CAT4101 circuits can result in significant differences in current. The variable resistor lets you set a specific current for every CAT4101 circuit, which insures that all LEDs will have the same light intensity.
2. If you're in a remote location and running from battery power, turning down the intensity will prolong battery life. Your exposure times may need to be a fraction longer, but that shouldn't add too much extra time to the cycle.
3. When using USB microscopes, the light intensity at full 1A current is so high that it can saturate the image sensor. Turning down the current reduces the intensity to a usable level.

Here's the protoboard with connector (from the previous step), with the locations for the CAT4101s circled in red:



This is with the recommended protoboard. If you use a different size, you'll have to reposition these slightly. But read all the following instructions for your protoboard, to make sure the new positions don't interfere with upcoming components or wiring.

I usually start by installing 4 CAT4101s on the top row:

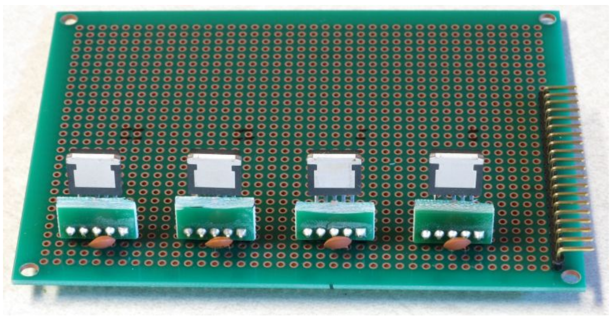


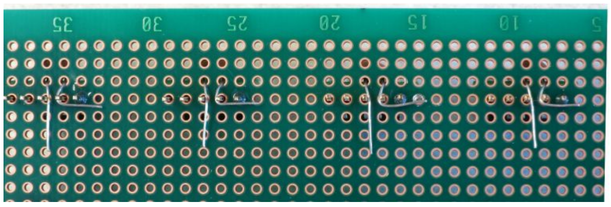
By not installing the second row right away, I get a little extra room in front for access. The black marks are where I marked the pin 1 position for all the CAT4101s. Notice the extra marks on the lower right - that's where I got it wrong the first time. Measure twice, solder once ;-). I start by soldering the pin 1 lead in place, then flipping the board over and seeing if the CAT4101 breakout is flush with the board and perpendicular. Spoiler: it almost never is. Touch the soldering iron to the pin 1 connection to re-melt the solder, then re-orient the breakout board manually until it's flush and vertical (as seen above).

Next, install the capacitors. Push the two capacitor leads into the holes next to CAT4101 pins 2 and 3, on the back side of the board, like this:

Push them in as far as you can - the more capacitor lead you get on the bottom, the easier subsequent steps will be. Flip the board over, and position the capacitor leads like this:

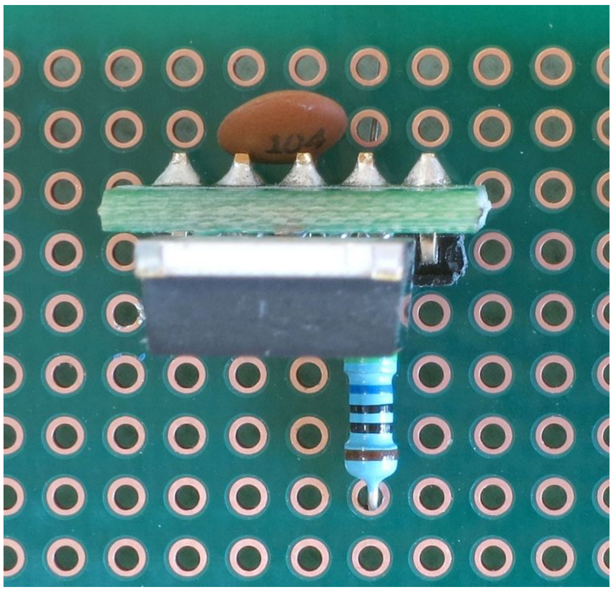
The lead next to the center pin 3 (GND) should be laid flat and straight touching pin 3 and extending beyond. The lead next to pin 2 (Vin), should wrap around pin 2, and then be laid flat against pin 1 (EN/PWM). Solder the leads





to the pins they're next two; make sure both pin 1 and pin 2 are soldered to the lead next to them. **Don't trim the excess leads - you'll need them in a bit**.

Next, install a 560R resistor on the front side of the chip, next to pin 4 (RSET):



Orientation doesn't matter with resistors, but my OCD usually compels me to make sure they're all oriented the same way. Flip the board over, and flatten the resistor leads like this, but don't solder them yet:

This pic also shows how the capacitor leads should have been soldered (missed taking that pic).

Now take a 5K trim pot:

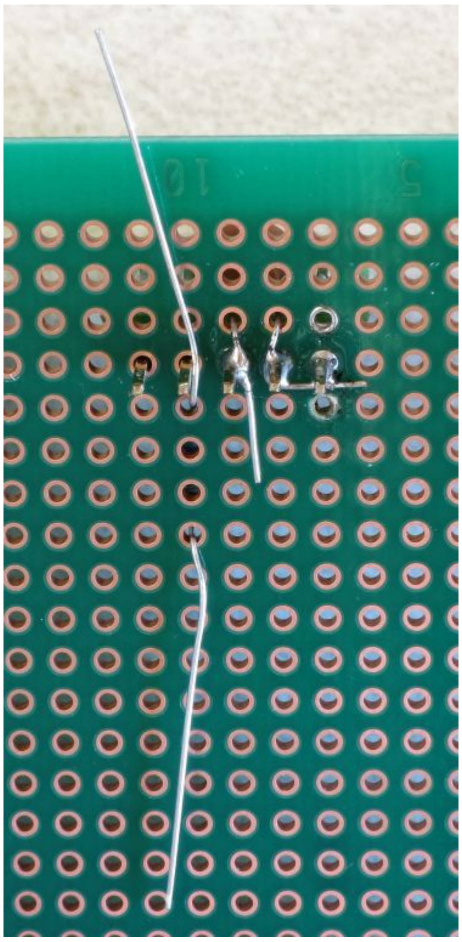
Cut off the lead furthest from the screw on top, converting it into a variable resistor:

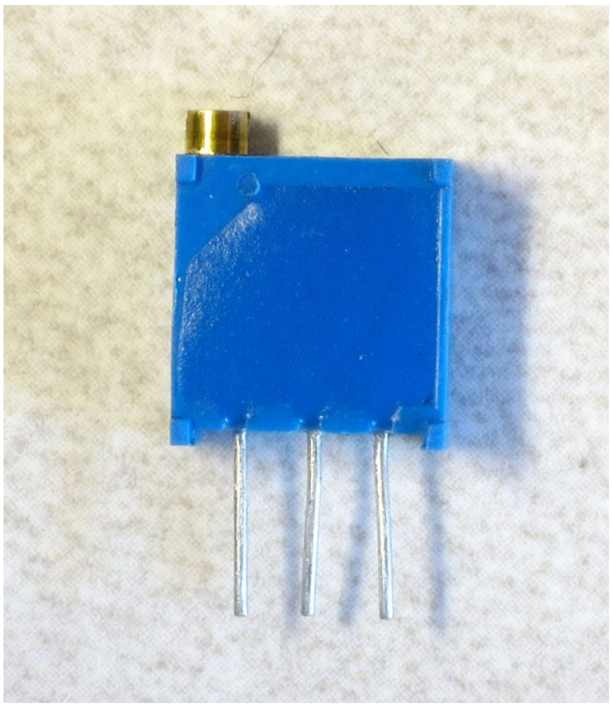
Insert it next to the GND pin (pin 3) on the top of the protoboard:

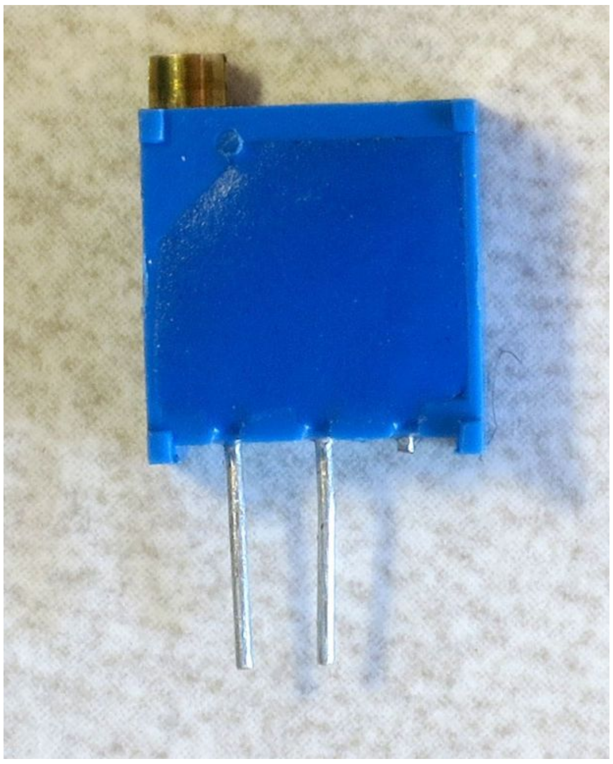
It's a snug fit, but not soldering the 560R resistor means you should be able to nudge it to the side and make everything fit.

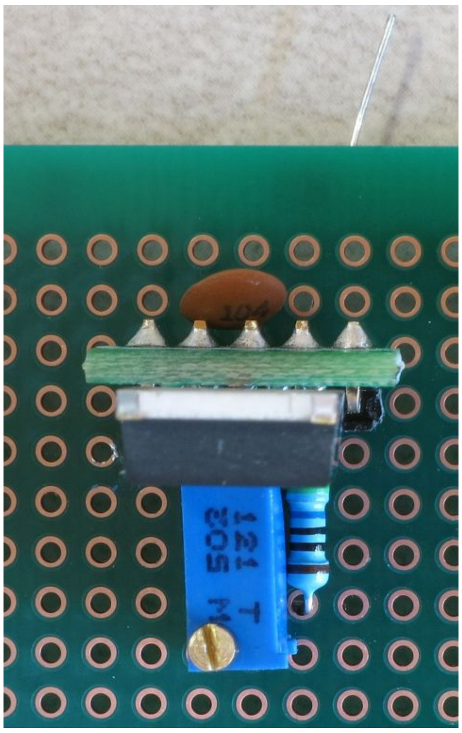
Now flip the board over, and arrange the variable resistor leads this way:

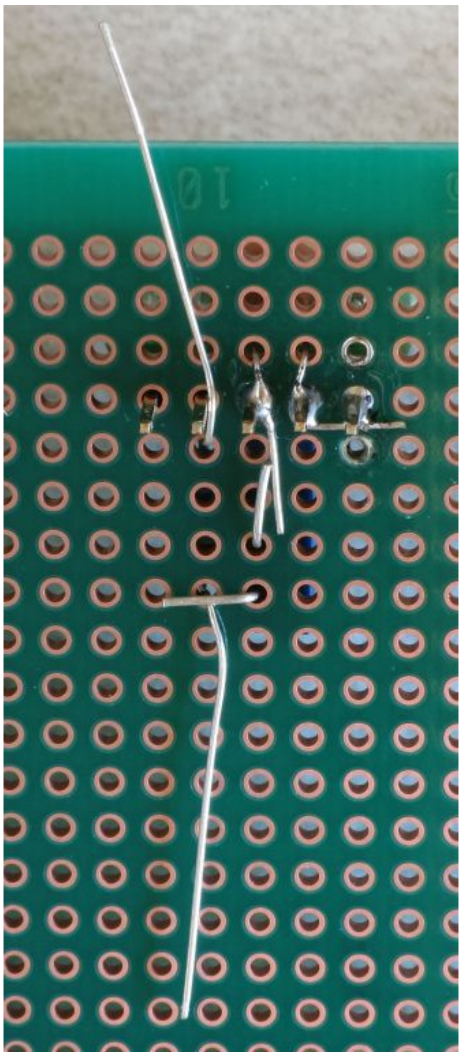
The variable resistor lead furthest from the CAT4101 chip should be bent over to touch one of the 560R resistor leads. The variable resistor lead closest to the chip should be bent towards CAT4101 GND pin 3, overlapping and touching the capacitor lead that's already soldered to pin 3. Now solder the 560R resistor lead at top to pin 4


10
5

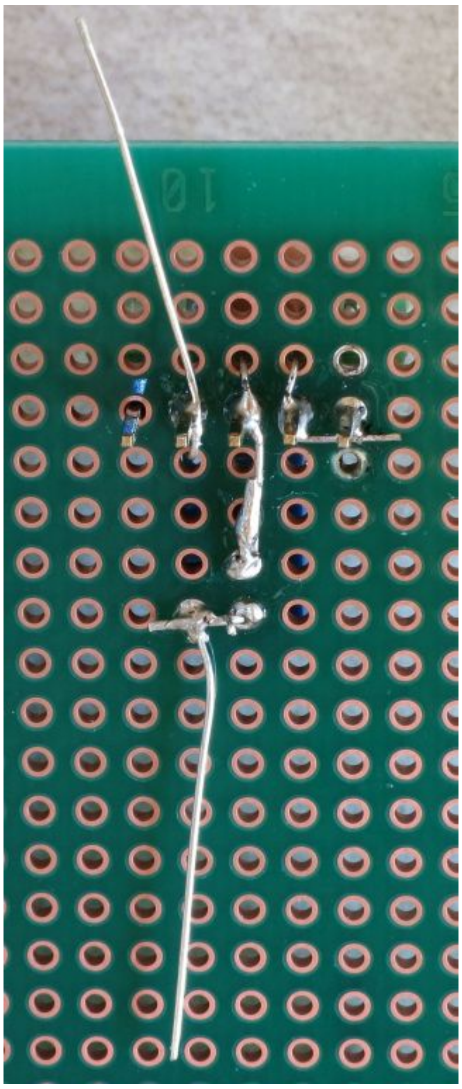





104
121 T
205 M



(RSET) on the CAT4101, solder the bottom 560R lead to the variable resistor, and solder the other variable resistor lead to the capacitor lead. Should look something like this:



Trim off the excess resistor leads, and you're done with this CAT4101 circuit. Repeat for the other 3 CAT4101s in the first row:

Which should look like this on the bottom, after all the soldering is done:

Now repeat the process with the second row of CAT4101s. Install the CAT4101 breakouts:

Then first install the capacitors, then the 560R and 5K variable resistors:

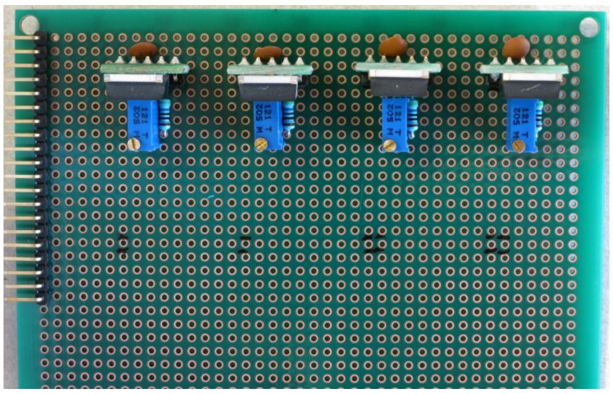
Which should look like this on the bottom:

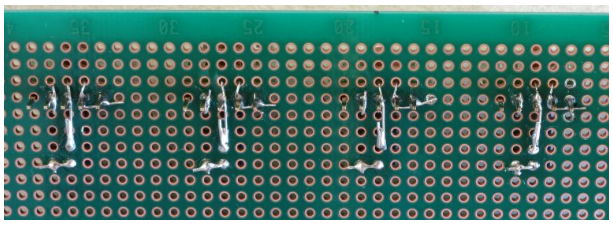
Isn't that nice, neat and clean? Don't get used to it - you're about to mess it up with a bunch of wires. On the plus side, you have a lot more room to work with on this board than you had on the MOSFET driver shield, so the amount of cursing you'll do should be a lot less.

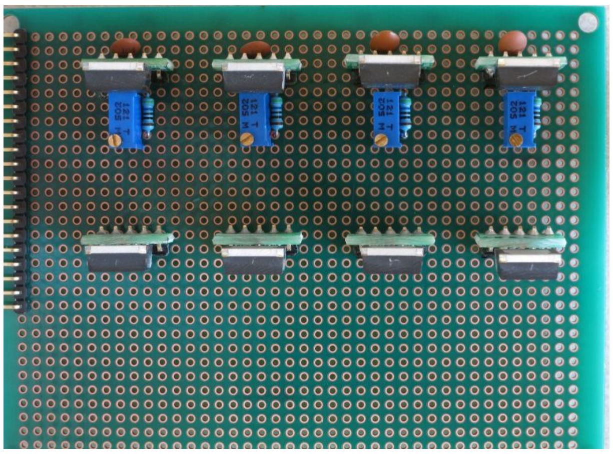
But before adding the wires, there are a few additional items that need to be soldered on the protoboard. First, the RJ-45 Ethernet jack on its breakout board:

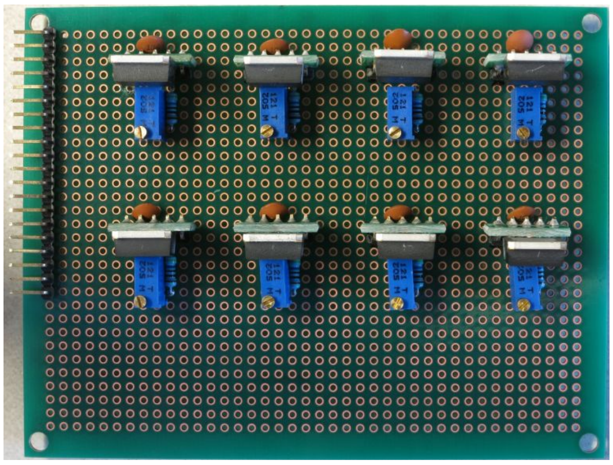
If you're using the protoboard recommended in the parts list, this is the correct position for it, all the way down in the lower-left-corner. Sliding the whole protoboard into the MOSFET shield board, it will just barely clear the edge of the board (and if it doesn't, a bit of filing should fix that):

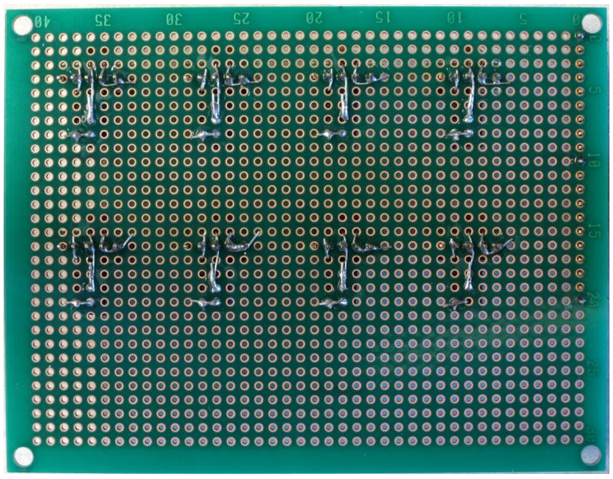
If you use a protoboard of a different size, and in particular one that's smaller in the vertical dimension, you will probably have to re-orient the RJ-45 so that it faces downwards in order to make it fit. Not a big deal.

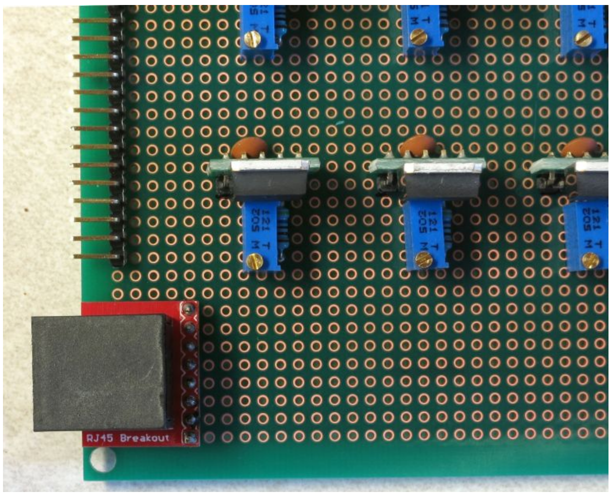


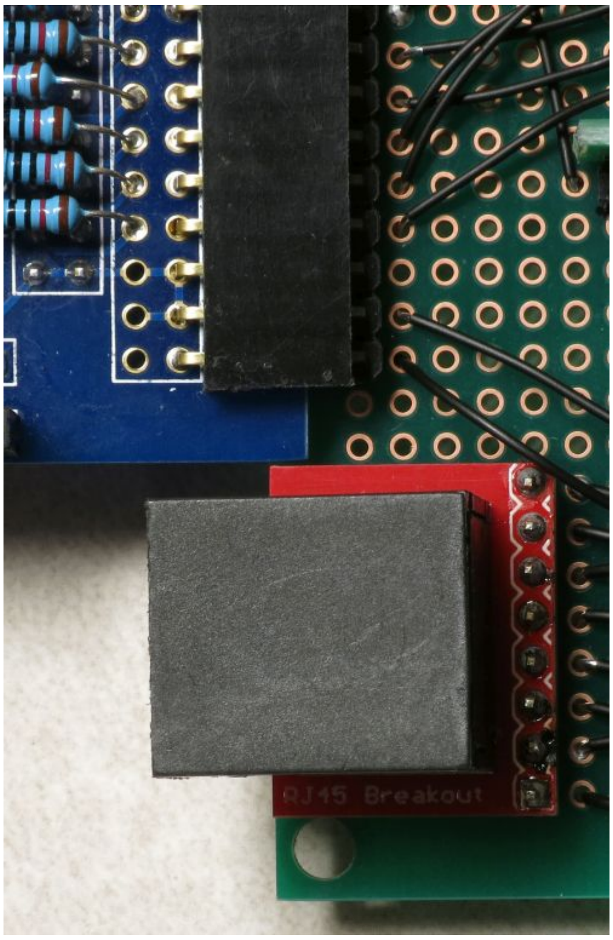









RJ45 Breakout



Unlike previous connections, where you want the connector to be flush with the board, for this connection you want the RJ-45 jack to tilt until the black knobs on the bottom touch the board:

Solder it into place using the first and last pins.

Next, put the buzzer in place on the board as shown below (+ on the right), and solder the two pins to hold it in place:

Now break off a strip of 5 male header pins:

And solder it next to the buzzer (1st and last pins):

It's a tight fit, but the 5-pin female header on the USB panel connector should still fit (or break out your file again):

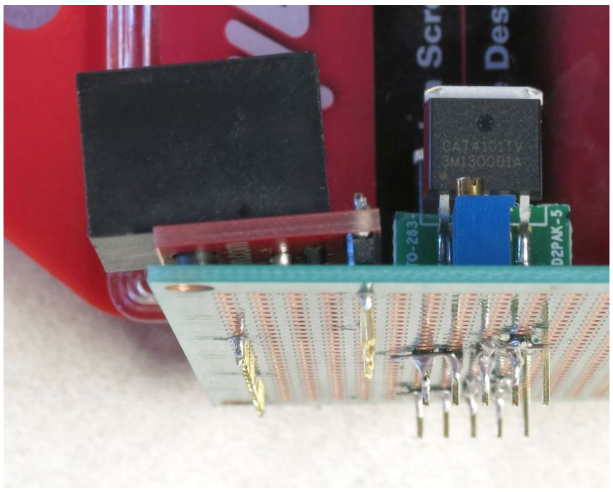
Finally, you'll be soldering a 2-pin male header next to the 5-pin header you just did, and a two-pin female header on the middle right of the board, as shown here:

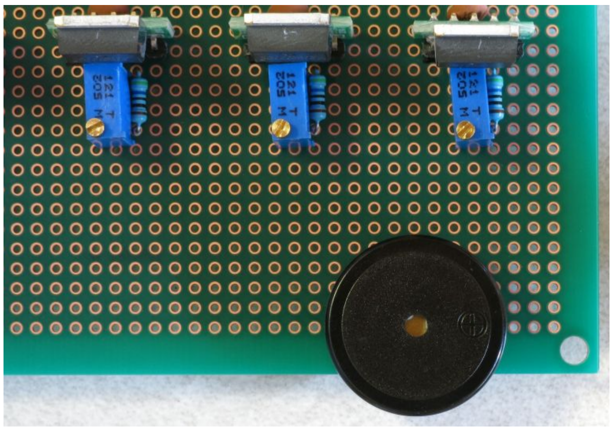
For the male header, use a blob of solder to connect the pins on the 2-pin and 5-pin header that are next to each other. This 2-pin header is for a jumper that will optionally power a servo to fire the camera shutter for those cameras that don't support remotes. Make sure the solder only touches the adjacent pins, and no other:

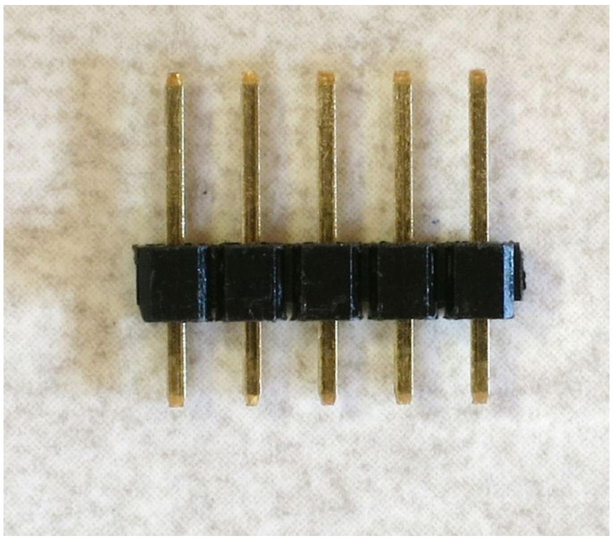
Next step is to wire up leads from the CAT4101s to the appropriate connectors. First, the LED connections that will ultimately connect to the ground (negative) lead on the LEDs, running along rows of the LED matrix (the rightmost CAT4101 connection in the schematic below, aka pin 5, LED in the figure below). Then, the connections from the Arduino to the CAT4101, that supply power and enable the CAT4101 (the two leftmost connections, pins 1 and 2, PWM and Vin in the figure below).

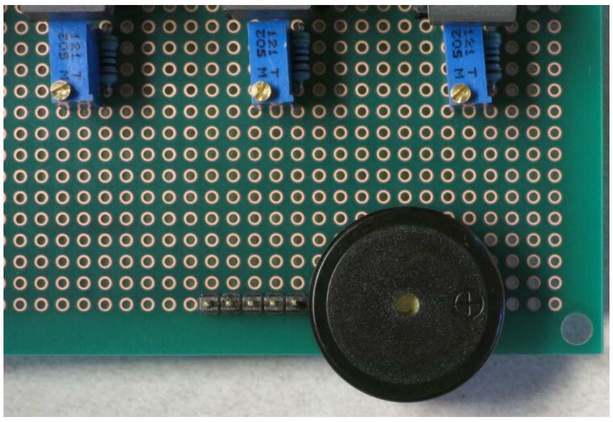
Each CAT4101 controls a separate row of the LED matrix, identified in the picture below with numbers 1-8 for the 8 maximum possible rows.

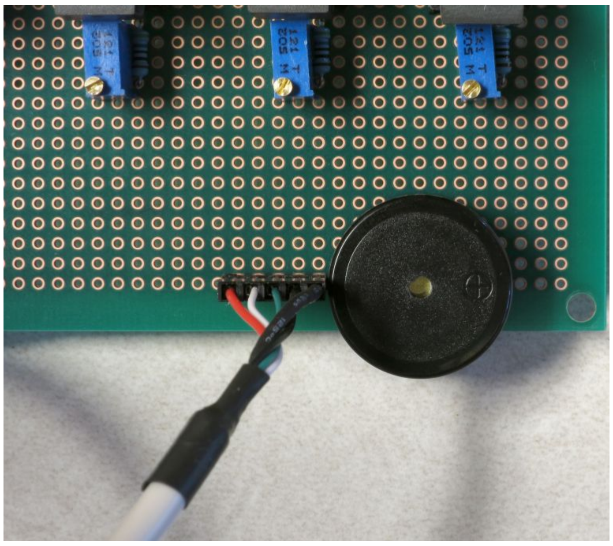
The connector on the left has the numbers 1 and 8 next to the positions where pins 1 and 2 on the 1 and 8 CAT4101s are wired to the corresponding Arduino outputs that power and enable the CAT4101s; the remaining connection

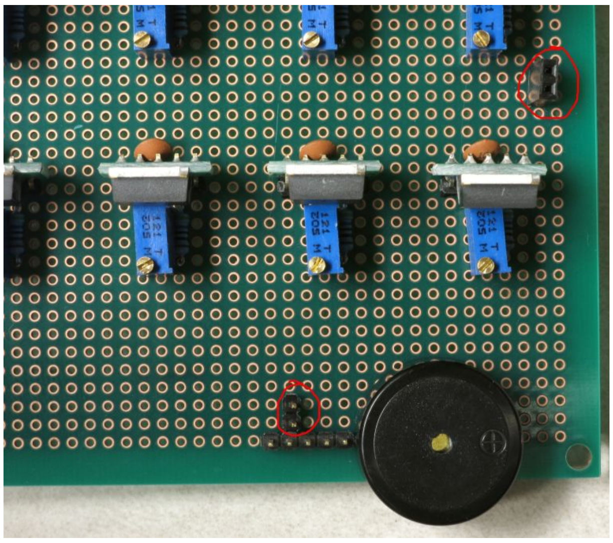

CAT4101TV
3M130001A
TO-263
D2PAK-5

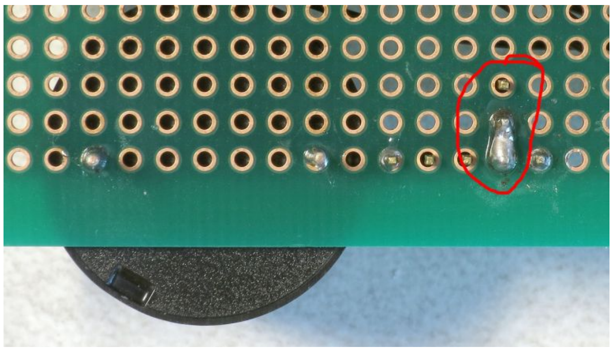

502
121
T
M

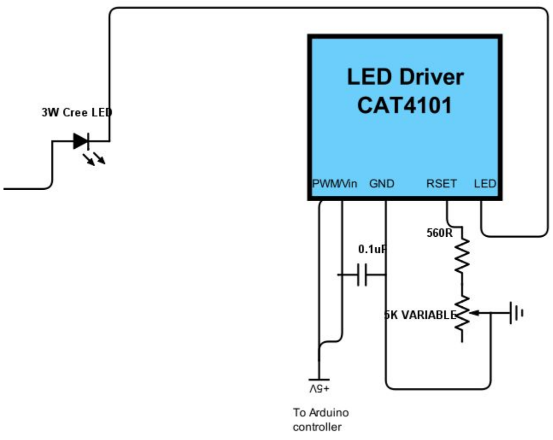


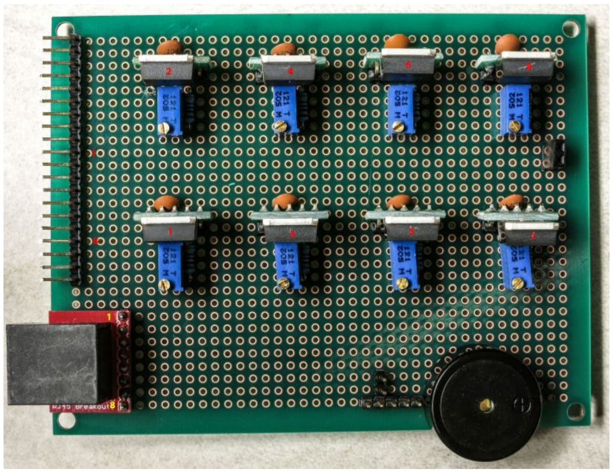






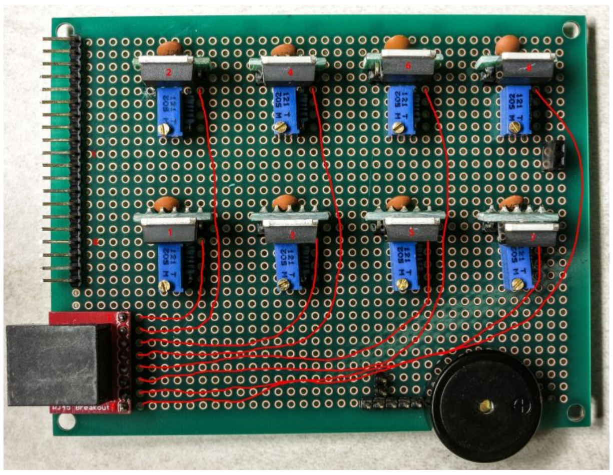



LED Driver
CAT4101
3W Cree LED
PWM/Vin
GND
RSET
LED
560R
0.1uF
5K VARIABLE
+5V
To Arduino controller

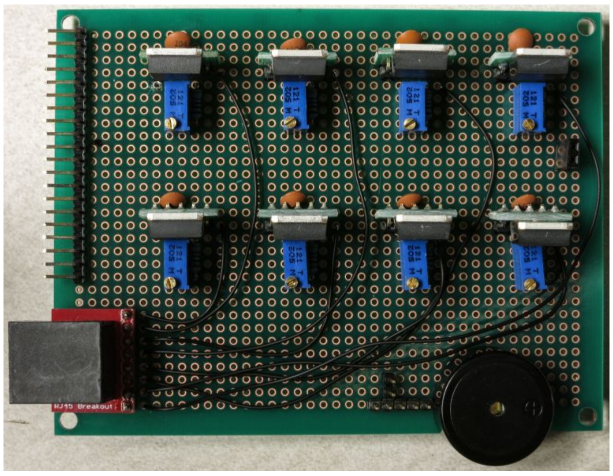


numbers 2-7 are sequential between the 1 and the 8, of course.

Similarly, the numbers 1 and 8 on the RJ-45 breakout board (lower left) are where you'll use the Kynar 24 AWG wire to link the pin 5 connection on the CAT4101 to the breakout board for the corresponding number, with the other pins corresponding to connections 2-7. I usually do the LED connection first - here's a schematic showing the wiring to be done between pin 5 and the RJ-45 breakout (in red):



You don't have to follow the wiring path exactly - it only matters that the matching connections are soldered correctly. Here's how my board looks after that step:

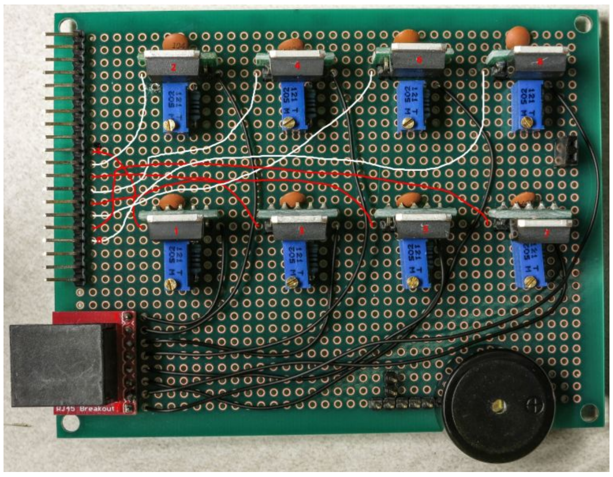


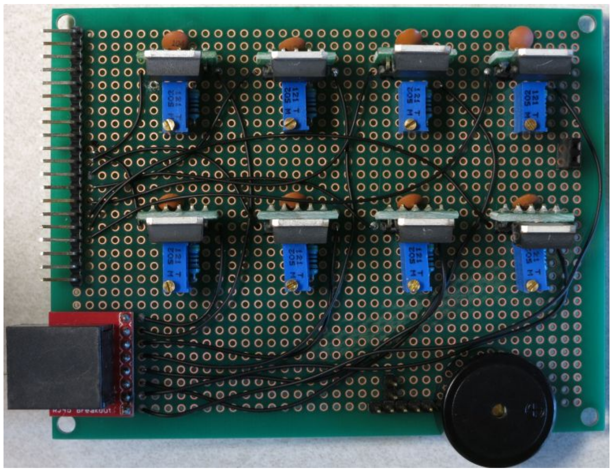
Next step is to electrically connect pins 1 and 2 on the CAT4101s to the corresponding pin on the protoboard connector on the left, which will receive control/power voltages from the Arduino. Since these were already shorted together when you installed the capacitors, you only need to solder the wire to pin 1. You can use the Kynar AWG 24 wire, but if you have thinner wire, you can use that as well, since this connection carries very little current (<20 mA). Here's the wiring schematic; I used a mix of red and white in drawing the wires to make it clearer, the color has no other significance:

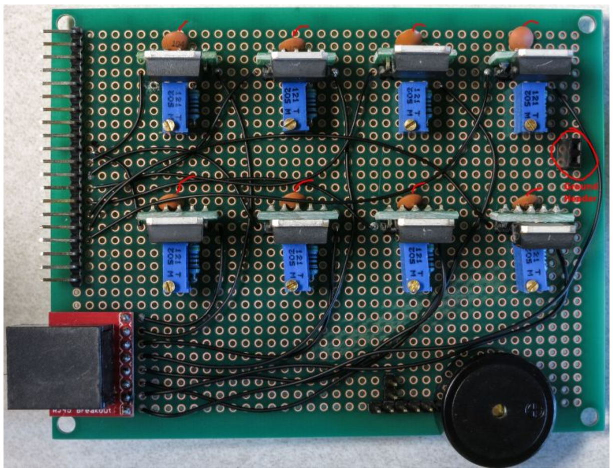
Once again, you don't have to copy the exact wire layout, you just have to make sure CAT4101 connection goes to the correctly numbered pin on the connector. Here's how mine looked after this step:

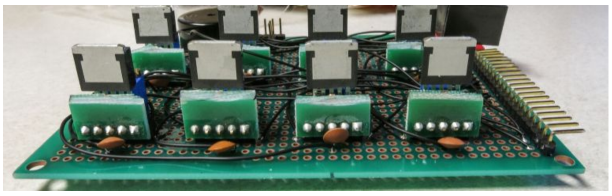
Time to finish wiring up the CAT4101 board. First step is to hook up the ground pins on all 8 CAT4101s to a common ground pin header on the right. Not going to draw all the wires here, as it's already too complicated. Plus, the exact layout of the wires doesn't matter - as long as the center pin of every CAT4101 gets connected to ground, the way the wire is laid out on the board doesn't matter. Here's a schematic showing where the ground wires should come off of the CAT4101s center pin, and where the ground two-pin female header on the right the wires should hook up to:

It's a bit difficult to see the exact connection spot, since the capacitors obscure it. So here's a side shot from the back, with the ground wires already soldered in place to the center CAT4101 pin (which is also already soldered to one of the two capacitor leads):

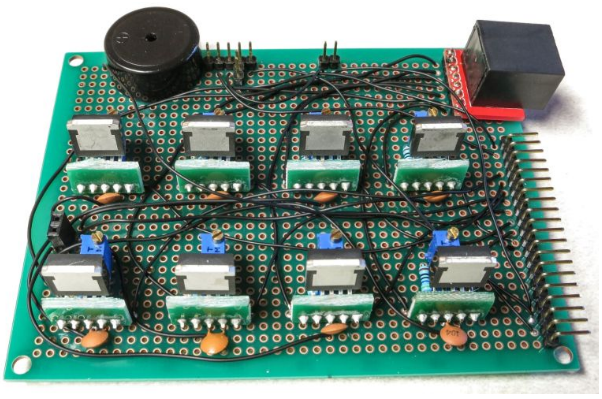

RJ45 Breakout


RJ45 Breakout

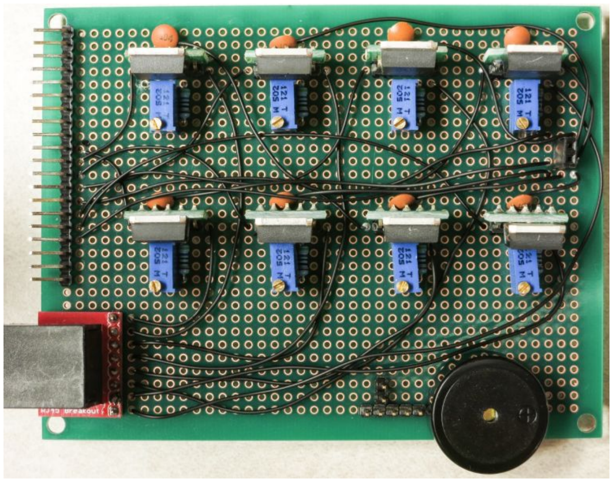

Ground
Header
RJ45 Breakout



And here's an oblique view that shows the ground wire connections to all 8 CAT4101s:



Here's a view from the top, with all the ground wire connections in place:



See how all the ground wires go to the two-pin female header on the right? Flip the board over, and the bare wire leads should all be bent over to overlay the header pins. Then use a lot of solder to make sure all the wire leads and pin headers are electrically connected; that's what the solder blob on the right below is:

Notice also how the wire lead from the ground wire connection is soldered to the center pin of the visible CAT4101s, and also to one of the capacitor leads.

You may be tempted to make this connection with shorter jumper wires between adjacent CAT4101 chips. Don't - it's specifically not recommended in one of the CAT4101 datasheets; they say that every ground connection should go independently to a single "star" ground that each CAT4101 connects individually to. I actually tried the jumper approach once, and ran into weird problems with LEDs turning on when they should be off.

Next step is to wire up the USB control pin and beeper pin from the connector at upper left to the appropriate positions. The pic below shows where the connections should be made:

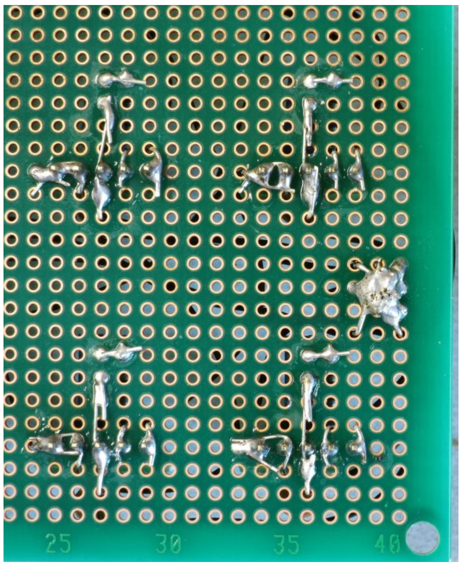
Here's how it looks on top after the wires have been connected:

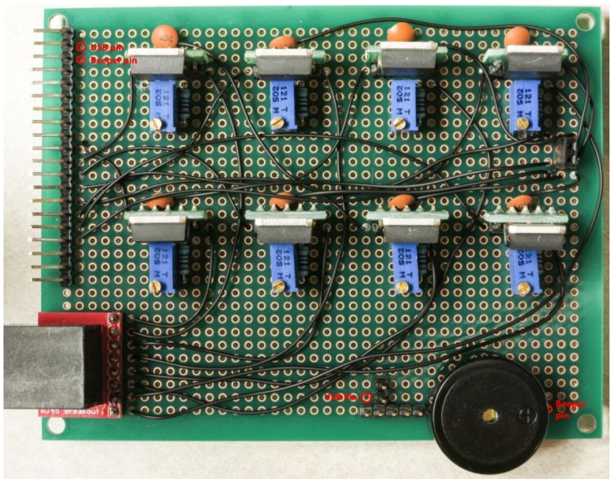
The 5V power supply for the servo shutter, and the ground connection for both the USB and beeper, need to be hooked up:

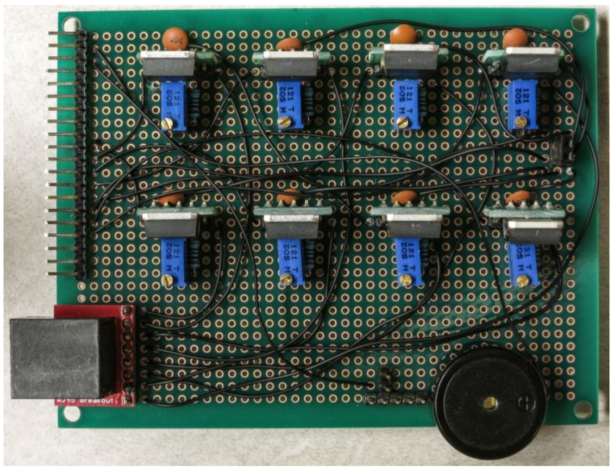
The 5V connection is straightforward, it just gets soldered to the top pin of the two-pin female header:

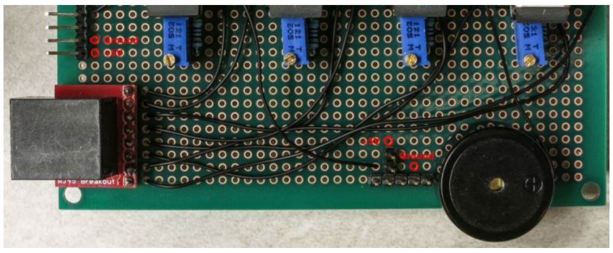
The ground connection is a bit more complicated, since it needs to be connected to multiple pins. Trim the insulation a bit longer on the right end of the lead, and when you insert it in the specified position, bend it around that fourth pin on the 5-pin male header so that it touches the 5th pin on the header and heads over towards the beeper ground pin. Then bend the beeper ground pin over to make contact with the wire lead. It should look something like this:

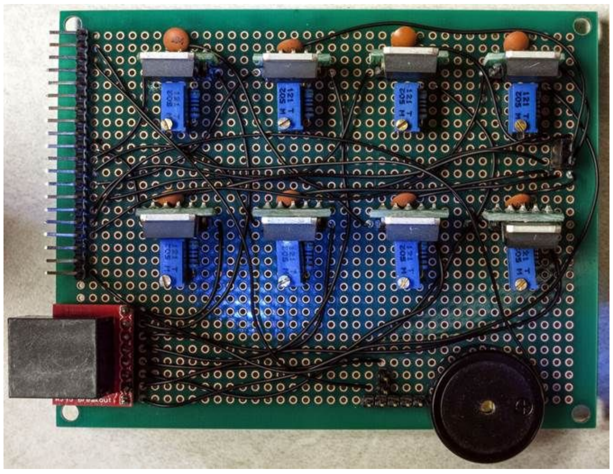
Now lay the solder on thick so that all those pins and the wire are electrically connected:

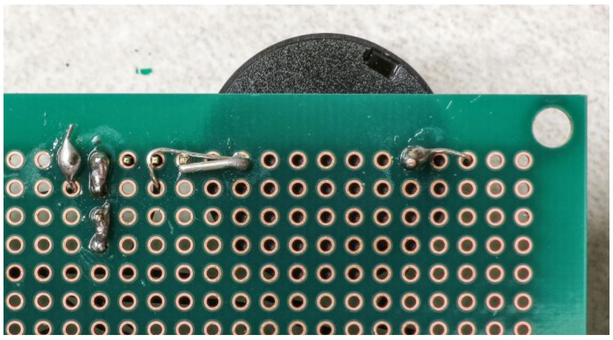


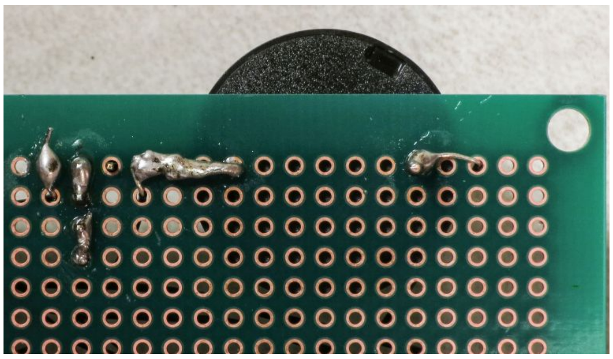




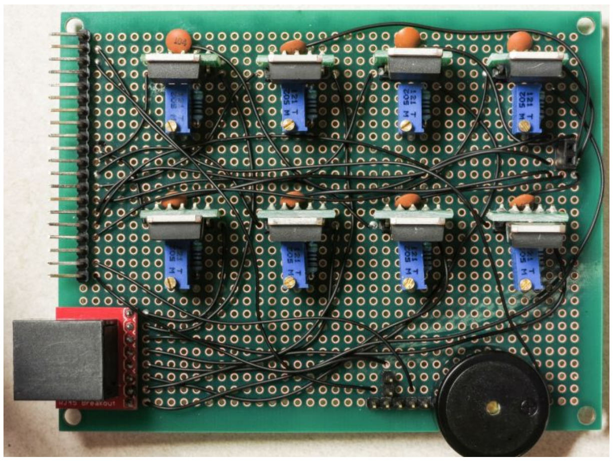




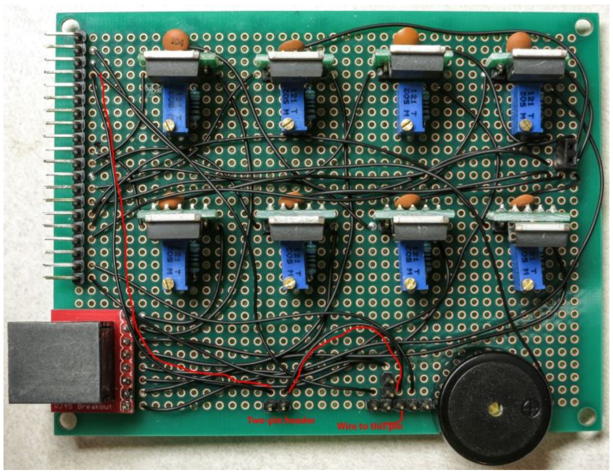




Check that all the pins/wires are soldered as you see above. On the top side, the board should now look like this:



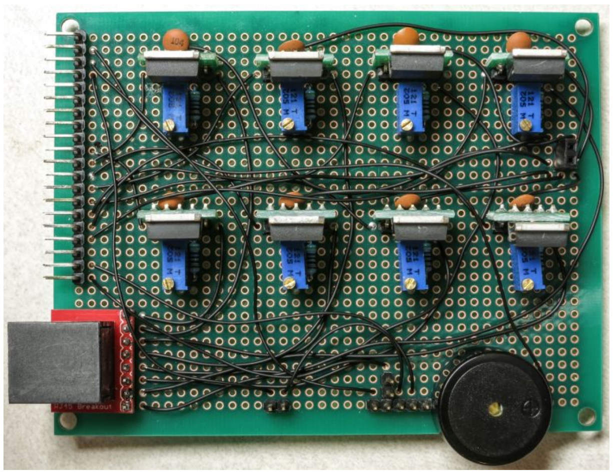
Originally, the board was done at this point. However, since I decided to add a servo shutter system to operate cameras that don't have a remote capability, I decided to add an additional electrical connection to the 5-pin male header that the USB panel connector will be hooked up to. And I wanted the option to be able to disconnect this if I didn't need it, to remove the possibility of problems or damage to cameras with misapplied voltages. So I added a two-pin male header to the bottom, soldered a wire connection from the connector at upper left to the left pin, and a wire connection from the right pin to the one remaining unsoldered pin on the 5-pin male header. Here's what it looks like; the drawn red traces parallel the actual wire connections:



And here's what it looks like without those red traces, and all the wiring done:

Double-check all the wires you see here to make sure your board has a corresponding connection. If they all check out, you're done with this board, and with the most difficult wiring of the entire project. Yay!

One more step, which I don't have a picture of. The Ethernet jack in the lower left-hand corner is held on only by the 8 soldered pins. That's reasonably strong, but it could use some reinforcement. Take some adhesive, and fill in the gap between the red Ethernet PCB board and the green PCB board. Thick stuff you can stick into the crack is best; take care not to block the Ethernet jack. I used Plastiweld, but hot glue or silicone adhesive would be reasonable alternatives.



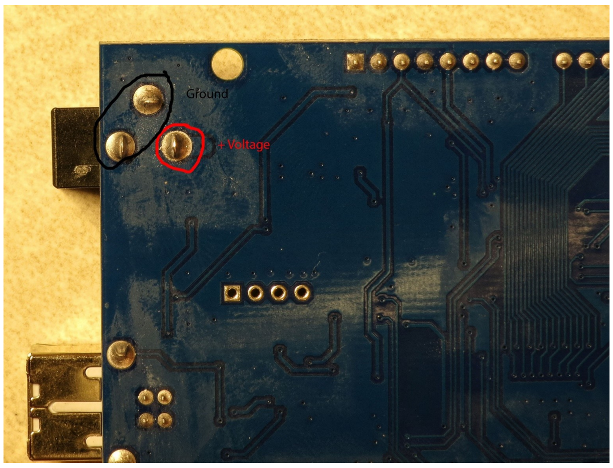
CHAPTER 7

# Power supply connections

This section deals with power supply connections for both the LEDs and the Arduino.

The Arduino Mega is designed to supply up to 500 mA of current from the +5V connection when running from a USB connection. That's not enough to power the 1A LEDs in this RTI system, so a separate power supply (8-12V) capable of supplying up to 2A of current is specified (2A because I've had problems with noise from power supplies that are only spec'ed for 1A). While there's a Vin header on the Arduino board that supplies full voltage from the power supply, I believe it shares the 500 mA current limit as well. So I decided to power the LEDs directly from the 8-12V power supply, bypassing the Arduino completely.

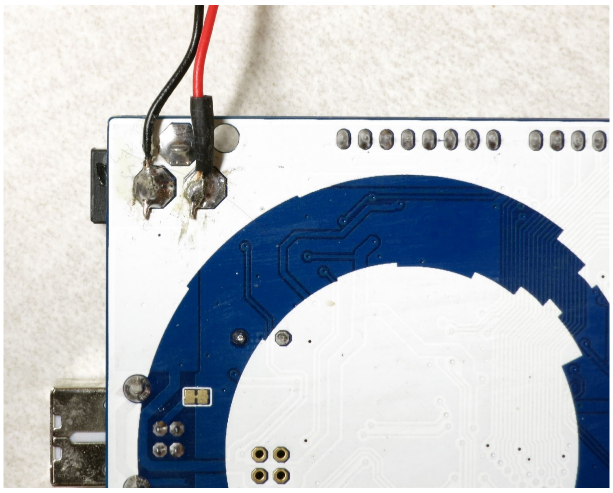
On the bottom of the Arduino Mega, near the power supply jack, are three solder connections to the power jack. Two connect to the power supply ground, while the third connects to the power supply positive voltage (8-12V):



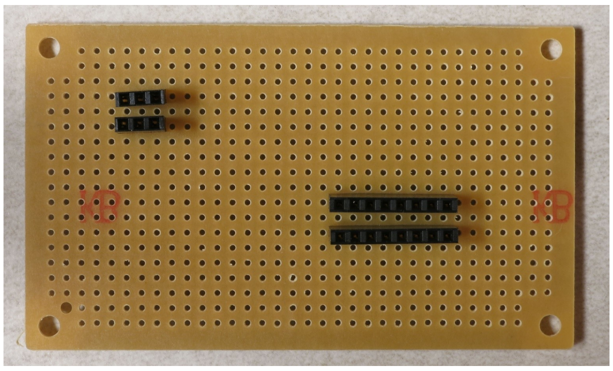
You will need to solder 22 AWG wires to both the positive voltage connection and one of the two ground connections; cut two pieces of wire about 4" (10 cm) in length, and strip the insulation of both ends of each piece of wire. The usual convention is that positive voltage is indicated by a red wire, while ground is a black wire.

There are large blobs of solder on the connections, so it may take a while to get the solder to melt. Use a large soldering iron or gun if you have one, otherwise either use a chisel tip on your soldering iron for maximum heat conduction or wait a long time using a regular point tip for the solder to melt. Then press the bare wire lead into the melted solder blob and let it cool; if necessary, add more solder to cover. It should look something like this:

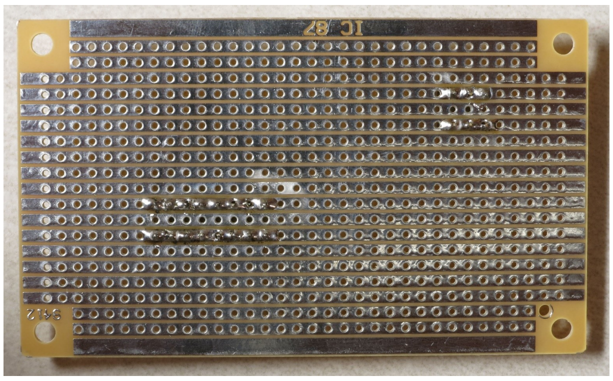
I used a small piece of heat shrink tubing on the red wire to make sure it wouldn't short out against the ground connections.



There are going to be a fair number of connections required to both the +8-12V/GND connections directly from the power supply, and also the +5V/GND connections from the Arduino board; for the latter, more connections than are actually available on the Arduino. To solve this pin connection shortage, you will need to solder two 3-pin female headers to a strip protoboard for the high voltage supply and ground, and two 5-pin female headers for the Arduino +5V and GND connections:

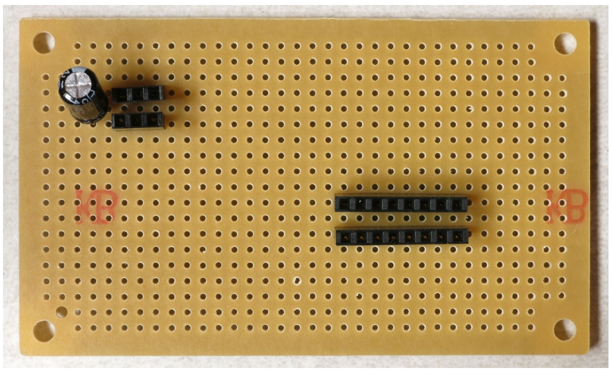


If you flip this strip protoboard over, you'll see that instead of every hole being independent of each other as on a plain protoboard, holes in a single row are electrically connected to each other with a metal strip. So if you solder every header pin to that metal strip, as in this picture, all the sockets in each individual female pin header will be electrically connected to each other:



Finally, although the +8-12V power supply should produce a fairly clean, noise-free voltage, I added a 100 uF electrolytic capacitor across the 3-pin headers to play it safe and smooth out any residual noise. The capacitor is soldered to the strips so that the side with the white indicator goes on top, as seen in this picture:

That white capacitor indicator goes to the ground side, and that's the header side where you will plug in the black ground wire from the Arduino; the + voltage red wire will plug into the other header. That will leave the other



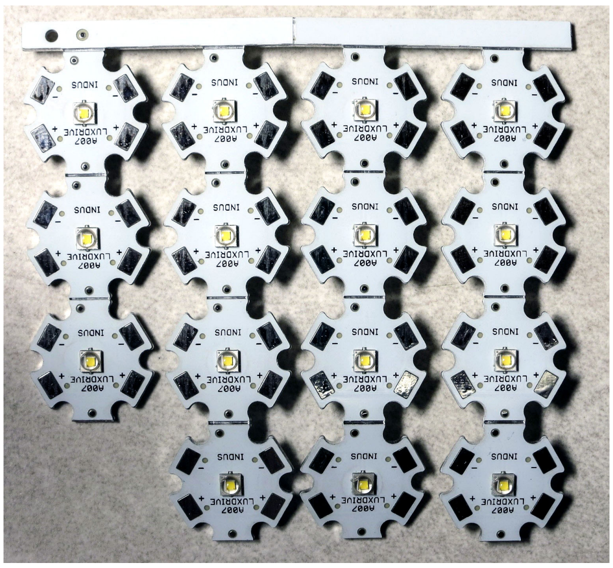
two female header sockets on this strip free to supply power and ground to the LEDs, and also to power an LED that indicates when the power supply is plugged in. No capacitor is need for the 5-pin headers, since it receives voltage that's already been filtered by the Arduino.

You now have the ability to power up the system to turn on the high-power LEDs. But to test both the system and the LEDs, connection wires will need to be soldered to the LEDs so that they can be hooked up to the system. That's the next step.

CHAPTER 8

# Preparing the LEDs for installation

When you receive your 3W LEDs in standard star packages, they'll like show up attached together, like this:



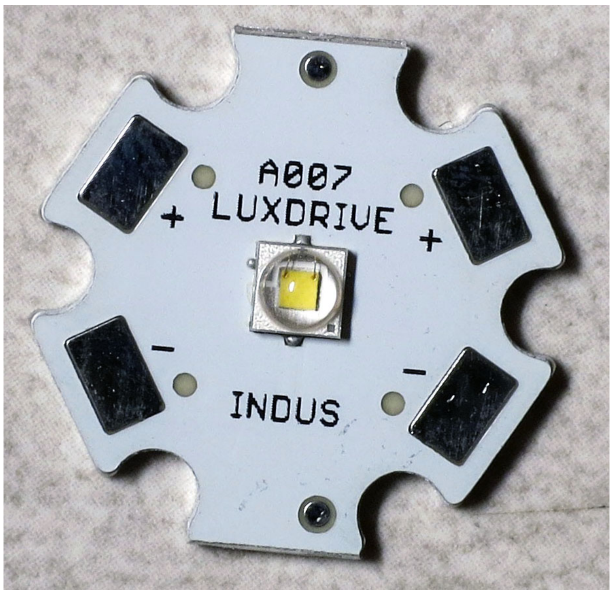
Just bend them back and forth along the scores until they detach from the metal strip and each other.

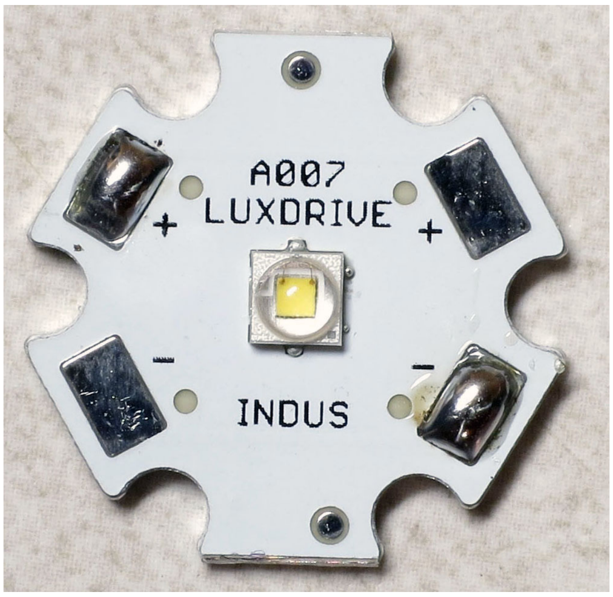
You're going to be soldering wires to one of the plus and one of the minus pads. There's a guide to soldering these star LEDs at LEDSupply.com, along with a YouTube video demonstrating the technique.

But I'm recommending a few changes from their approach for the LEDs for the RTI system:

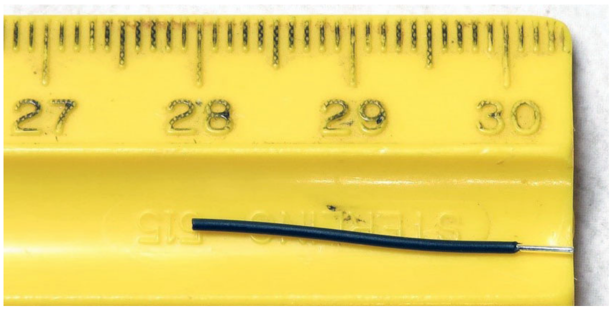
1. Solder wires to opposite pads, not adjacent ones.
2. Don't tin the wires - in this case, they're so thin (24 AWG) that it's not effective.
3. If you have one, use a chisel tip on your soldering iron.
4. When heated, lay the chisel tip against one of the pads so that the flat part is roughly perpendicular to the surface of the LED pad. Then apply the solder wire to the place where the flat part of the chisel tip and the LED pad touch.

The solder should melt and flow smoothly across the pad to create a nice blob across the pad. Repeat on the other pad, then repeat for every single LED you have.

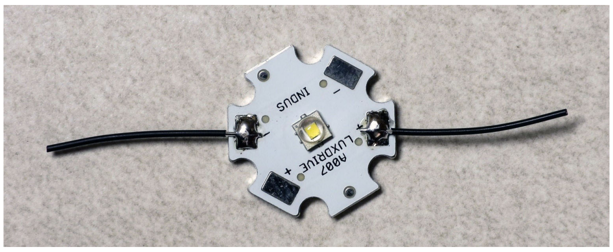

A007
LUXDRIVE
+
+
−
−
INDUS


A007
LUXDRIVE
+
+
−
−
INDUS

Grab your Kynar 24 AWG wire spool, and strip about 3/16" of insulation (about 3-4mm) off the end. You can make it a bit longer, but try not to make it shorter. Then cut off about 1" of the wire from the spool (about 2.5cm), including the bare lead.

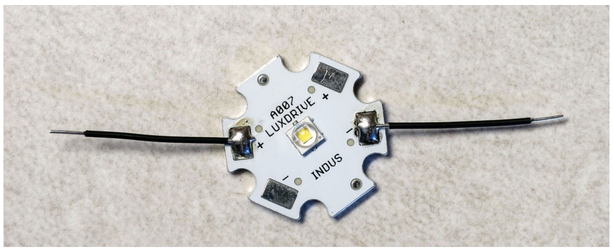


Repeat this so that you have two of these wires for every LED. Hold the bare wire lead against the solder blob, and press the soldering iron down on the wire to solder the bare wire lead to the solder blob on the LED star pad. Both wires should stick straight out from the LED pads, in line with each other:



If you mess up, just remove the solder and wires with solder wick or a solder sucker, and try again.

Now strip about 3/16" / 3-4mm off each of the two LED wires:



Exact length of these wires isn't critical, so if you strip off too much, don't worry about it, just clip it off.

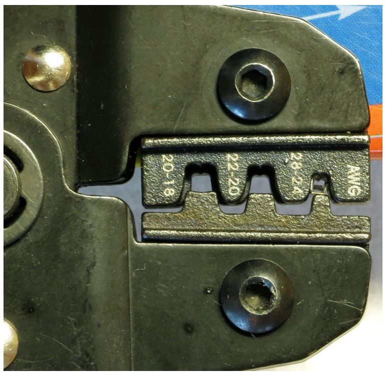
Now you're going to crimp male Dupont pins onto each of the bare wire leads. Hopefully you coughed up the money for a ratcheting crimper, because otherwise this is going to be a bit painful. You might try practicing the following approach with some scrap wires before trying it on a real LED; it may take a bit of practice to figure it out. Here's a Dupont pin pic:

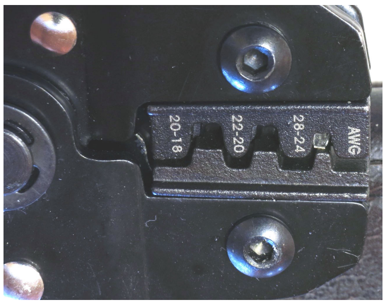
The pin part is on the right, and there are metal "wings" on the left that the crimper will push down onto an inserted wire and insulation to hold it in place. See that squarish bit in the middle? You want to make sure that you don't stick that into the crimper, but everything to the left of it can go in. Here's the approach I use:

1. Insert the clamp part of the male Dupont pin into the notch marked AWG 28-24, the one furthest to the end. It should go in so that the "wings" you see above face down towards the bottom of the notch. Make sure you don't insert it too far.
2. Start ratcheting down the crimper, click by click, until you see the "wings" start to bend into a rounded square shape at the end. On my crimper, that takes 4 clicks. The pin will now be held firmly in place by the crimper, with the pin sticking out on one side:

And the "wings" bent into a rounded square shape on the other side (look closely in the following pics and you'll see that):

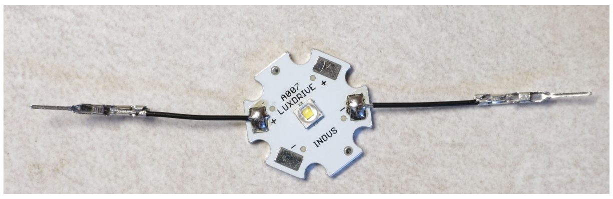
0.25
0.5
1.0


20-18
22-20
28-24
AWG


20-18
22-20
28-24
AWG

Now insert the bare lead on one of the LED wires into that rounded square so that just a bit of the insulated part goes inside the rounded square. Not super-critical here, but there will be crimps in upcoming steps where you don't want it to go in too far. It's best to hold the star LED face down for this step, with the LED facing down and the bare metal of the star facing up - this will make an upcoming soldering step a bit easier.

Once the wire has been inserted into the proper position, squeeze the crimper completely to finish the crimping process. The crimper ratchet should release, and you can pull the wire with crimped pin out easily. Repeat on the opposite lead, and you should wind up with a wired star LED that looks like this:

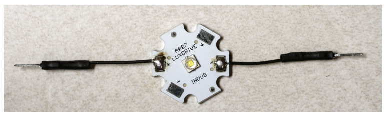


Notice how the open part of the pin faces up, and you can see the crimped wire inside the pin. In principle, the crimping should be adequate to hold the wire in place and ensure a good electrical connection. However, probably because I don't do it well, I've had problems with that. So I usually add a bit of solder to hold the wire in place. Use a narrow tip on your soldering iron, and touch the heated tip on top of the place where the wire is crimped in place.

Then lightly touch the solder to the area with the crimped wire until just a bit of it flows into place to solder the wire. Too much solder won't hurt with these male pins, but when you solder female pins in an upcoming step, too much solder can clog up the hole and cause problems with later insertion of a male pin. So this is a good time to practice the technique to minimize the amount of solder.

The final step is to add some insulation to cover up all of the metal Dupont pins except for the end pins that will plug in to female pins. Cut two 1/2" (12 mm) pieces of 2mm heat shrink tubing for each LED:

Place them over the crimped pins, and heat them to shrink them:



How to heat them? There are two common approaches:

1. Wave a flame, typically from a cigarette lighter, briefly underneath the heat shrink tubing. Not crazy about this approach because of the burn danger, and also because if you hold it too long you can melt the tubing instead of just shrinking it. But it is cheap and readily available.
2. Use a source of hot air. I have used a standard hair dryer to do this successfully, but it requires the "High" setting, and you have to hold the heat shrink right next to the heating element. A heat gun, if you have it, works much better, since it reaches higher temperatures than the hair dryer. You can find one online for

less than $20, and sometimes even less than $10. Also useful for stripping paint, loosening bolts, and even starting charcoal fires.

You’ll need to repeat the process for every LED you have, so that they all look like the picture above. Yes, all of them.

CHAPTER 9

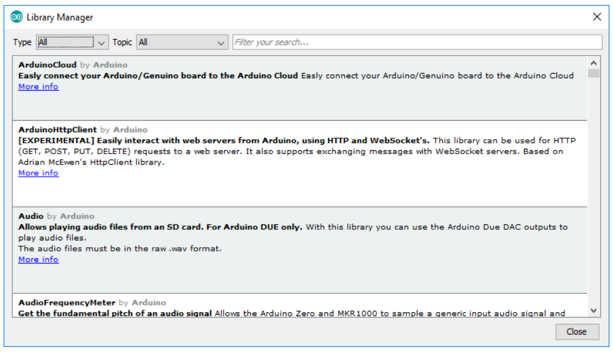
# Testing the system components

The RTI system uses an Arduino Mega Board as the system controller, to which you will need to upload various pieces of software for system testing, calibration and control. This section is a quick rundown of what you will need for this, including some special configuration issues.

First, go to the Arduino website, and download the Arduino IDE (integrated development environment) appropriate for your computer's OS; there are versions for Windows, Mac, and Linux. Install the software, then run it – if successful, it will start up either with a basic test program or an empty program window.

The Arduino IDE comes pre-loaded with a number of special libraries to support common hardware and software requirements. The RTI-Mage control software uses one of the preloaded libraries, SoftwareSerial, to support the optional Bluetooth HID adapter that triggers camera shutters for those cameras controlled by a PC (e.g. USB microscopes). But there are several additional libraries that need to be installed to support the optional OLED display, and the IR wireless remote shutter capability.

For OLED support, from the software menu select *Sketch* -> *Include library* -> *Manage libraries* . . . . This will bring up the Library Manager screen:



**ArduinoCloud** by Arduino
**Easly connect your Arduino/Genuino board to the Arduino Cloud** Easly connect your Arduino/Genuino board to the Arduino Cloud
More info

**ArduinoHttpClient** by Arduino
**[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's.** This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian McEwen's HttpClient library.
More info

**Audio** by Arduino
**Allows playing audio files from an SD card. For Arduino DUE only.** With this library you can use the Arduino Due DAC outputs to play audio files.
The audio files must be in the raw .wav format.
More info

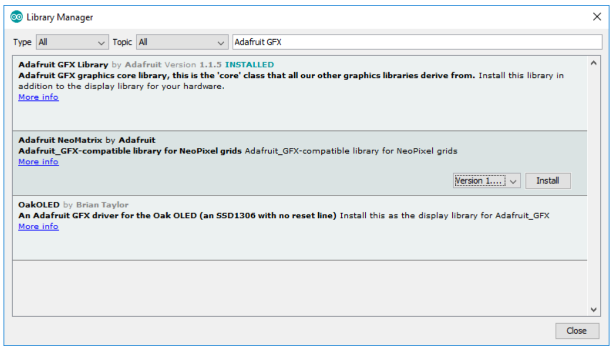
**AudioFrequencyMeter** by Arduino
**Get the fundamental pitch of an audio signal** Allows the Arduino Zero and MKR1000 to sample a generic input audio signal and

You want to install two libraries needed to drive the OLED display. The first one is the Adafruit GFX Library, which is a general graphics library, so type *Adafruit GFX* into the search box:

To install the library, click on the one you want and then press the *Install* button at the lower right. I already have the GFX library installed, so I've selected the next one down to show the Install button.

Now do the same for the Adafruit SSD1306 library; this is the actual one that runs the OLED, calling routines from the GFX library as needed.

The libraries are stored in the main Arduino data directory; on my Windows system, this is a folder called *Arduino* located in my Documents folder (not sure where on a Mac or Linux system, but it should be easy to find. There is



**Adafruit GFX Library** by **Adafruit** Version **1.1.5** **INSTALLED**
**Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from.** Install this library in addition to the display library for your hardware.
More info

**Adafruit NeoMatrix** by **Adafruit**
**Adafruit_GFX-compatible library for NeoPixel grids** Adafruit_GFX-compatible library for NeoPixel grids
More info

**OakOLED** by **Brian Taylor**
**An Adafruit GFX driver for the Oak OLED (an SSD1306 with no reset line)** Install this as the display library for Adafruit_GFX
More info

a subfolder called *libraries* where the critical library files are stored. After installing the two libraries above, you should see folders for both of them.

Now go into the folder for Adafruit SSD1306, and open the file called *Adafruit_SSD1306.h* in any text editor. Near the top, you should see these three lines:

```
//   #define SSD1306_128_64
     #define SSD1306_128_32
//   #define SSD1306_96_16
```

These lines specify the display resolution used by the library. Two slashes in front make the line a comment, so for the above lines, the default display resolution is 128 x 32. The 0.96" OLED display in the components list has a 128x 64 resolution, so you'll have to comment out the 128x32 line, and then remove the comment slashes from in front of the 128x64 line:

```
     #define SSD1306_128_64
//   #define SSD1306_128_32
//   #define SSD1306_96_16
```

Save the file, and you're probably done with that library for now (with one minor possibility that you'll have to go change one other OLED parameter, but I'll get to that in a future step).

There's one more library you'll need to install, but this one isn't in the Library Manager. This is the *multiCameraIrControl* library written by Sebastian Setz. Sebastian's original site is now down, but you can retrieve it from the files section on the Hackaday project page (Creative Commons license, so no problems with doing this).

This library is needed to control the wireless IR remote shutter capability of the system, which works with cameras from Canon, Nikon, Sony, Olympus, Pentax or Minolta that have an IR sensor. Download the zip file from the site above, and unzip the folder *multiCameraIrControl* into the libraries subfolder of the Arduino folder. Restart the Arduino program, and that library is ready to go.

Even if you don't use the OLED or IR capabilities, you still need to install the libraries. The program will call for them, and if it doesn't find them, it won't compile successfully. You could remove references to them in the program, but much simpler just to leave them in there; even if the OLED and IR hardware isn't installed, the Arduino won't care.

To install a program onto the Arduino, first load it into a program window by selecting it from the *File* -> *Sketchbook* menu. Connect your Arduino Mega to a USB port on your computer; if it's the first time, the computer will have to install drivers for it. From the *Tools* -> *Board* menu, select Arduino Mega 2560. On the *Tools* -> *Port* menu, once the Arduino has been recognized by the computer, there should be a *COM* port listing for it; select that port to tell your computer where to send the program data.

Once the configuration is done, click on the right-facing-arrow icon near the top to upload the program to your Arduino. Status messages will show up at the bottom, and if everything works successfully, you should see a *Done Uploading* message at the bottom. If you get an error message, double-check to make sure you've selected the right Arduino board and COM port.

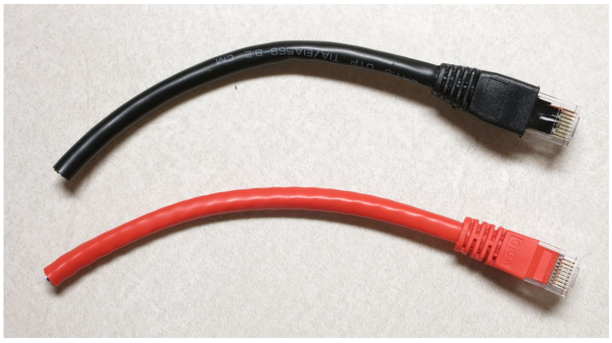
## 9.1 Preparing the LED connect cables for testing

Now that all the LEDs are wired up, there are two simultaneous tests that need to be done:

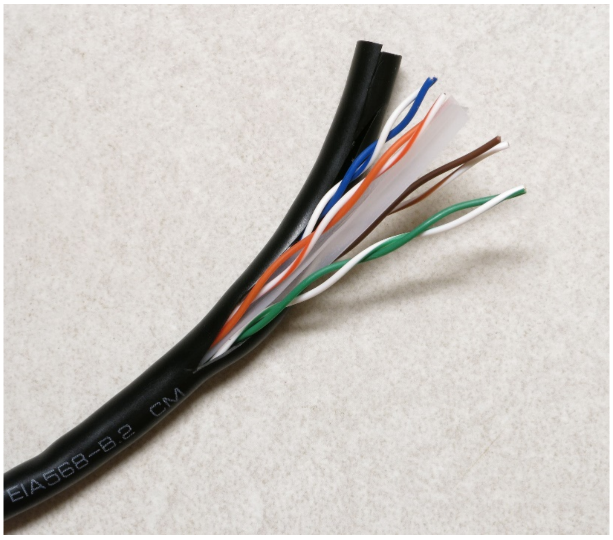
1. Test the LEDs to make sure they're all wired up and working properly
2. Test the MOSFET/CAT4101 driver board combination to make sure it's working

In order to do this, there needs to be a way to connect the LEDs to the driver board combo.

Take the two Ethernet cables you have (red and whatever other color you've chosen), and cut about 6" (16 cm) off one end, including the jack.



Use a sharp utility or razor knife to slit open the external insulation at the cut end of each cable (cut about 2-3" down the length of the cable), taking care not to cut the wires inside. Then remove the insulation.



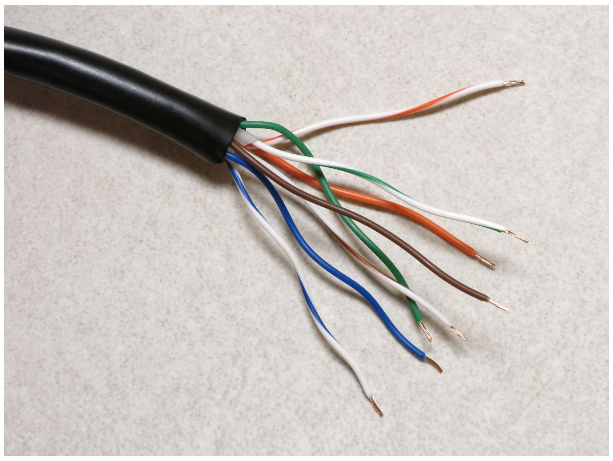
Trim off the external insulation; if the cable has internal plastic ribbing, like the one above, trim that off as well. Separate the 8 wire strands, and strip about 3 mm of insulation off the end of each one.

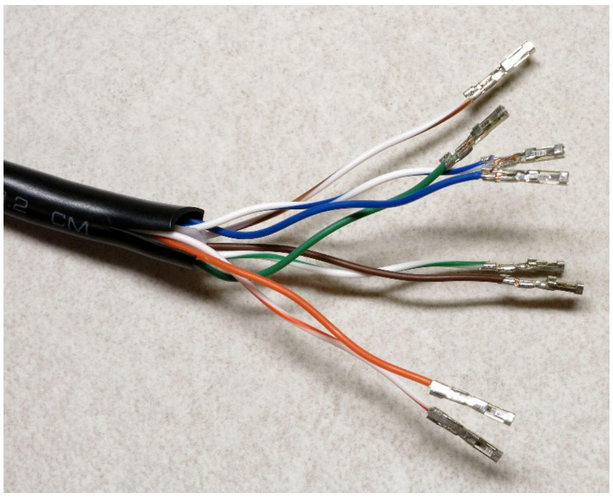
Now crimp female Dupont pins onto the end of each wire, using the same technique as when you crimped male pins onto the LED wire. Here's a female pin, with the crimpable end on the left:

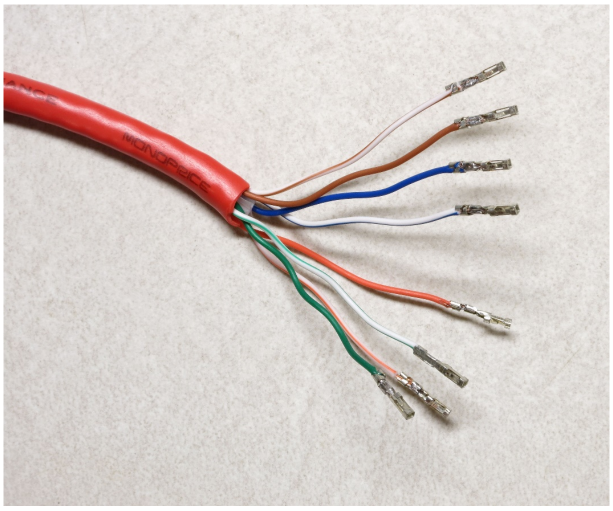
When crimping be careful not to insert the wire too far into the Dupont pin, as that may make it difficult later to insert a male pin into the female pin socket. I usually solder the wire to the pin as well for a more secure hold, but be careful with this. Unlike the Kynar wire you soldered to male pins in the previous step, this insulation melts very easily. You just want to touch the soldering iron to the wire crimped in the pin, and immediately touch the solder to the tip of the iron until it just flows a bit; then lift the iron up right away. If you get it wrong, just cut off the pin, strip the wire insulation, and try again. This is good practice for an upcoming step.

Repeat for the other Ethernet cable:

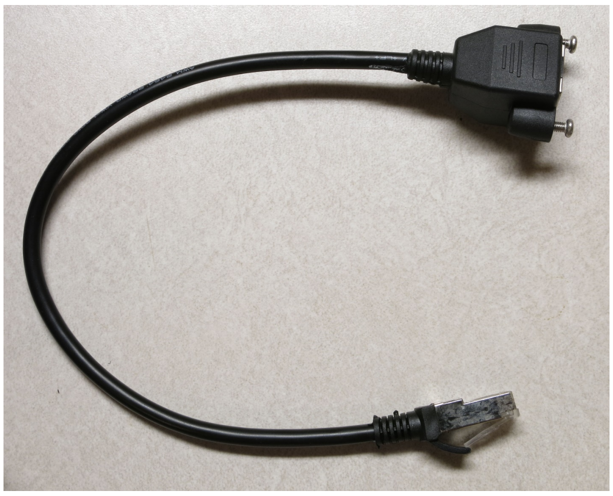
The ground cable connector (black in this case) will plug into the Ethernet jack on the CAT4101 board. The red positive voltage cable needs to plug into the MOSFET driver board, but it can't do so directly because there's no





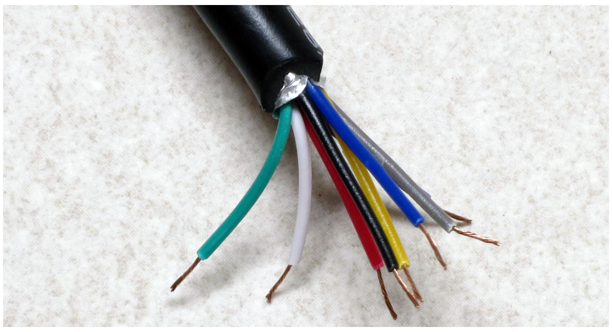


Ethernet jack on that board (no room for it). So we have to modify one of the two Ethernet panel jack cables to plug into the 8-pin male header on the MOSFET driver board. Here's what that cable looks like:



You want to cut the jack end off as close to the end as possible; that's the bottom connector in the previous picture. Take care not to cut off the female panel end, the top one above. If you do, you will be very sad, as you'll have to get a new one. Also, you have two of these Ethernet panel jack cables; only cut the jack end off of one of them.

Once again, use a sharp utility knife or razor to cut the insulation about 2" down the length of the cable on the cut end, then trim off the cut insulation, and strip about 3mm off the end of each wire.



Now crimp female Dupont pins onto the end of each of these wires, and solder them. As before, be careful not to insert the wire too far into the pin crimp end, but make sure that the insulation does go slightly into the pin crimpers. Don't have a picture of this, but it will look similar to the ends of the Ethernet cables you did before.

Now there's a small problem you have to figure out. If you take a look at the first two cables in this step, there's a good chance they have different wire colors than the Ethernet panel cable you're currently working with. So you'll need to figure out which wire color on the Ethernet panel cable corresponds to the matching wire on the Ethernet cable. That's because each wire connects to a numbered pin on the Ethernet jack, and you want to make sure to have matching pin numbers everywhere. Here's the numbered wiring color code for the Ethernet cable (standard T568B cable):

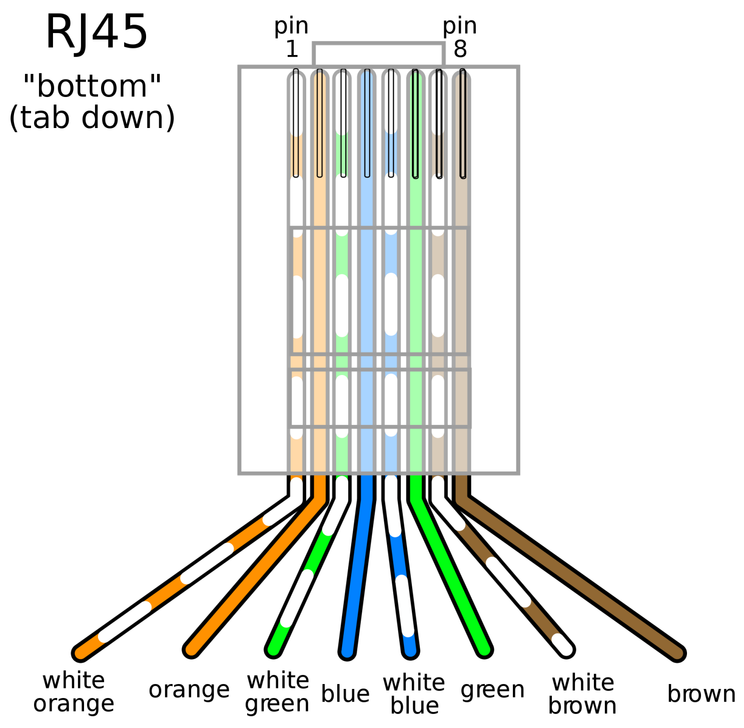


Fig. 1: Source: http://www.polesio.co/utp-cable-wiring/wiring-imgs-199686/

To figure out which wire on the Ethernet panel cable corresponds to the correct wire on the Ethernet cable, plug the Ethernet cable into the panel jack, then use the continuity checker on your multimeter to figure out which wire is connected to which. In this case, I wound up with the following correspondences:

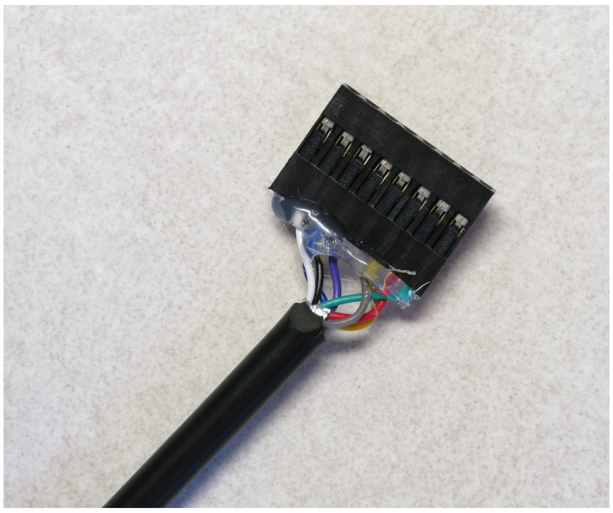
| | | |
|---|---|---|
| Pin 1 | White/Orange | Green |
| Pin 2 | Orange | Red |
| Pin 3 | White/Green | Yellow |
| Pin 4 | Blue | Gray |
| Pin 5 | White/Blue | Purple |
| Pin 6 | Green | Blue |
| Pin 7 | White/Brown | White |
| Pin 8 | Brown | Black |

Your wire matchups may differ; just be sure to write down which color corresponds to which pin.

Next, grab the 8-pin 2.54mm single row female pin header, and insert the female Dupont pins into the large end of the header in pin order, i.e. #1 at the top, then #2, #3 all the way down to #8 at the other end. The pins need to be pushed all the way down into the header, not just until the end of the pin is flush with the top of the header. You may need to use a small screwdriver, or a thick wire, to push it all the way in. The top of the female Dupont pin (with the crimped wire visible) should be inserted into the header so that it will be visible in the header's open hole, like this:

A couple of points here:

1. I cut the wires a bit shorter than 2", and it was a pain to get them to fit; that's why I recommend at least 2" in length.
2. I have the wires in correct order here, from 1 on the right to 8 on the left, but I would have done better to put them in in the reverse order. See that little arrow at the top left? That's supposed to indicate pin 1. It's not a



big deal, as long as I remember to insert this cable into the 8-pin male header on the MOSFET driver board so that the #1 wire (green) is at the top. But doing them in reverse, I wouldn't have to remember which color wire is the #1 wire.

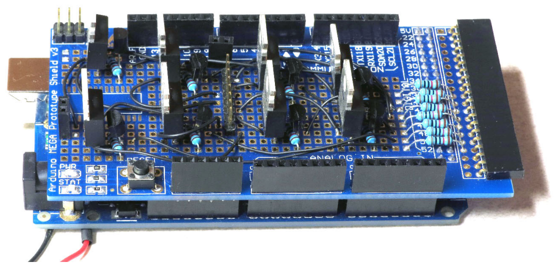
3. Test the connections using a multimeter, to make sure that none of the wires got damaged when you inserted them into the female pin header.
4. I used some hot glue to glue the wires in place so that they wouldn't pull out by accident, and I'd recommend that once you've tested the connector to make sure it's working properly, you use some kind of glue to do the same. When inserting or removing this connector, never pull it out by the wires – always pull on the connector.
5. If you can't find an 8-pin 2.54mmm single row female pin header, you can use 8 1-pin female Dupont connectors, one for each wire. It'll be a bit of a pain as you'll have to connect each of the 8 wires individually to the MOSFET driver board, but it can be done.

## 9.2 Test of the electronics and LEDs

Now that you've got all the cables done, it's time for the big test of the electronics and LEDs.

Connect the Arduino Mega to your PC with the USB cable, and upload the testing program System_tester.ino into the Arduino Mega controller using the Arduino IDE; you'll find that program in the Files section. Disconnect from the cable when done.

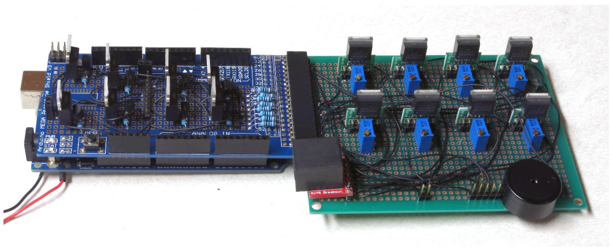
Plug the MOSFET driver board shield into the Arduino. Do this slowly and carefully – it's easy to misalign pins, or bend them when inserting the shield.

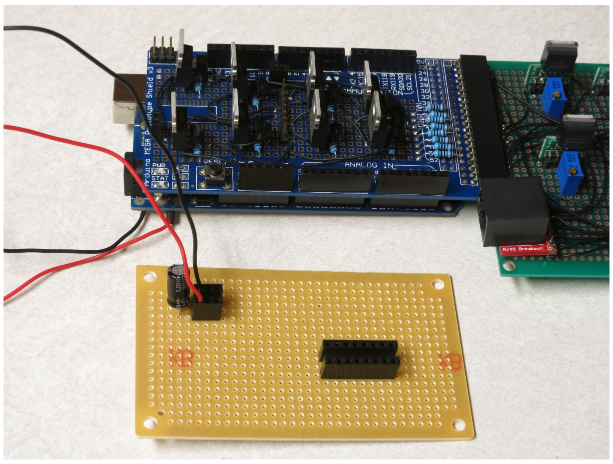


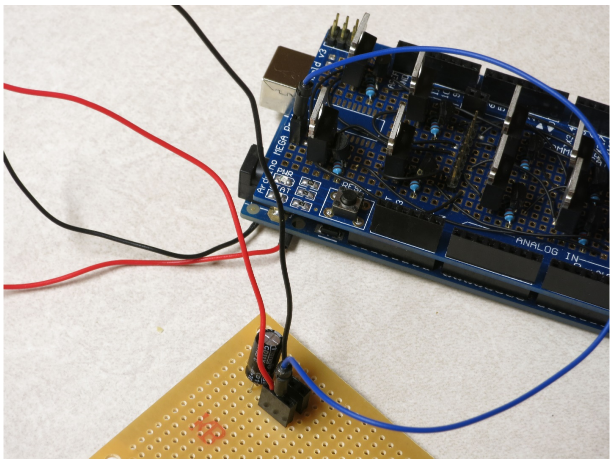
Plug the CAT4101 board into the MOSFET driver board.

Plug the black ground wire soldered to the Arduino power input into the top left header of the power strip board (next to the capacitor half marked with a white stripe), and plug the red positive wire into the bottom left header.

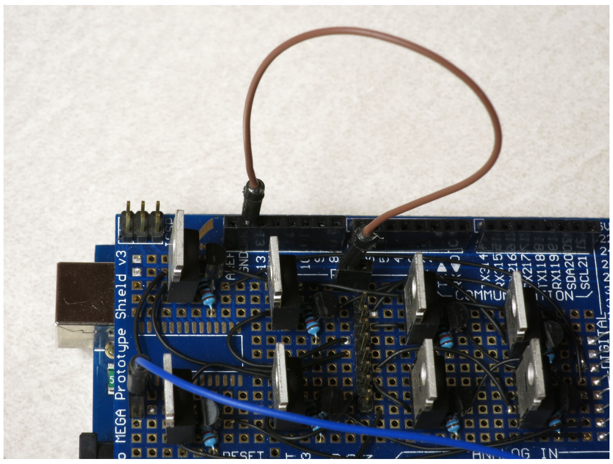
Connect a jumper wire between the positive power strip header and the two pin header on the lower left side of the MOSFET shield board (the blue wire in this picture):



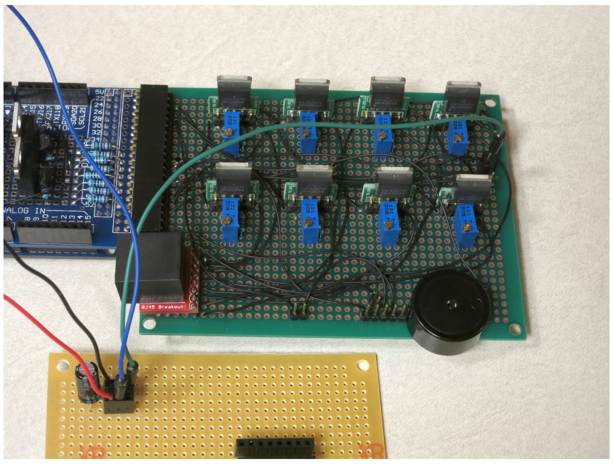




Connect a ground header on the MOSFET driver shield to the two-pin female header in the top center of the board using a jumper wire, the brown wire in this picture.



Connect a jumper wire to the ground power strip header on the power board, and the two-pin header on the far right of the CAT4101 board (the green wire in this picture):



Plug the unmodified Ethernet panel cable into the Ethernet jack on the CAT4101 board, then plug the ground Ethernet tester cable into the panel jack.

Plug the modified Ethernet panel cable, with the 8-pin female header, into the 8-pin male header on the MOSFET board. Make sure that pin/wire #1 is at the top.

Plug the red Ethernet cable with the female Dupont pins on the end into the modified Ethernet panel cable connected to the MOSFET driver board; this will be the positive voltage supply to the LEDs.

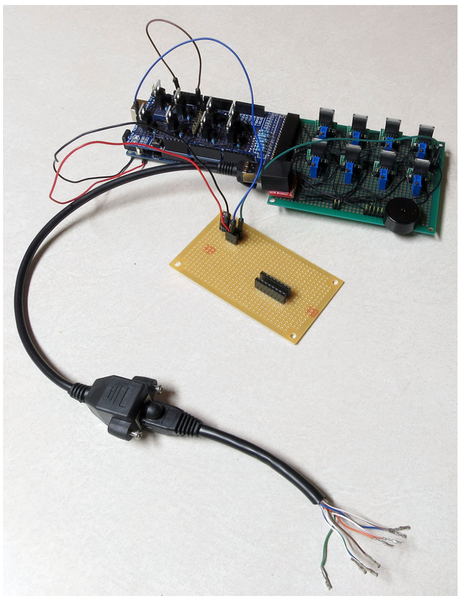
Connect multimeter leads to the male header pins at the bottom of the CAT4101 board, and set the multimeter to a voltage scale greater than 5V DC. When the tester is running, it will turn on and off voltages to two of these pins, and supply constant 5V to a third pin. Here's the connection to the pin that supplies 5v to the USB cable or IR LED to fire the camera shutter, which should cycle on and off:

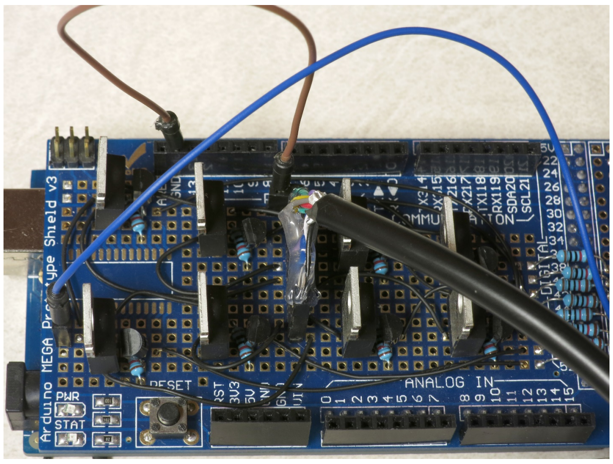
And here's a closeup of that:

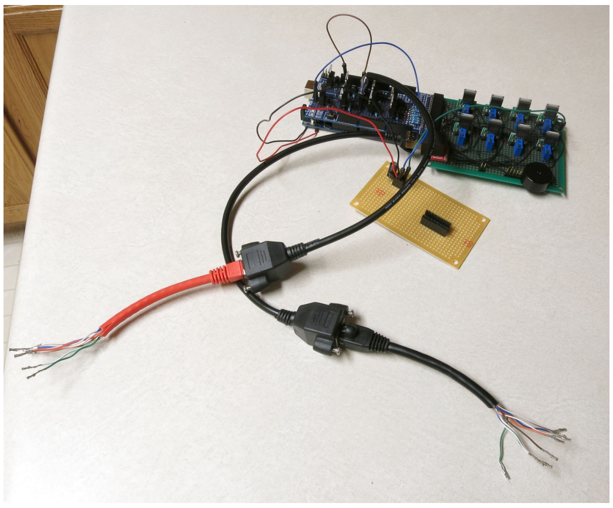
Here's a closeup of the testing connection for the servo control pin; this one should also cycle on and off:

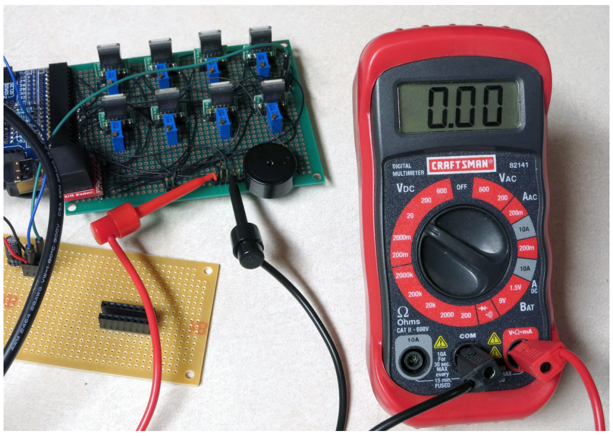
And here's the connection for the constant 5V that powers the servo:

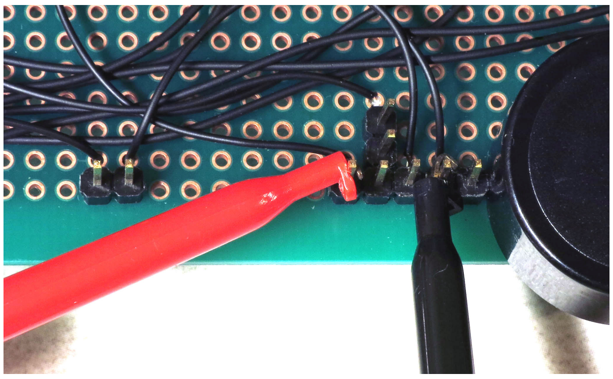
Grab 8 LEDs, and plug the positive LED wire into the positive Ethernet connector wires (should be from the red one, coming from the MOSFET driver board). Then plug the negative LED wire into the matching wire from the ground Ethernet connector from the CAT4101 board. In other words, orange/white to orange/white, green to green, white/brown to white/brown, etc.

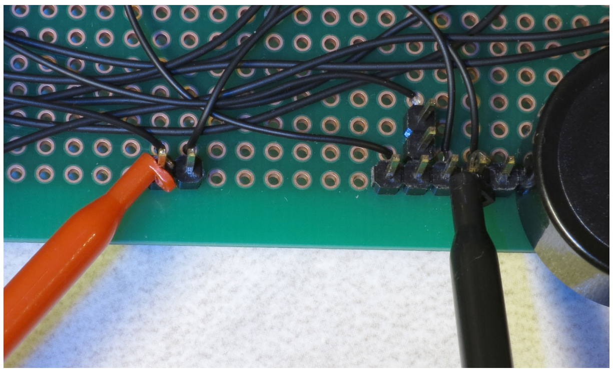


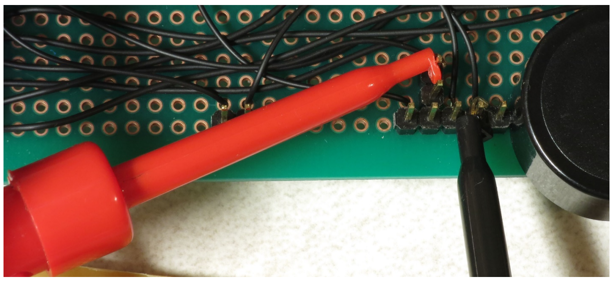

Arduino MEGA Prototype Shield v3
PWR
STAT
RESET
ANALOG IN
COMMUNICATION
DIGITAL



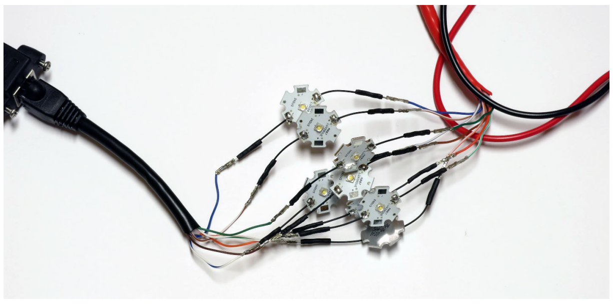

0.00
DIGITAL MULTIMETER
CRAFTSMAN
82141
VDC
VAC
AAC
OFF
Ohms
BAT
COM
V-Ω-mA



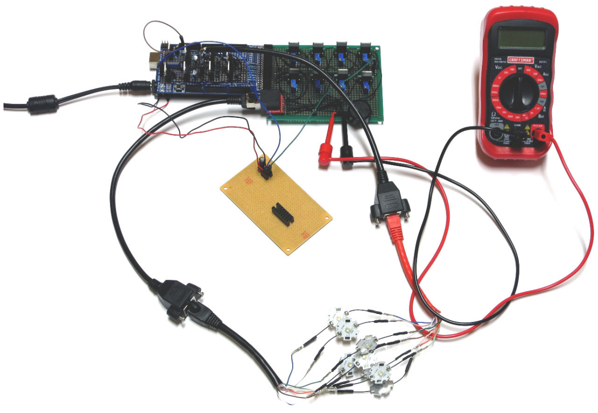




You're connecting the #1 column wire to the #1 row wire, the #2 column wire to the #2 row wire, etc.. The tester program will turn on matching row/column number drivers simultaneously, which should light up that LED.



Now you're ready to plug in the 9V power supply into the Arduino – the full setup should look something like this:



If everything is working correctly, the beeper should sound, and each LED should turn on individually in sequence for 0.25 seconds (250 milliseconds), then turn off. While the LED is on, the multimeter voltage should read zero when connected to either the USB/IR pin or the servo control pin; when the LED is off, it should read zero. When connected to the servo power supply, it should always read 5V.

If everything is working as it should, it will look something like this video.

If it's not working, here's some quick things to check:

1. Make sure the + leads on the LEDs are connected to the wires from the red cable, which should in turn connect to the MOSFET driver board.
2. Make sure the LEDs are connected to wires of matching colors.
3. Make sure the female 8-pin header is firmly connected to the MOSFET driver board, and that the #1 pin connection is at the top.
4. Double-check to make sure that all the jumper and power wires are correctly positioned, and firmly seated in the female headers.
5. Make sure the power supply is properly plugged in.

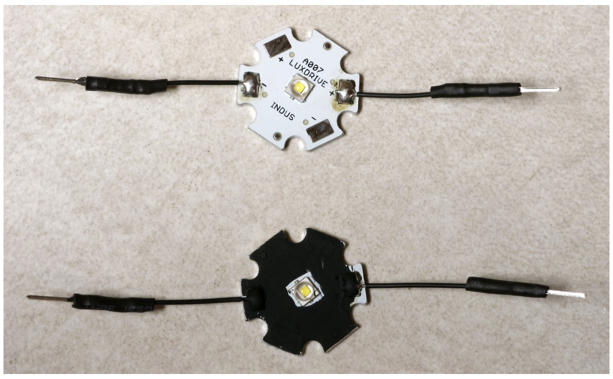
If it is working, then once you've cycled through all 8 LEDs at least once, unplug the power supply and disconnect all 8 LEDs, setting them aside. Select a different set of LEDs, and connect them to the wires as you did the first set.

Plug in the power supply, and let the LEDs run through at least one full cycle to make sure they're all working. Once that's done, repeat the process until you've tested every LED to make sure they're all working correctly.

CHAPTER 10

# Installing and wiring the LEDs

Once you've tested all the LEDs, it's time to put them in the dome. But there is one final consideration. The interior of the dome was painted flat black to minimize light scattering. But you're about to install LEDs that are mostly white in color, and will scatter light quite effectively. I calculated that for the 12" dome I'm building here, the 48 LEDs will cover about 10% of the total area inside the dome, which could lead to quite a bit of scattered light. To reduce that, I paint the LEDs with a non-conductive flat black paint:



However, since I still need to know which wire is positive and which negative/ground, I leave a little bit of the end next to the positive lead wire unpainted so that I can keep track of the polarity.

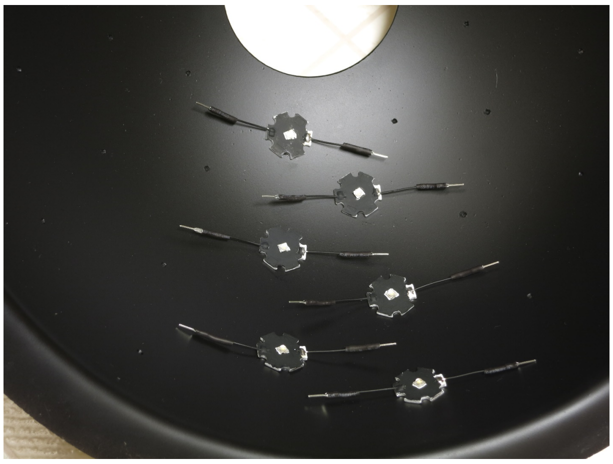
To install the LEDs inside the dome, I put a small dab of adhesive on the back of the LED. I use a silicone-based adhesive because I've found that if you don't use too much of it, it's possible (with some effort) to pry off the LED if something goes wrong. I then press the LED on top of one of the position markings you created way back in step one, with the positive lead facing to the right as you look directly at it:

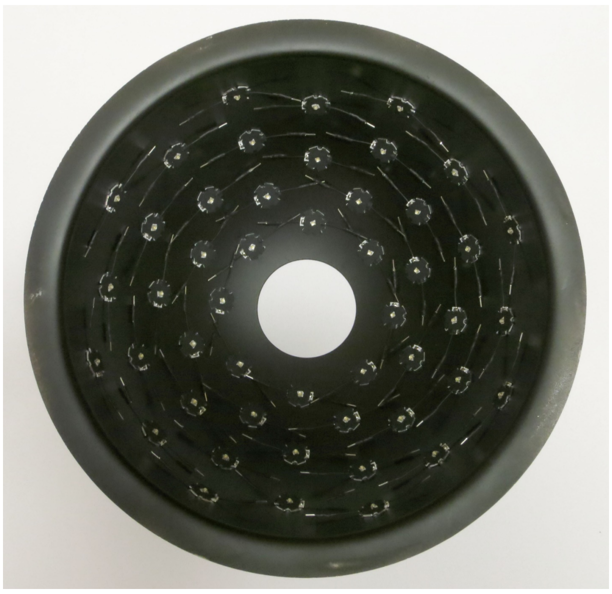
Continue the process until all of the LEDs have been installed:

Before the adhesive dries, check to make sure that all of the LEDs have the positive lead facing to the right, and the wire leads face left and right when you look directly face on at the LED. Once you're sure everything is oriented correctly, set the dome aside to allow the adhesive to set for at least 12 hours, and preferably 24.

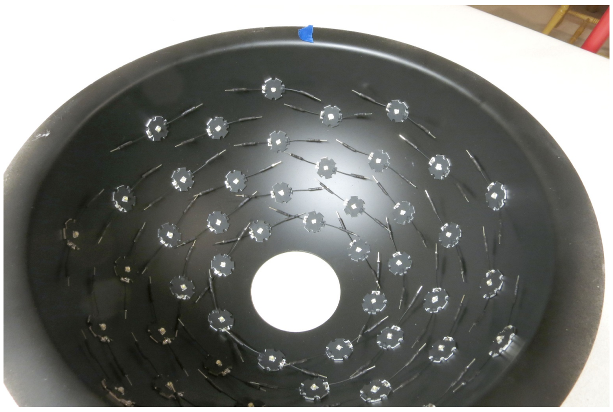
It may look like an intimidating amount of wiring needs to get done, but don't worry – there is a method to the wiring madness that keeps it from becoming too complex to keep track of.

Once the adhesive you've used to attach the LEDs to the inside of the dome is dry, it's time to do the wiring that will control and supply power to the LEDs. But there's a bit of prep work that needs to be done first.



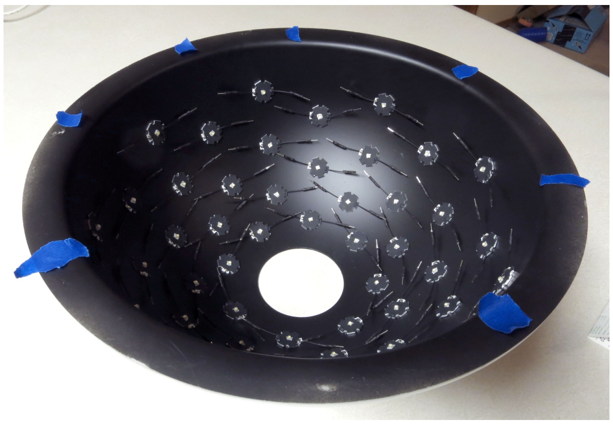


Flip the dome so that the interior faces up, and place a marker (masking tape here) on the bottom of the rim in a position that bisects two of the LED columns:



As you can see, one of the LEDs in the lowest row (closest to the base) is to the left of the marker, and the LED in the next row up is to the right. That's actually not a critical reference position, you could wire the dome with the marker to the right of the first and the left of the second. But for now, probably best to follow this convention, since all of the wiring in these instructions (and the pictures) will work off the above marking. Beyond that, pick any pair of columns you want – doesn't matter.

Next, place additional markers on the base of the rim, between LEDs on the lowest row (above those on the next lowest row), three on each side as shown in these pictures:



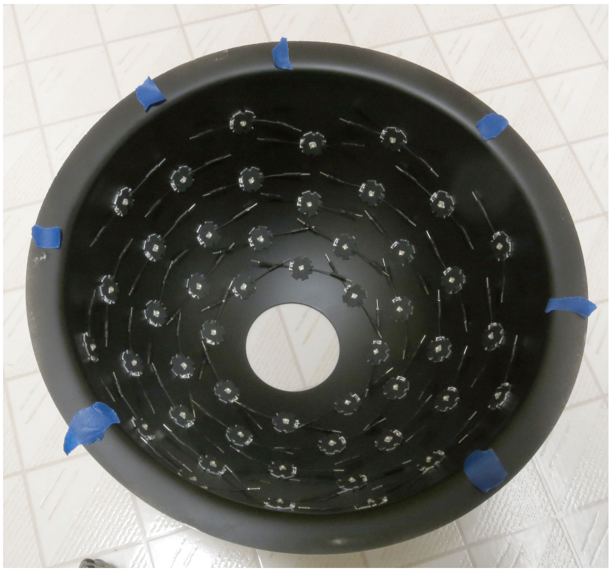
Now flip the dome over, and put marks on the rim of the dome corresponding to the tape marks; doesn't require absolute precision, just as close as you can. Differentiate the first mark you put down from the rest somehow (an asterisk in the example below):

For all the marks except the first one, mark the position where you will drill two holes above them, the first about 3/8" above the bottom, the second about ½" above that. These will be for zip ties that will guide and hold some wires in place.

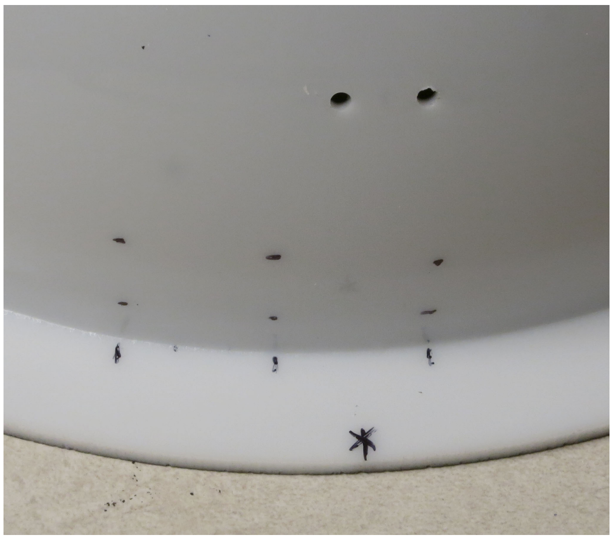
In the next pic, there will be similar ½"-spaced marks directly above the first reference mark, placed about one-third and two-thirds distance from the base to the top of the dome. These are also for zip tie holes to hold wires in place. However, after finishing the wiring, I realized that these weren't really necessary for the 12" diameter dome I'm using here, and probably wouldn't be necessary for any dome up to at least 18" in diameter; the wires are short enough and stiff enough that they don't need to run through guides to stay in place. So I have those marks in place, and wound up drilling them out, but didn't install any zip ties through them. For smaller domes you can skip these.

Pull out your electric drill and 1/8" plastic drill bit, and drill holes through the marks you've just made. Use lubricant (vegetable oil is fine), and don't use too much pressure, or you may crack the dome.

If you're building a larger dome that requires vertical wire guides, drill those holes as well (didn't need them for my small dome, but didn't know that yet):



Now mark three additional sets of holes. First, put two marks 3/8" above the base of the rim, 1" apart, centered on the main reference point. Measure about 1" to the left of the leftmost hole, and put another mark there. Then, put two marks about ¾" above the left two marks, and one mark about ½"-5/8" above the rightmost mark. These holes are for zip ties that will hold the power cables in place.



Drill 1/8" holes with your plastic drill:

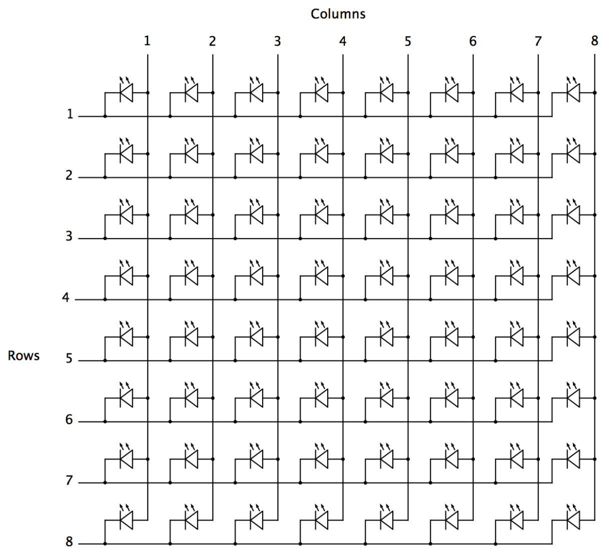
Now mark one position about ½" above the rim, directly above the main reference mark. Put a second mark about 5/8" above the rim, to the right. I have it about ½" to the right of the rightmost holes, but ¾" or even 1" probably would have been a better choice.

Drill a 1/8" starter hole at each of these marks, then use a step drill (Unibit) to drill ½" holes at each of these locations; this is where the power cables will go into the dome. The picture shows holes closer to ¼" in diameter, but I found those to be too small and expanded them later on to ½".

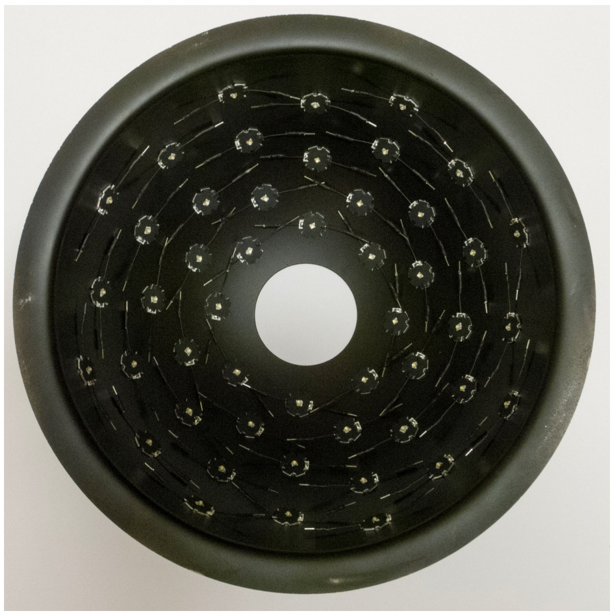
Clean up the debris from the drilling (and the marks you made). Now it's time to get wired.

Can't avoid it any longer – it's time to wire up all the LEDs in the dome into a matrix configuration, like the one shown schematically here:

This is an 8x8 matrix, where the rows get connected to ground control, and the columns connected to positive voltage control. For the RTIMage system, the P-channel MOSFETs control the columns/+ voltage, while the CAT4101s control the rows/ground (and also set the current). For the small 12" dome I'm building here, there are only 6 rows, but a full 8 columns, so two of the CAT4101 chips will remain idle with this system.



So here's a shot of the LEDs inside the dome, after being glued in place in a previous step:



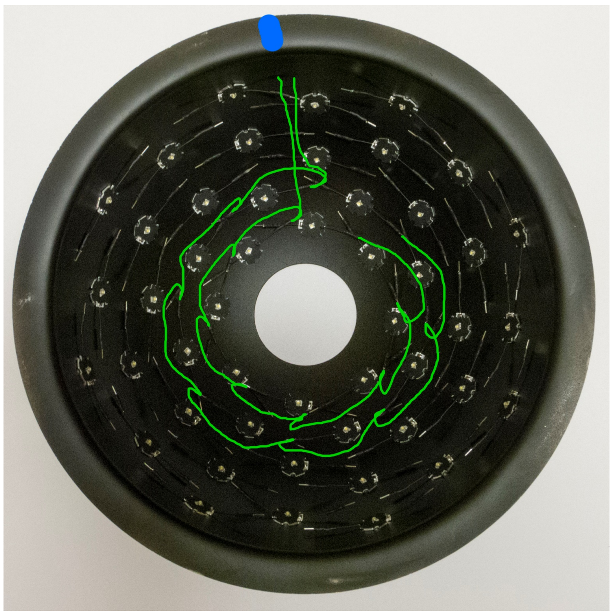
Even with only 48 LEDs, instead of the maximum 64 the controller can support, the prospect of wiring all these up may be a bit daunting. Not going to pretend that this is going to be fun – you're probably looking at 4-5 hours of tedious work. But I've come up with a system that I hope works reasonably well, keeps the pain level down, and lets you fix any mistakes fairly easily. Also remember that this is a small dome, 12" in diameter, so things are packed in pretty tight; with larger domes, there will be more space to work with.

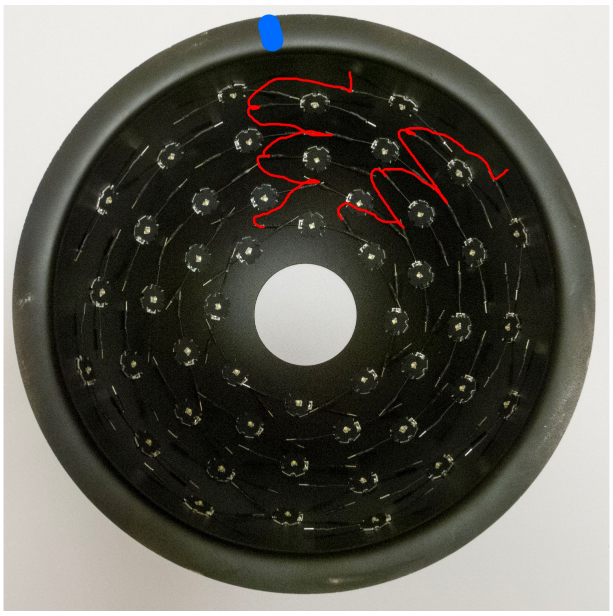
You're going to be connecting the LED ground leads in a row, and all the LED + leads in a column. Remember that when you painted the LEDs black, you left a little bit unpainted to mark the + side, but painted the -/ground side black, and you can see that above. So you'll be wiring up the ground rows like this:

. . . where the blue mark at the top represents the main reference mark from the previous step. The wiring from only two rows is shown here in green, but all of the remaining rows will be wired in a similar fashion – you connect all the ground LED leads in a single row together, and then the last wire needs to be long enough to reach within about an inch or so of the bottom of the dome.

Similarly, you'll be connecting all the + LED leads together in a column. Unlike the row, though, the column elements are a bit staggered, so you'll need to zigzag back and forth:

The last lead will be a short one at the bottom, which you'll be connecting to another wire.





**Warning:** Some of the upcoming pictures will show the + leads bent down towards the bottom, like this:

I had an idea that this would make the wiring a bit easier. I was wrong, and ultimately wound up bending them back. Just leave the + LED leads as straight as they were when you glued the LEDs inside the dome – it makes the wiring a lot easier.

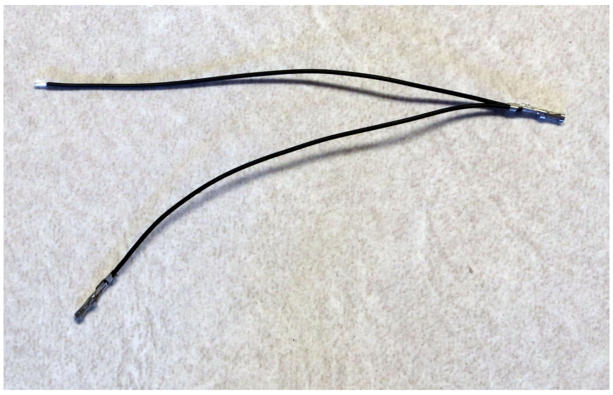
Let's start with the rows. You're going to have to connect 8 pieces of wire together, 7 bridging the 8 LEDs in the row, and the last one leading down to a hole you drilled near the base in the previous step (it's the hole at right in the picture above). What length should the wire be? The wire length will depend on the size of the dome, and the number of columns in the row.

You'll have to choose a wire length long enough to connect adjacent ground leads, plus some slack to let you bend the wire into a position that doesn't block light from the LEDs. However, a good first guess for the wire length can be obtained by measuring the distance between two adjacent LEDs in a row, then adding about 1.5" to that.

Start by cutting two pieces of wire that length, and stripping off 2.5-3mm of insulation off each end. Crimp a female Dupont pin onto the end of one wire, the same way you did in previous steps. Now comes a slightly trick step. You need to hold the uncrimped ends of two wires together, and then crimp both of them in a single female Dupont pin. The tricky part is that the crimp end of the Dupont pin is just barely big enough to fit two wires, but you have to stick them in just the right way.

The best way I found is to pinch the two wire ends together as close as possible, and then slide them into the crimper vertically (perpendicular to the general orientation of the crimper). Most times, you will feel them slide in a bit, and then stop as the insulation catches on the end of the pin. Don't crimp at this point; while it might work, I've found that more often than not the crimping doesn't hold, and one or both wires fall out.

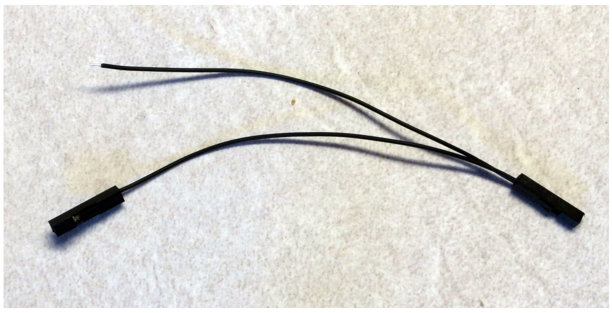
You need to get a little bit of the insulated part of both wires into the crimp end of the pin. To do this, you need to gently jiggle and twist the two wires in the crimper, until you feel them start to slide a bit further in. This is your signal that you've got the insulation in the crimp end, and you can finish the crimping process. These first two wires crimped together should look like this:

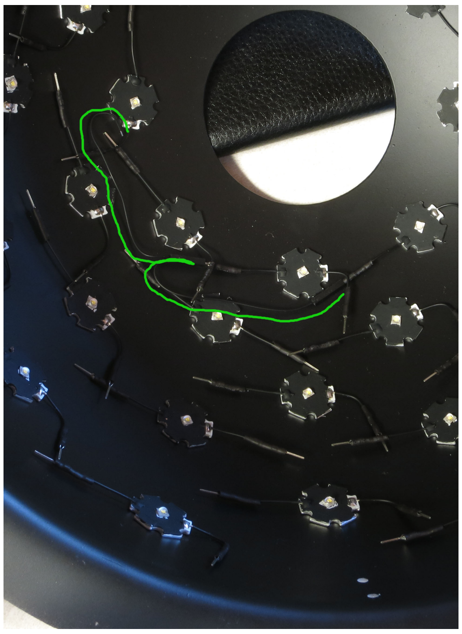


As in previous crimpings, put a bit of solder on the crimped end to secure the wires and ensure good electrical contact. Then take two of the 1-pin plastic female Dupont housings, and slide the pins into them. They may slide in easily to the end, but more often than not they will likely slide in partway and then stop. They need to slide in all the way to the end, otherwise the LED male Dupont pin will not be able to make contact with the female Dupont connector above. The best method I've found for dealing with recalcitrant connectors is to hold the wires at the base of the connector firmly, grip the plastic housing with a pair of needlenose pliers, then push the housing down towards the base of the connector; works pretty well. The pair of wires will now look like this:

Now you'll need to check whether your initial guess for the wire length (distance between LEDs + 1.5") was correct. Go back to the dome, and see whether the two connectors are able to reach the first two LED pins, plus a bit of slack in the wire to let you bend it into position. In the picture below, I've underlined the wires in green, showing that they do reach between LED leads when bent properly, with a bit of slack:

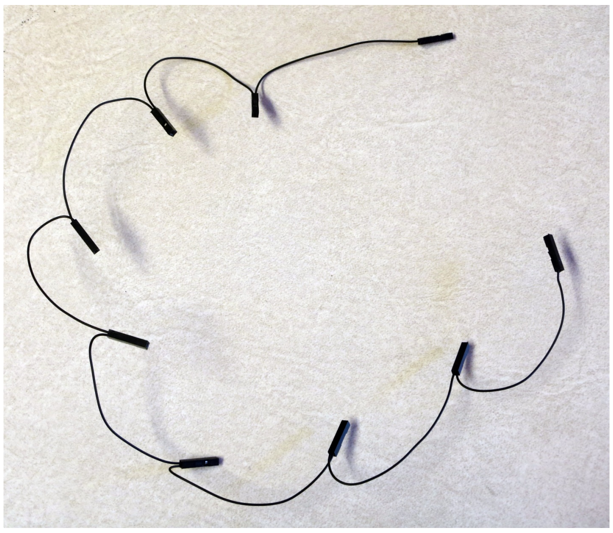
It may look from the picture as though I inserted the LED pin into the female connector. That's because I did - but you shouldn't. The pins are easy to bend, and if they bend and break you'll need to pry off the LED to fix it, which is not fun. Best to only insert the pins once, and there's some prep work that will make the pins slide in





more easily (more on that in a bit). For now, just check for the proper wire length. If it looks fine, then you can stick with that LED distance + 1.5” guesstimate for all of the remaining rows. If it seems like a close fit, feel free to add a bit more length to the wire – better too long than too short.

You’ve cut and crimped two wires for the top row; now cut 5 more wires of the same length to connect the ground pins on the remaining LEDs in that row, plus one more last wire long enough to reach from the last LED in the row down to the hole at the base of the dome. Keep crimping and soldering two wires together until you get to the last longer wire, then crimp and solder a female connector to the single end of that last wire. Stick the female pins into the plastic housings, bend the wires so that they’ll connect more easily to the LEDs, and you should have something that looks like this:



The first LED connector is at lower right, and the longer wire that should reach to the base of the dome is near the top. Stick a male Dupont pin into each end, then use a multimeter to make sure you have good electrical connectivity between both ends. For extra safety, you can check to make sure that every intermediate connector has electrical connectivity, but I’ve found that if the ends are electrically connected, the rest of the connectors are usually good as well.

One more thing. Take a male Dupont pin, and slide it in and out of each of the female connectors 2-3 times. I’ve found that this helps make the LED pin slide in more easily, making it less likely that it will bend.

Time to wire up the top row. Start with the LED on the end of the row, and slide the connector onto the LED ground pin. The safest way to do this is to hold the ground pin firmly flush against the side of the dome so that it can’t move, then hold the female connector/housing against the dome surface and slide it onto the pin by pushing against the back end until you feel it go in as far as it can. Be careful not to force it, as this might bend the LED pin. If you’re not sure about this, practice with some scrap connections outside the dome until you get a feel for it. Repeat this with every other LED ground pin in the top row. When you’re done, it should look like this (wires and connectors are paralleled by the green line):

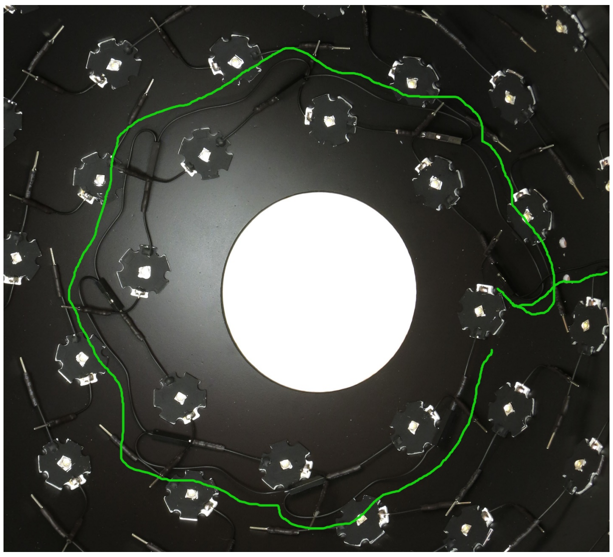
The last wire on the right is the lead that goes down to the base of the dome (out of the picture), and will ultimately be connected to one of the two power cables.

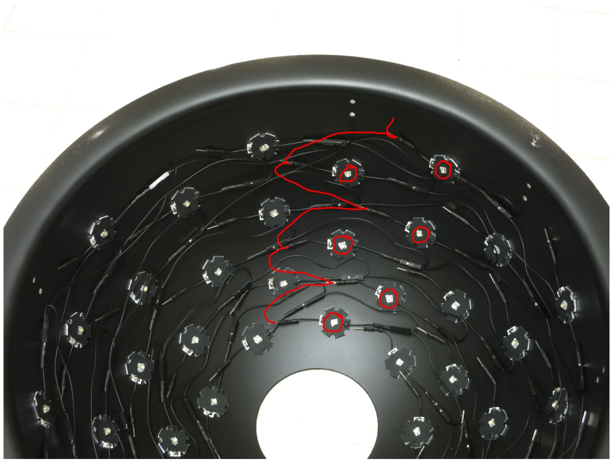
Don’t worry too much at this stage if wires are blocking LEDs – wait until all the wiring is done before bending wires out of the way.

Repeat this process for all the rows in your dome, 6 in this case:

Now repeat the process for the columns, using the wiring pattern in the picture below for every LED column in the matrix:

The bad news is that every wire between adjacent LEDs in the column will probably have to be a different length, since the distances are shorter near the top and longer near the bottom. The LED distance + 1.5” rule is again a good starting guess, possibly even a bit longer than it needs to be. The good news is that if you keep track of the correct lengths for the first column, you can just repeat the pattern for every column, since those distances should be the same for every column. The last lead in the connector wire, the one closest to the bottom of the dome, should be about 1.5” in length.

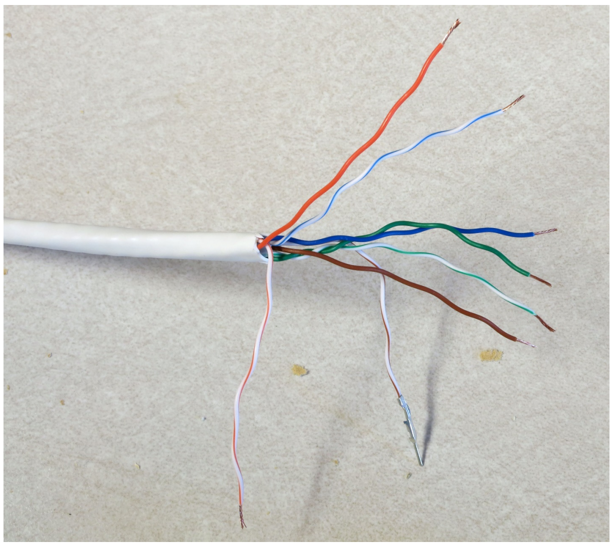


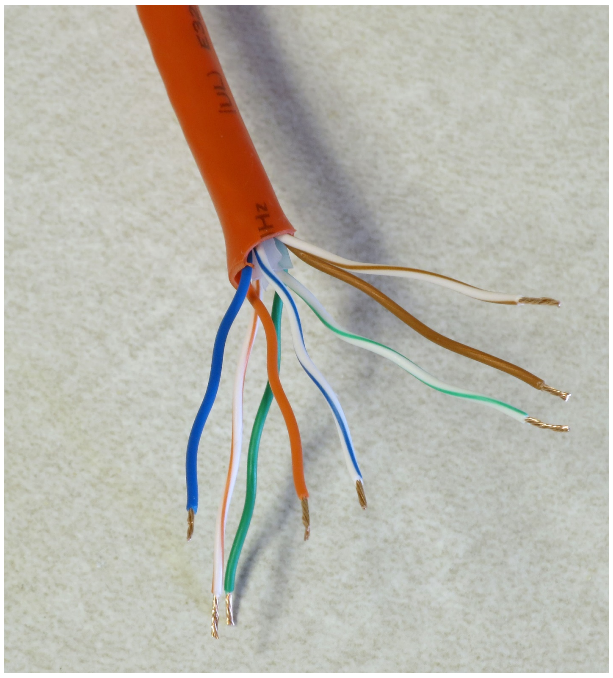


Now that all the wires are done, bend down any wire slack reasonably flush with the interior surface of the dome (doesn't have to be touching), while not blocking any of the LEDs. The picture above shows a reasonably good example of this. When you do the bending, be careful not to bend any of the wires where they're soldered to the LED; bend it too far, or too many times, and it may break. For all except the bottom row, try to bend the wires below the LED row, closer to the bottom of the dome; for the bottom row, bend them to be above that row. Doesn't have to look pretty, since the dome interior will be unseen most of the time. It's also OK to have the wire touch or go across an LED star, as long as it doesn't block the actual LED in the middle.

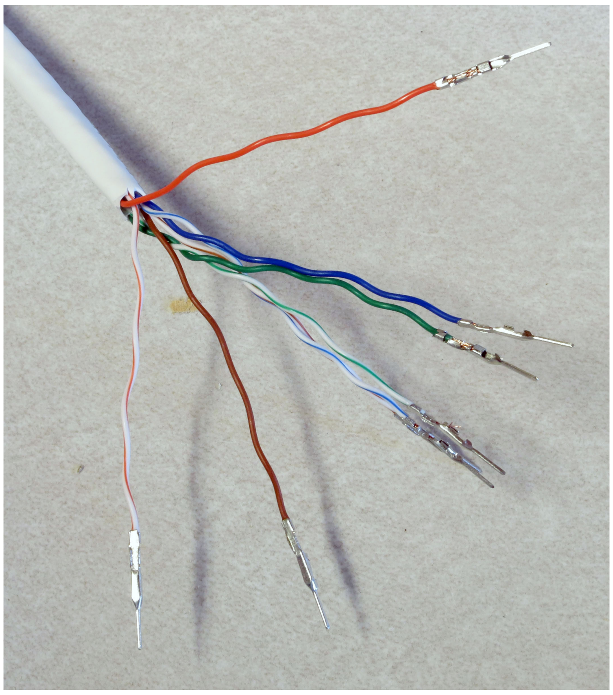
Now that the LED matrix is fully wired, the last step is to make the connection between the matrix rows/columns and the power cables. Power is supplied using Ethernet cables, the same ones you chopped one short end off of in a previous step to make a testing cable. Grab the remaining cables, and trim them both to be the same length. The recommended original length was 7 ft., and 5-6 ft. is a reasonable trimmed length (longer for big domes, shorter for smaller ones). If you start with a 5 ft. cable, 4-4.5 ft. is OK.

You should have one red cable, which will supply positive voltage to the columns, and one cable of some other color, which will connect the rows to ground. In this case, I have a white cable that will do the ground connections (to color-coordinate with the white dome). Using the same method as in a previous step, cut off about 2" of exterior insulation from both cables, trim off any central plastic rib, unwind the paired wires, and trim off about 3 mm of insulation from the end of each wire.

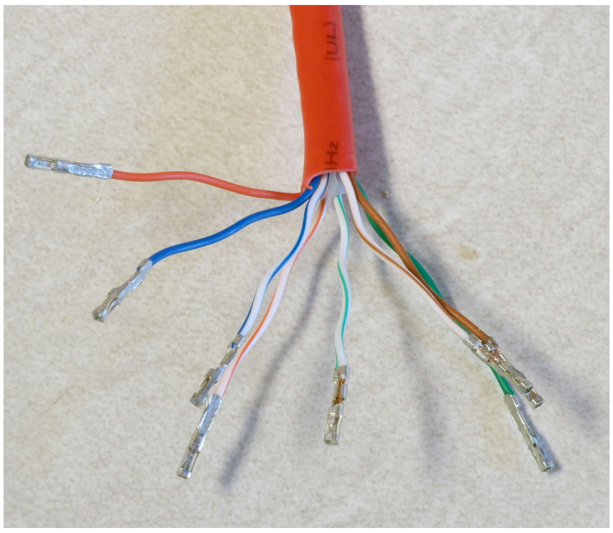




Crimp male Dupont pins onto the ground (rows) cable wires:



Crimp female Dupont pins onto the MOSFET (columns) cable wires:

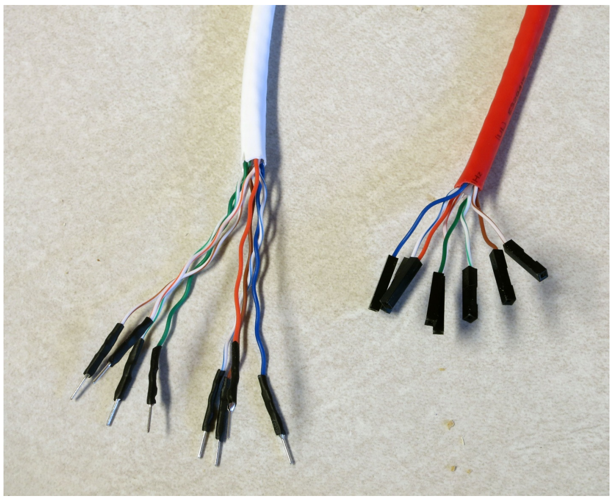


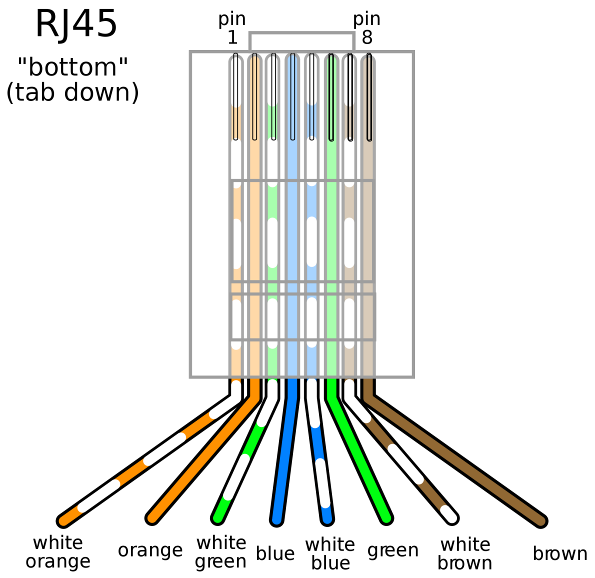
As with earlier similar steps, put a bit of solder on the crimped end to hold the wire firmly in place and ensure good electrical connectivity. Then put some heat shrink tubing on the male Dupont pin connectors to insulate them, and slide the female connectors into plastic housings to insulate them. With the latter, you may have to use a pair of needlenose pliers to get them to slide all the way into the housings. You should wind up with the ends of the Ethernet cables looking like this:

You'll notice that the wires on the red cable are shorter than those on the white cable, even though I specified that you strip off about the same length of external insulation. That's because I decided later on that the original length of 1" on the red cable was too short, and stripped off more insulation to make them 2" long.

I've also made a very subtle mistake here. Each of the wires coming off the white cable attaches to one of the rows in the dome. But there are 8 wires here, while there are only 6 rows in the dome – two of the wires are unneeded. Looking at the Ethernet cable wire chart:

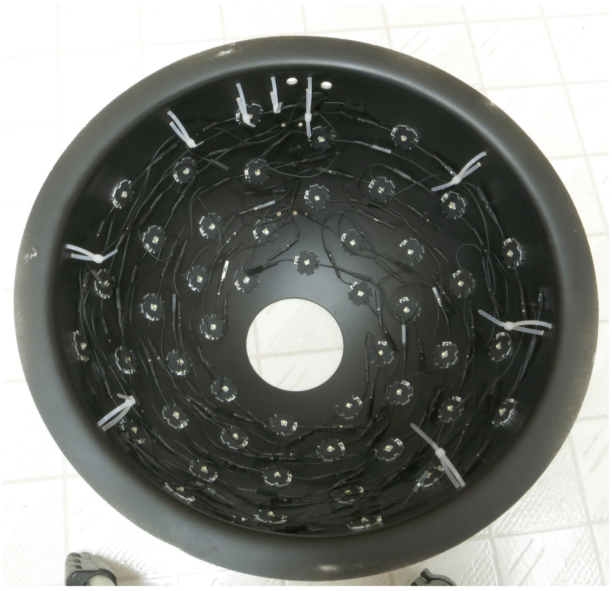
For the dome I'm building here, I don't need the cables for pins 7 and 8 on the white cable, since there are no rows 7 and 8. Those correspond to the wires with white/brown and brown colors, so I cut those off. Obviously, if you



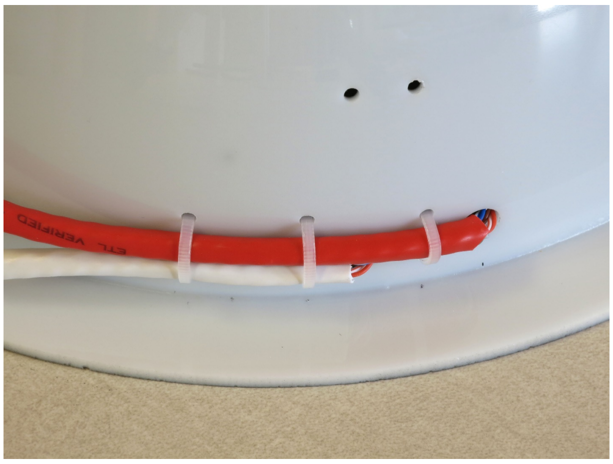

RJ45
"bottom"
(tab down)
pin 1
pin 8
white orange
orange
white green
blue
white blue
green
white brown
brown

build a dome with the maximum allowed 8 rows, you would leave those wires in place. There are 8 columns of LEDs in the matrix, so I leave the wires on the red cable untouched; if there were fewer than 8 columns, I could have trimmed off some of the higher-number pins as well.

Next, install zip ties in the hole pairs around the rim; the ratchet part of the zip tie should be on the inside. If you're using additional zip ties up the side, install them there as well; as I mentioned in a previous step, I drilled the holes for this dome but decided that the zip ties weren't necessary for such a small dome. Don't tighten them up yet, leave them mostly "unzipped" for now.



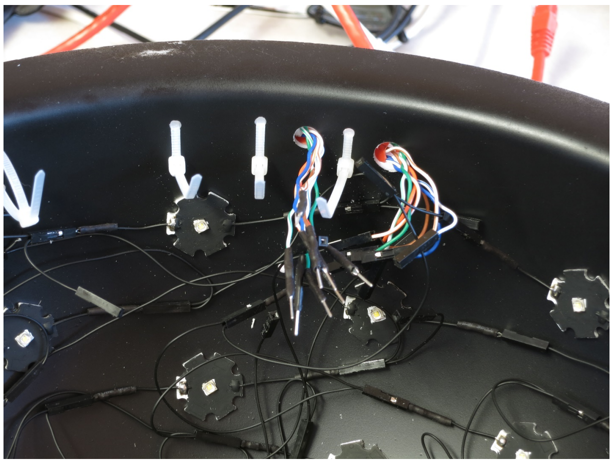
Feed the Ethernet cables as shown in the photo below through the zip ties near the large holes, feeding the wires from the cables through the holes into the inside of the dome. Make sure the red cable feeds into the indicated hole, with the ground cable (white in this case) going into the other hole. When the cables/wires are in place, tighten the zip ties as tight as possible – use pliers if necessary to make the ties tight enough to hold the cables firmly in place. Trim off the excess for these specific zip ties on the inside after you're done, to get them out of the way.

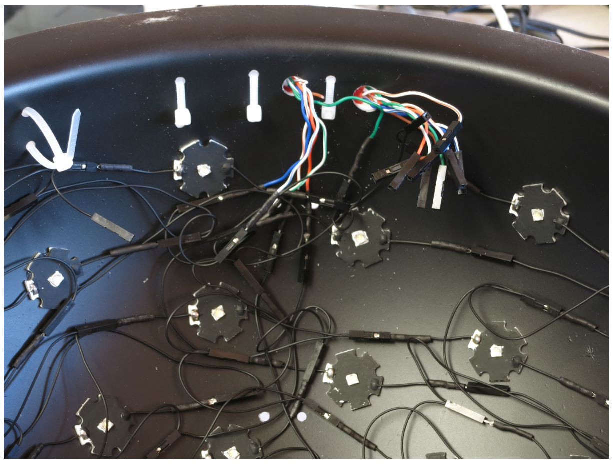


Here's how those cable wires look on the inside, before they're connected to anything:

The wires from the ground cable (the white one) have male pins, which will be plugged into the female pins coming off the ends of the rows. You'll want to follow the Ethernet cable chart above to figure out which color wire plugs into which row.

For example, wire 1 is white/orange, so you'll plug that pin into the female connector for row 1, wire 2 (orange) to row 3, etc. until all the rows are connected. Try running these connection wires underneath other wires in the dome as best as you can. Final result should look something like this:

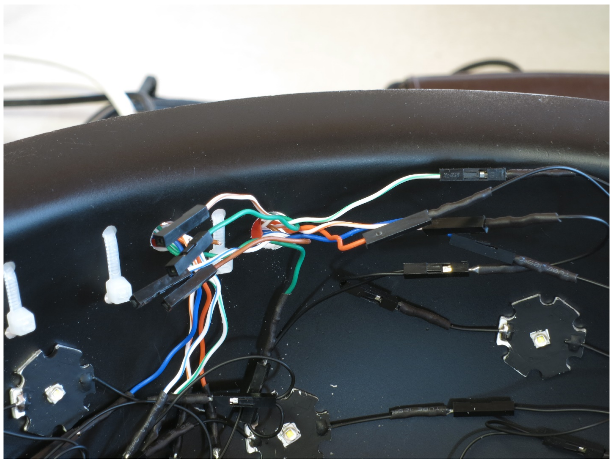




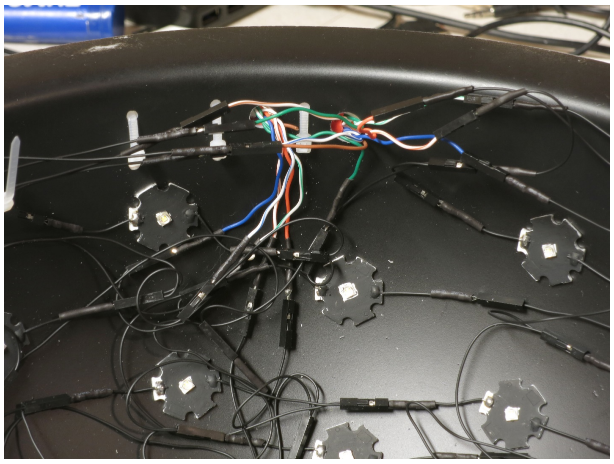
Looks messy, but this won't be in view during regular operation.

Next come the connections to the positive voltage cable (should be the red one, always). There are female connectors on this cable, and female connectors at the base of the wiring of every column. So you'll need to measure and cut 24 AWG Kynar wire lengths that will be long enough to run from the base column connector to the corresponding wire from the red cable.

When measuring the wires, measure along the surface of the dome, just above the base. Crimp male Dupont pins on both ends, solder and heat shrink tube the connectors on these wires, and then connect them to the corresponding female connectors; run them through the zip tie wire guides along the base of the dome. First, do the first four connections running in order, white/orange to the nearest column on the right, orange to the second, white green to the third, blue to the fourth:



For the other four, work in reverse order in the other direction. In other words, brown (pin 8) will connect to the closest LED column on the other side, then white/brown (pin 7) to the next closest, and so on; as before, run the wires through the zip tie guides. You could run the wire for pin 8 all the way around the edge of the dome, but doing it this way reduces the amount of wire you need. When done, the full set of positive voltage connections to LED columns should look like this:



Time to test the wiring. I've written two programs that will test all the LEDs; you should find these in the Files section :

- the first one, Dazzler, lights up the LEDs at random
- the second one, Serial_Test, lights up the LEDS one at a time, first by row, second by column.

Upload these individually to the Arduino controller using the Arduino IDE (described in an earlier step). The default setting for both these programs is 8 Rows, 8 Columns; if your dome has fewer Rows or Columns, modify the appropriate constants in the program. So for my test, I changed the Rows constant from 8 to 6 (no decimal point). If you've finished the control box while waiting for the LED adhesive to dry, use that; otherwise, use the same wiring configuration in the system test done earlier.

Plug the dome's red cable into the MOSFET board Ethernet connection, plug the ground cable into the CAT4101 board Ethernet connection, then plug the 9V power supply in – if you only have the USB cable plugged in, the lights will not go on. Dazzler looks cooler, and spots major problems faster. Serial_Test runs slower (set the time in milliseconds with the LED_Time constant; default is 200 milliseconds = 0.2 seconds), which lets you pin down a specific LED that might be a problem.

Hopefully, all the LEDs will light up, and everything will work fine. But if it doesn't, don't worry – didn't work for me the first time, either. If a row doesn't light up, double check the matching wire connection on the ground cable; for a dark column, check the connection on the red cable. For an isolated LED, carefully check the connections to both pins of the LED. In my case, one of the columns didn't light up the first time I tried it; after unplugging and re-plugging in the appropriate column wires, it worked perfectly.

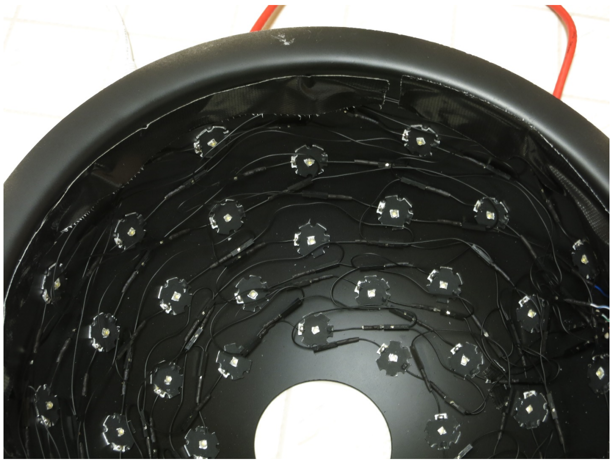
When all the LEDs are working, the wiring is done! Tighten up all the zip ties, and trim off the excess on the inside. If the dome will be sitting permanently in one location, you can leave the interior as is. Since this dome is designed to be portable, I cover the wires on the bottom of the interior with black Gorilla tape to keep them more secure (less prone to being accidentally yanked):

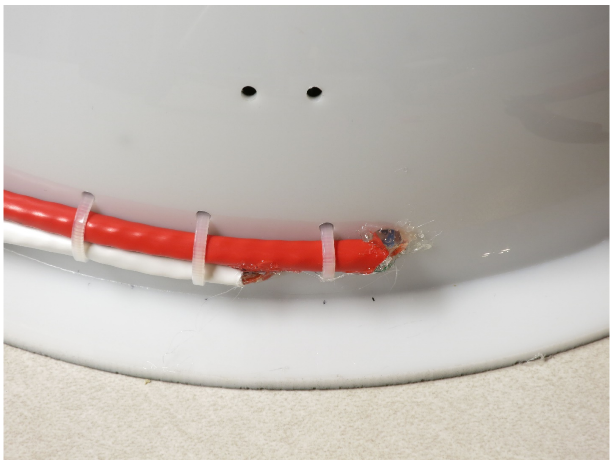
One problem with Gorilla tape is that it's glossy, so it will reflect light from the LEDs to places you may not want it. To fix this, I painted the Gorilla tape with flat black paint to dull the finish. I like to seal up the two holes where the wires/cables go into the dome, just to keep them from rubbing on the edges. Use silicone adhesive, or in my case, hot glue:

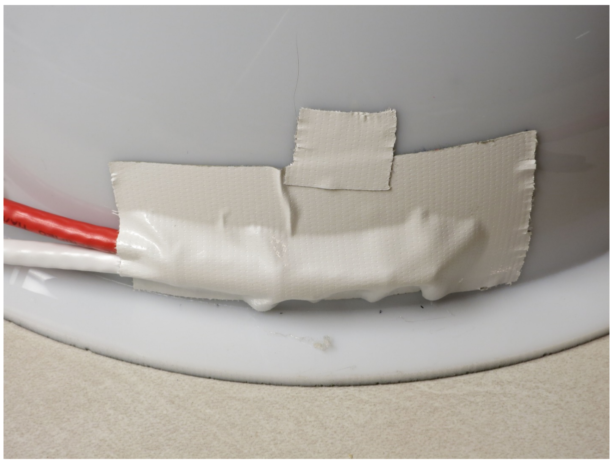
Finally, I put some Gorilla tape over the cable zip ties and holes, to make it look cleaner:

I also have an extra piece of tape covering up the holes I drilled but didn't use. Feel free to omit this tape if you want, or use some other method to cover this area up.

That's it – dome is done! Handle with care.





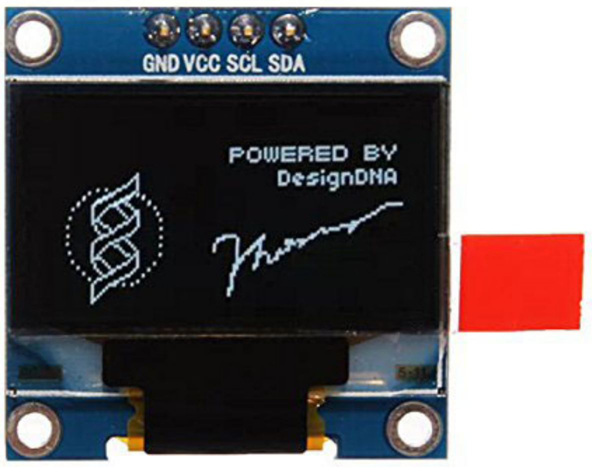


CHAPTER 11

# Assembling the control box

The big addition I made to the RTIMage controller in my latest generation is a small OLED display (0.96"width, 128 x 64 pixels, maximum 8 lines x 21 characters per line). While it's entirely optional, I highly recommend including it in your build. It gives instant feedback on the system parameters, and lets you know the status of progress during an RTI photography run. But there's a bit of prep you need to do in order to properly install and use it.

First off, while many listings for this product show that it already has pins soldered at the top:

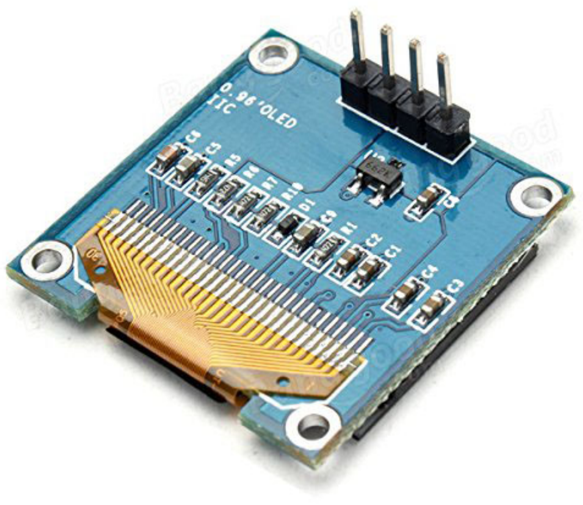


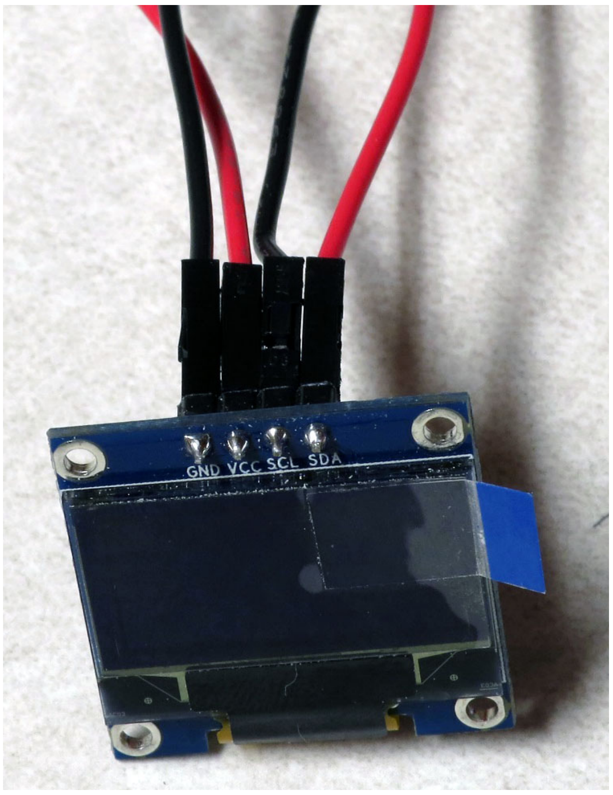
That isn't necessarily the case for the shipped product you might receive. If not, just break off 4 header pins from the 40-pin header strip on the parts list (should already be shrunk to a much smaller size because of previous removals), and solder it in place as seen above.

You'll need some way to connect the pins to the Arduino. Cut 4 pieces of AWG 22 wire about 7" or so in length (two red, two black is recommended). Strip about 3 mm of insulation off one end of each wire, then crimp and solder a female Dupont pin to that end (insulation won't fit, but the wire is so fat that you should get a good crimp regardless). Stick the female Dupont pin into a plastic housing (all the way in). Strip about 5mm off the other end. Fit the black wires onto the header for the pins marked GND (ground) and SCL; fit the red wires onto VCC (voltage) and SDA.

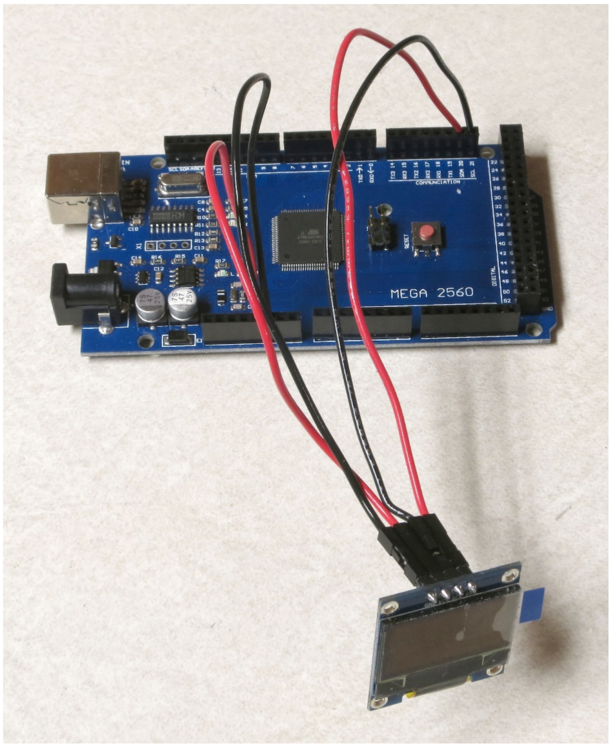
Now insert the wires into the following headers on an Arduino Mega:

- GND goes to any Arduino GND


0.96'OLED
IIC


GND VCC SCL SDA

- Vcc goes to any Arduino 5V
- SDA goes to Arduino SDA
- SCL goes to Arduino SCL



This OLED panel communicates with the Arduino using the I2C/IIC serial bus (Inter-Integrated Circuit). Every chip that uses this protocol is supposed to have a special address, allowing multiple chips with different addresses to use the same bus. The display I bought had an I2C address on the back of the PCB; unfortunately, that I2C address was wrong. To find the correct I2C address for any device attached to an Arduino, you can use a program called I2C scanner. It's at this page on the Arduino site, but a Google search will bring up many other locations.

Plug your Arduino into your PC with a USB cable, and upload the I2C scanner program using the Arduino IDE (see the earlier Arduino setup instructions for info on how to install and configure the Arduino IDE). Apparently nothing happens. Not true – the program is scanning every possible I2C address to look for devices, and if it finds one, it sends that information back through the USB cable. To see that info, go to the Tools menu in the Arduino IDE, and select Serial Monitor. A window will pop up, and eventually you will see something like this:

```
I2C Scanner
Scanning...
I2C device found at address 0x3C  !
done

Scanning...
I2C device found at address 0x3C  !
done

Scanning...
I2C device found at address 0x3C  !
done
```

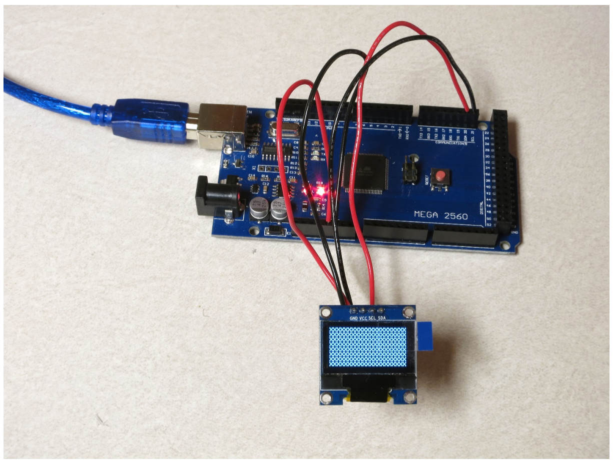
This will go on forever if you let it. So the I2C address of my display was 0x3C.

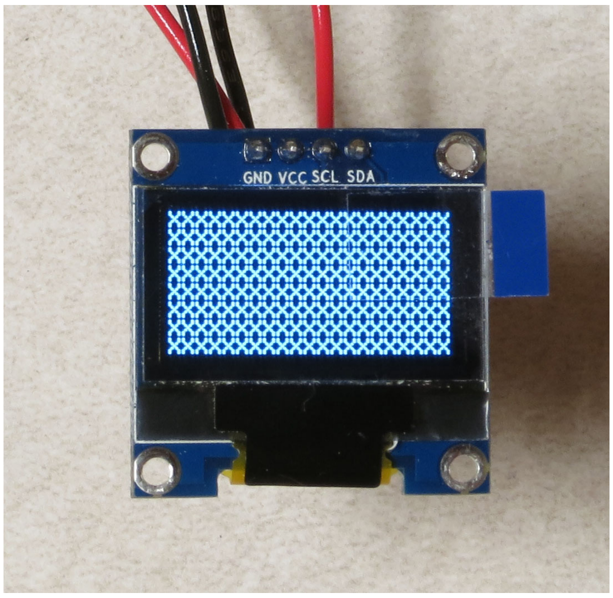
Next, you'll need to test your OLED display to make sure it's working, and to find out the limits of the area where something is visible in the display. You'll need the latter because the display area is often not in the middle of the board, but displaced slightly up towards the top (as in the picture above). Download the zip file RTIMage_OLED_Tester from the files section, then copy the folder into your Arduino documents folder. Open it in the Arduino IDE, and look for the following line:

> display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C address

That 0x3C is the I2C address for my display. If your display has a different I2C address as revealed by I2C scanner, you need to replace 0x3C with that new address.

Upload the program to your Arduino. The first thing you'll see is an Adafruit splash screen; this program is a modification of an Adafruit program distributed with the Adafruit SSD1306 library, and the licensing of this program requires that any modified version show this splash screen. After the splash screen disappears, the screen will fill up with Xes, like this:

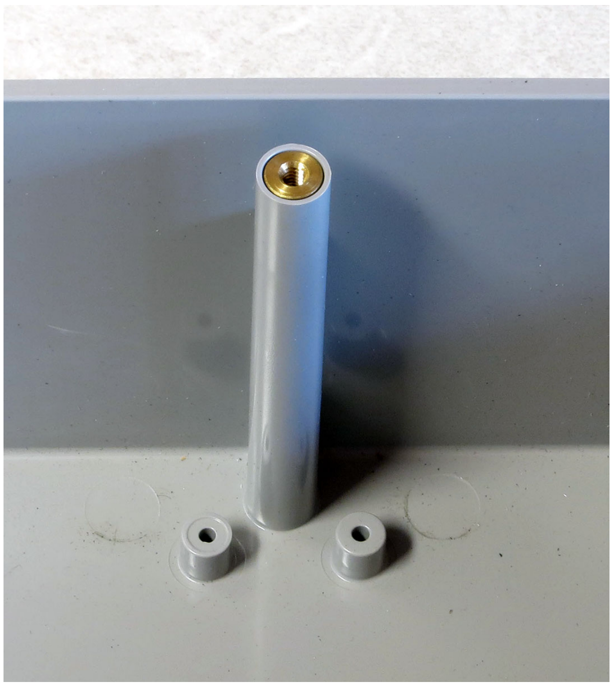




You will need to somehow record the extents of the display area relative to the total board dimensions, and the holes. You'll need this information when installing the display in the control box, as it will determine the position of the box cutout for the display, and the positions of the mounting holes. Once done, remove the wires (but save them for the final installation), and set the display aside for now.

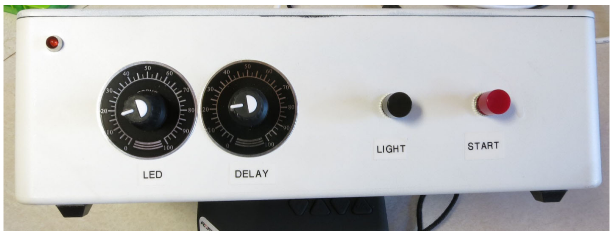
The following steps lay out the configuration and assembly of the RTIMage control/power box. The exact layout and assembly here is only valid for the enclosure box specified on the components list, the Polycase DC-96P (http://www.polycase.com/dc-96p or equivalent). If you use an alternate enclosure, you will almost certainly have to modify the layout to some degree. However, you will need to install the same components as in this build, so reading these sections is a guide for how you should plan and execute your specific assembly procedure.

One thing you have to watch out for with the Polycase is that there are multiple internal struts that support the threaded inserts for the screws that hold the lid in place; you'll need to avoid drilling/cutting through these, or positioning items too close to them:



Start by picking one of the long sides of the box, and designating that as the front. To help align the holes and cutouts on the front, I draw a pencil line right through the middle of the box; make sure you have the lid on when you measure the distances for this:

For my previous builds, there were 5 objects to be installed on the front: two 10K potentiometers, two push-button switches, and one LED power indicator. Here's the front panel of one of my previous builds:

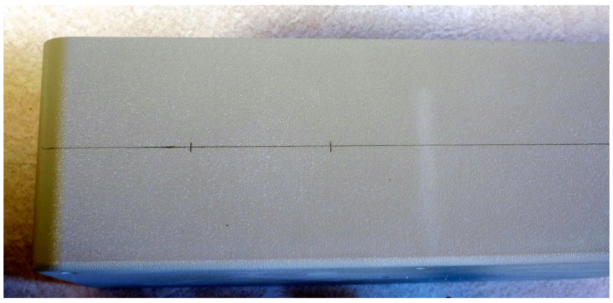


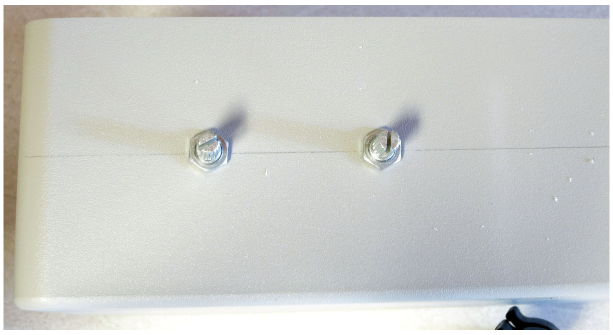
For this build, I'll be including one more component, the OLED display, and modifying the front layout to allow space for this. If you choose not to install the OLED, you can use the layout above.

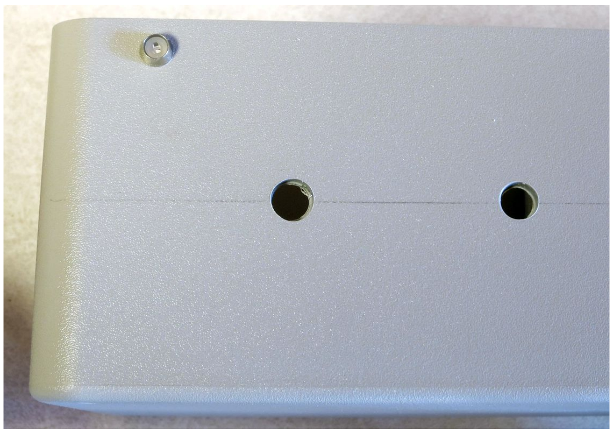
First, mark two drill holes on the left-hand side for the two 10K potentiometers:

Measure the size of the potentiometer knobs, then drill holes at the mark just large enough for the potentiometers to fit through. Although this plastic is soft enough for regular drill bits, I recommend using a narrow step drill for all large drill holes coming up – it makes a cleaner cut through the plastic, and if you drill the hole too small, you don't have to change the bit to drill a larger one. After drilling the holes, check to make sure the potentiometers fit:

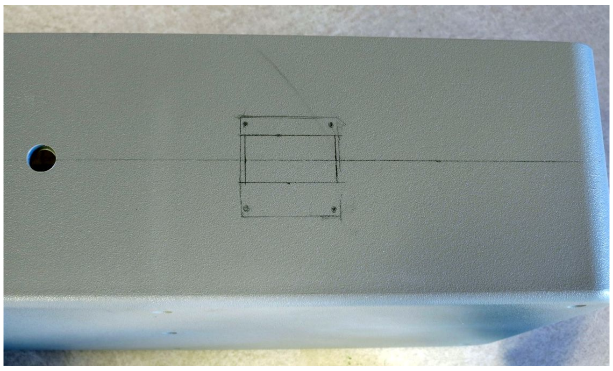
Next, drill a hole in the upper left hand corner for the 3mm LED holder, then install it to check for the right size. Make sure you're not too close to the top, otherwise the lid may not fit on; I put my hole in a just barely acceptable position:







Moving to the right side, if you're installing the OLED display, you'll need to mark the location of both the cutout for the main display (which you determined in an earlier step), and the screw holes that will hold the display in place. My approach was to place the OLED display in what I thought was the best position, then trace around the outside of the display board (also marking the positions of the screw holes); I then marked the area I needed to cut out of the box face to have the display part of the board visible:

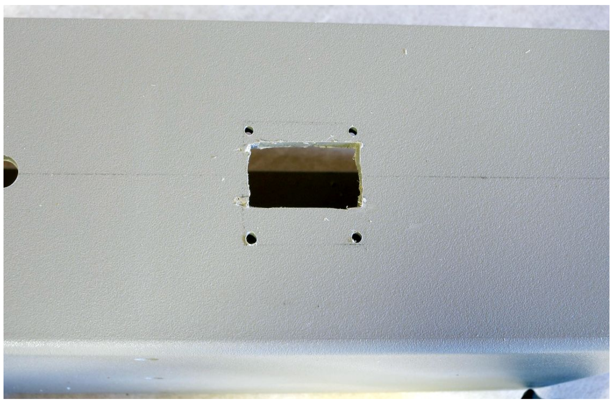


Now, a brief diversion into cutting rectangular holes into the box, since you'll have at least three of these (for two Ethernet panel jacks and one USB panel jack), four if you install the OLED display. If you Google "cut square holes plastic", you'll come up with lots of suggested techniques. Here are my suggestions:

- Go through the rest of these instructions, and mark the positions of the cutouts. Then find a friendly woodworker or machinist, and ask them to make the cutouts. Should take them less than five minutes, and the cutouts will look great.
- Drill as many and as large holes as possible within the limits of the cutout, then use a sharp utility knife (boxcutter) to trim out the remaining plastic, and square off the edges.
- I used a variant of the above technique, but used a Dremel rotary tool with a 561 cutter to remove the bulk of the plastic and trim towards the edges. USE SLOW SPEEDS! I then used a utility knife for the final finishing work.
- If the plastic is less than 1/16" thickness, you can use a "nibbler" to cut out most of the plastic after drilling a starting hole. However, the plastic in the Polycase box is too thick for a nibbler (I know this for sure – bought a nibbler, then found out it wouldn't work on that box).

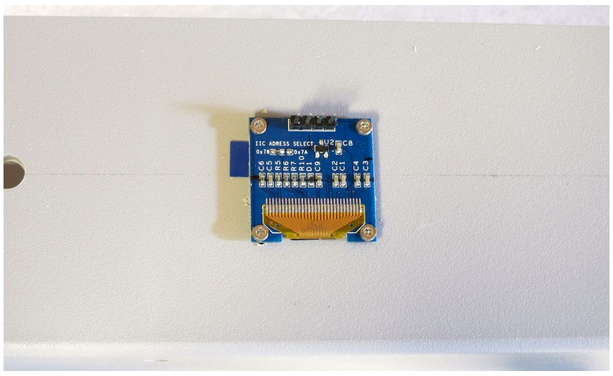
If you're doing this yourself, but have never done it before, one suggestion would be to do the other three cutouts first for practice. They're on the sides and back, so if they look bad when you first do them, they won't be as noticeable as the one on the front.

I went ahead and did the front cutout first, along with drilling holes for screws:

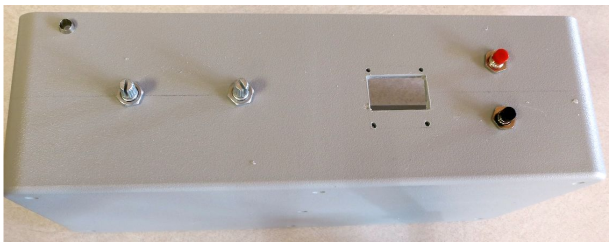


Doesn't look great in this picture, but I went back and cleaned it up a bit with a utility knife and file, so the final version looks better.

To make sure the screw holes were drilled in the right place, put the display on the box over the cutout, and insert the mounting screws to make sure they fit and are in the right place. If they don't fit exactly, drilling the holes one size larger may fix the problem. I used #1 screws/nuts/washers to hold the display in place:

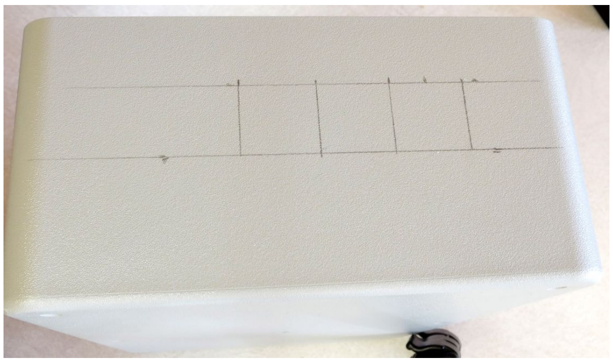


Next, drill holes for the two front panel pushbuttons, and check to make sure they fit:



That's it for the front panel; remove all the installed parts for now.

Next is the right side panel; you'll need to make two cutouts for the Ethernet panel jacks. Your natural inclination may be to put them near the bottom, but if you do, the CAT4101 driver board will block installation of the panel jacks. Here's where I marked the positions for mine:



The top edge of the cutout is 2 cm below the top of the box, the right edge of the right cutout is 3.5 cm from the back edge of the box, the cutouts are 2 cm x 2cm in size, and are spaced 2 cm away from each other. Cutting them out:

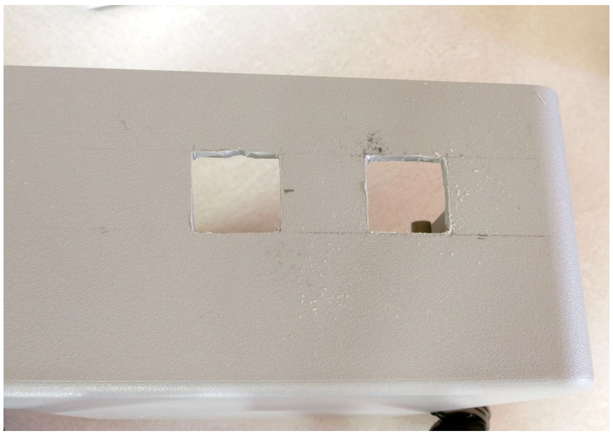
Grab one of the Ethernet panel jack cables, and measure the distance between the two screw holes. Drill two holes that distance apart evenly spaced on both sides of the cutout horizontally, and in the middle of the cutout vertically:

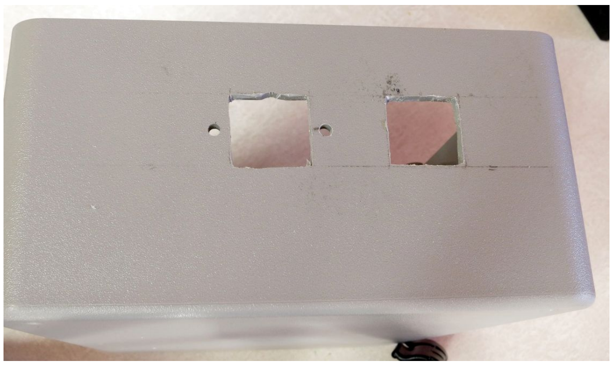
Install one of the Ethernet panel jacks to make sure it will fit:

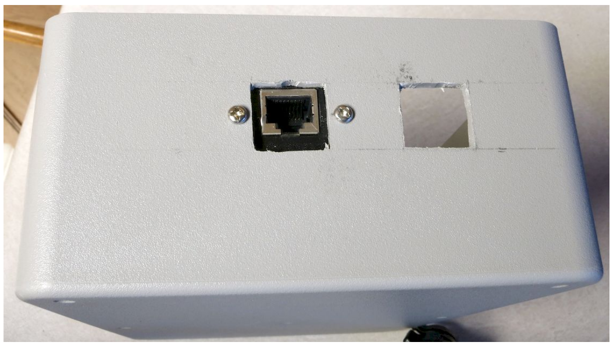
Drill holes in a similar position on the other cutout, and do the same check to make sure the panel jack will fit:

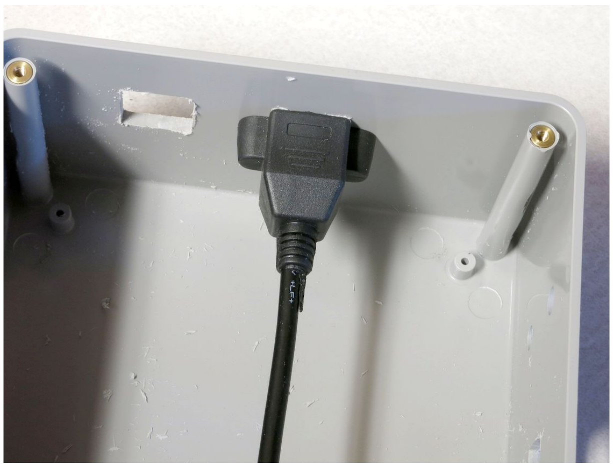
You're done with the right side of the box, now move on to the rear. Drill three holes on the upper right side for the three toggle switches, then install them to check the fit. The pencil line was just to make sure they were all neatly on the same horizontal line, spaced about the same distance apart:

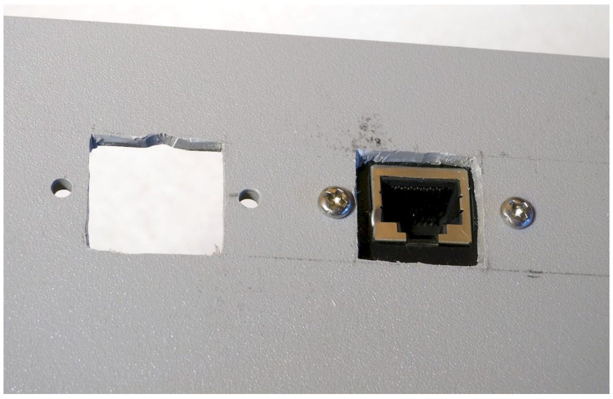
On the upper left side of the rear, mark a cutout for the USB shutter control cable about 1.8 cm wide and 1 cm high. You should position it about 1.5 cm below the top edge, to make sure it clears the electronics when the system is full assembled.

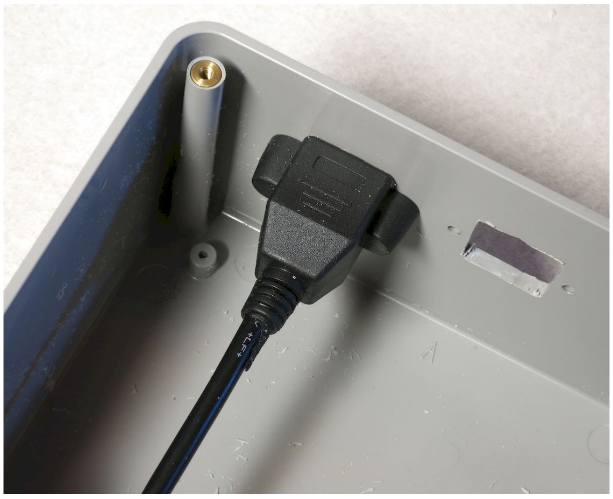


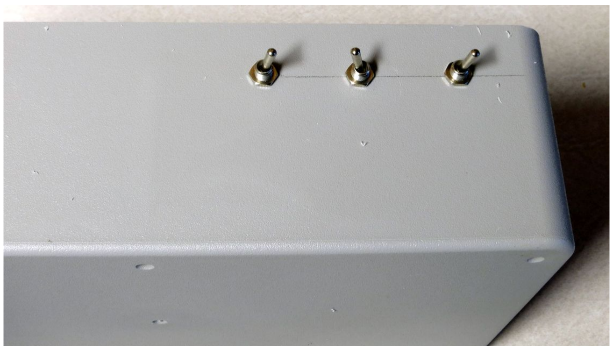


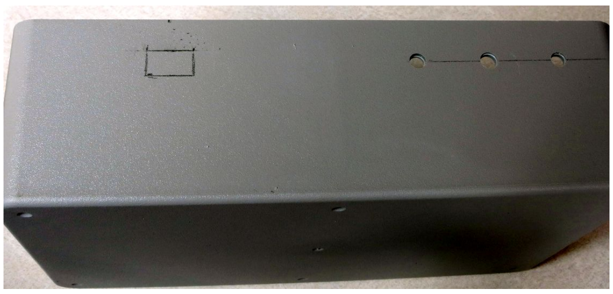




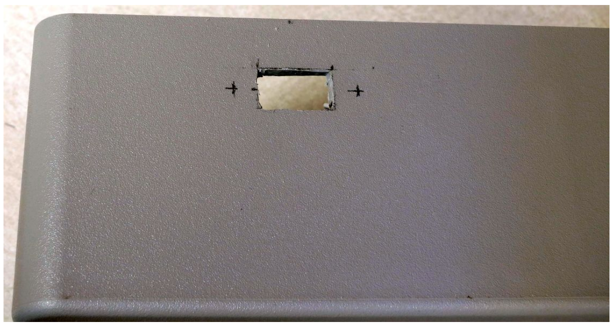


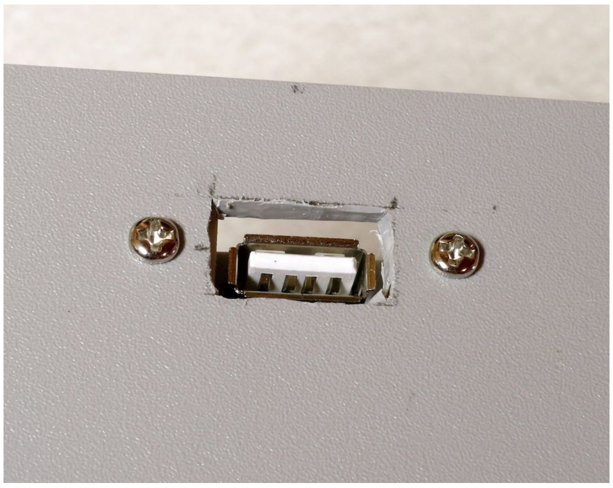


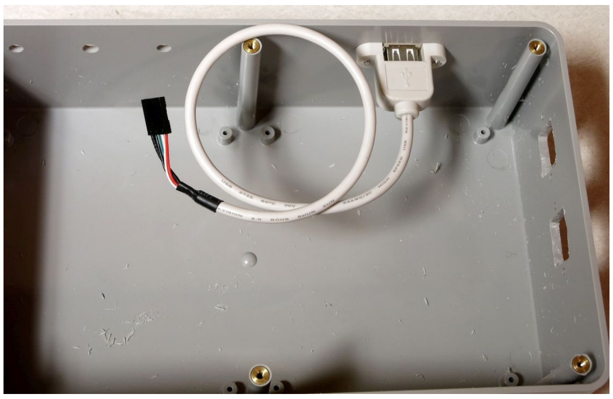




Cut out the plastic, drill two holes for the USB panel jack screws, then attach the USB panel jack to make sure it fits:







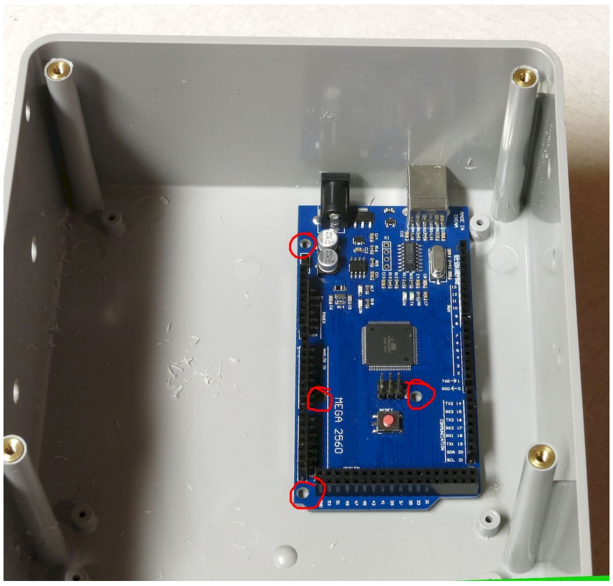
To make sure the cutout is big enough, and the panel jack is positioned correctly, grab a USB cable and check to make sure you can insert it fully into the jack.

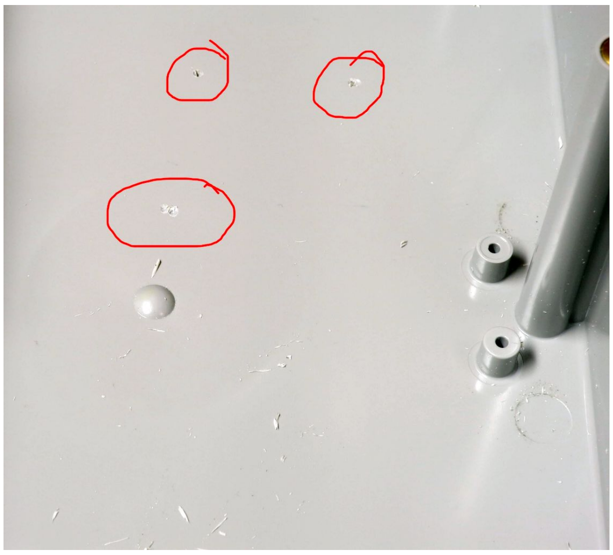
Next, take your Arduino Mega, and slide it into the rear left corner so that the USB jack is flush and square against the left side of the box, and the top edge of the Arduino is flush against one of the short PCB screw posts at the bottom. All the edges of the Arduino should be parallel to the nearest edges of the box:

Grab a drill bit that will fit into the holes in the Arduino PCB board, push it through the 4 holes indicated in the picture below (circled in red), and twist it to make a mark that will indicate where to drill a hole later on:

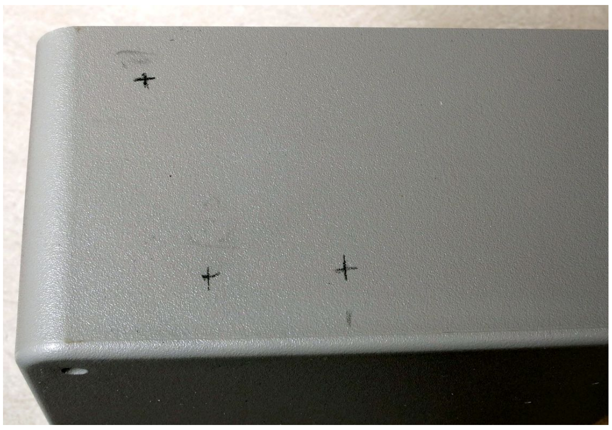
Ignore the two additional holes at the top of the Arduino– they don't have enough space for a screw or nut to be useful. The picture below shows three of the four drill marks, top one got cut out of the picture by mistake:

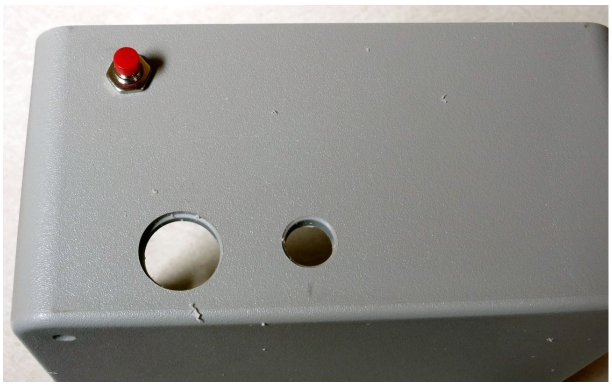
Next, go to the left side panel (the only one left that shouldn't have any holes in it). You will be drilling three holes. The first, in the upper left, is for the reset button, and exact position isn't critical – just make sure the button will fit in the hole, and it won't keep the lid from fitting.





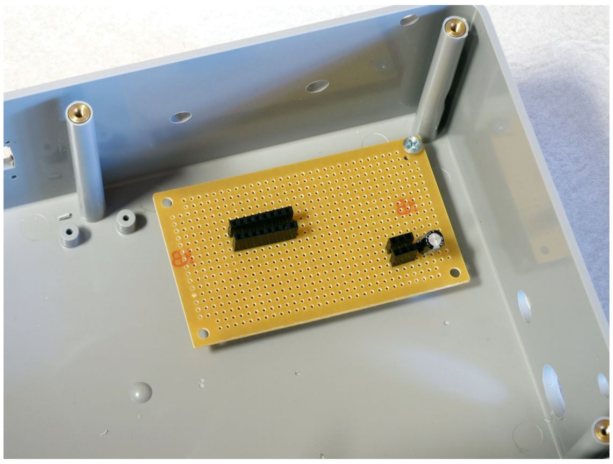
The other two holes are for the Arduino USB jack and Arduino power jack. The one on the left is the USB jack, and the drill mark is 3.75 cm from the back edge (the flat part, not the curved part), and 1.6 cm from the bottom. Drill a hole about ¾” in diameter at this location using a step bit. The one on the right is for the Arduino power jack; make a drill mark 6.8 cm from the back edge and 1.7 cm from the bottom, then drill a hole ½” in diameter.





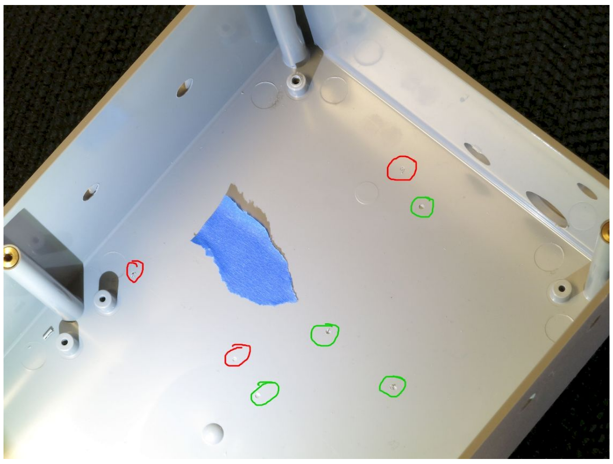
Note: If you drill the two Arduino holes and then stick the Arduino back into the box, it may look like the holes are too high. Don’t worry – you’ll be installing the Arduino on top of spacers that will raise it to the right height.

One more set of marks to make on the inside bottom of the box. Install the power strip board using one 4-40 sheet metal screw to attach it to the PCB upright in the front left corner of the box; position it as square as possible:

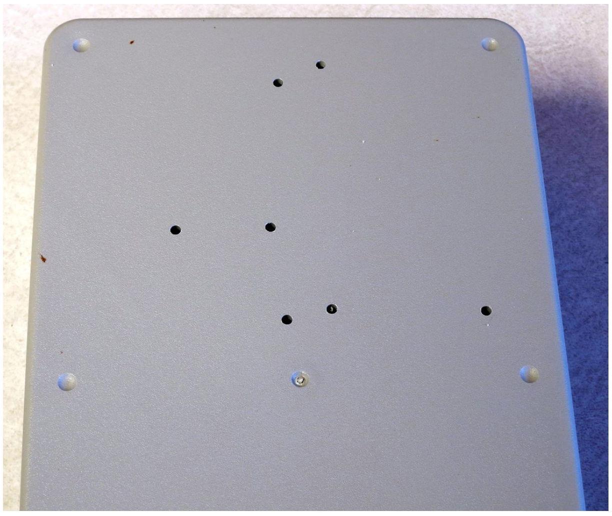


As with the Arduino board earlier, use a drill bit through the open 3 holes of this board to mark a drill position in the bottom of the box directly underneath the board. After removing the board, you should see the following marks, with the power strip board marks circled in red, and the Arduino marks made earlier circled in green:

The blue masking tape? That had me confused, too, until I remembered that the camera had trouble focusing for this picture due to the lack of detail, and adding the masking tape fixed that problem.



For all 7 drill marks above, drill a hole for a 4-40 screw (or its metric equivalent). The bottom of the enclosure will now look like this:



While the system doesn't consume that much power, and shouldn't get too hot, I think ventilation/cooling is never a bad thing for electronics. So I recommend drilling 4 ¼" holes centered in the 4 quadrants of the bottom of the enclosure:

And four ¼" holes at the bottom of the rear of the enclosure:

That's it for cutouts, and almost it for holes – two more to drill, but you'll have to start putting the system together to mark the positions for those.

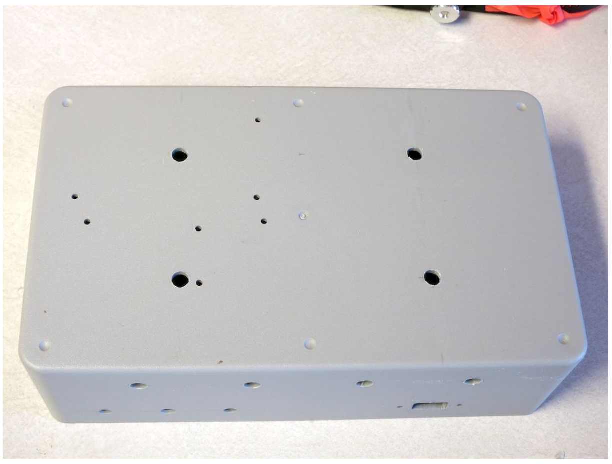
Now that most of the holes (drill/cutouts) have been made in the enclosure, it's time to start installing stuff in there. First, take 4 ¾" 4-40 screws (preferably nylon), and insert them through the holes drilled for the Arduino. They'll want to fall out, so until they're secured, you should tape over them on the bottom of the enclosure to keep them in place.

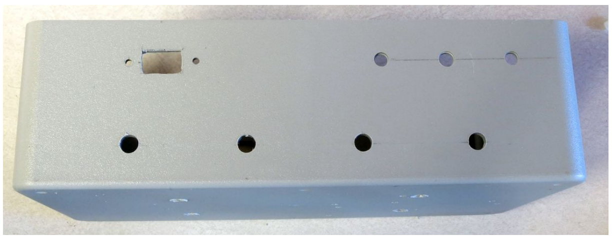
Place a ¼" spacer on each of them.

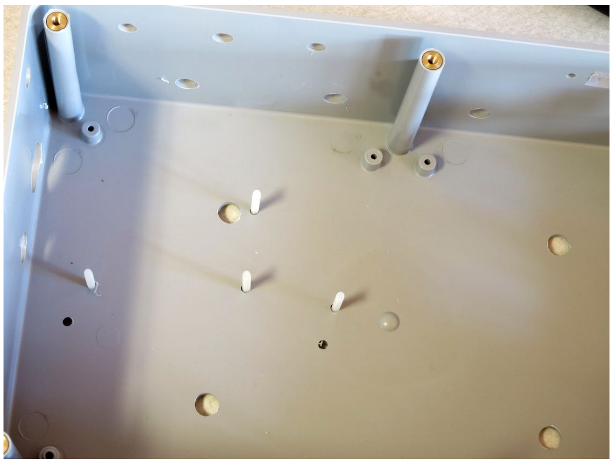
Place the Arduino on top of the screws so that they go through the holes in the Arduino. Put a washer and nut (preferably nylon) on the center 2 screws, and tighten the nuts to hold the Arduino in place:

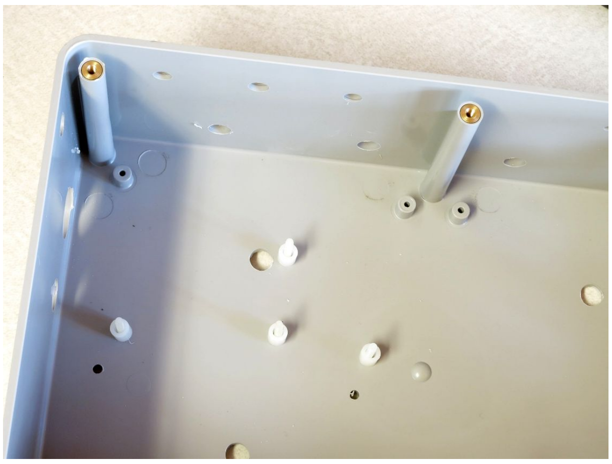
Check to make sure you can plug both the USB cable and the power cable into the Arduino through the holes drilled in the left panel:

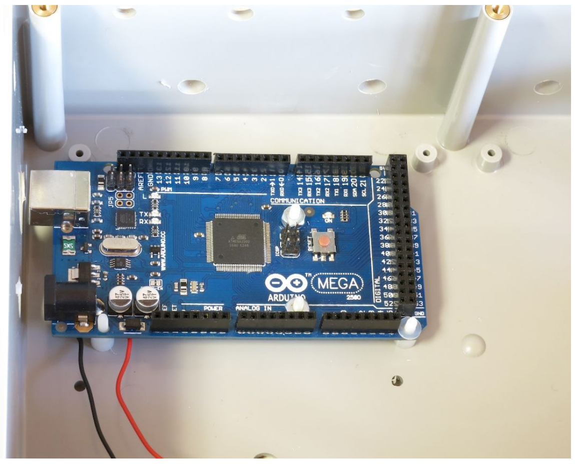
You can see the tape holding the screws in place below. If they don't fit, take the Arduino out and drill the holes larger until they do fit (which I have done on every previous control box I've built; this is the first one where they fit on the first try).

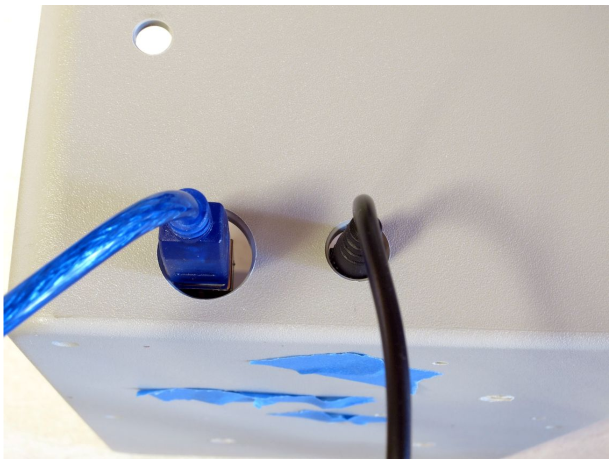




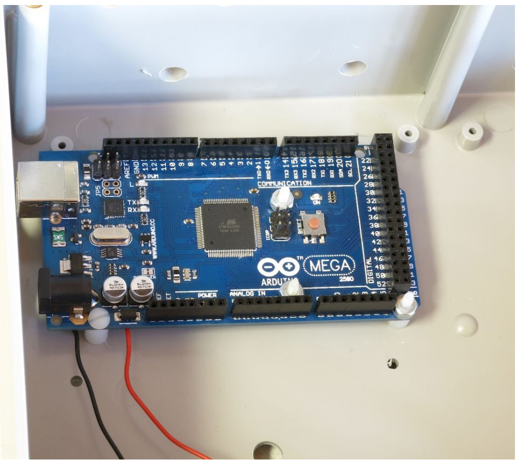






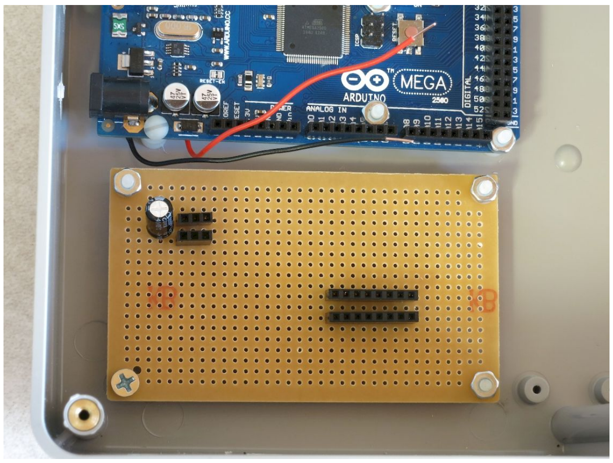


If everything fits, attach nuts to the other two screws. First, the screw on the right - there's just barely enough room for the nut to fit the two adjacent headers, but not enough to actually turn the nut in place. So you'll have to turn the screw from the backside to fully tighten it. The screw on the left is even worse – there's no room for a nut there at all. So you'll need to carefully pull it out, leaving the spacer in position, then insert it the opposite way and attach washer/nut on the outside.



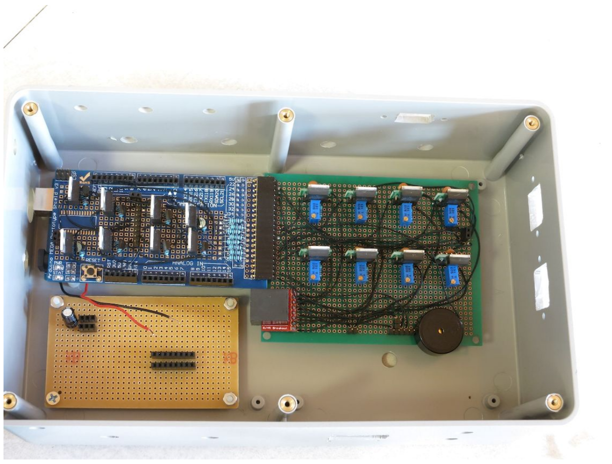
Install the power strip board with one 4-40 metal screw in the PCB standoff in the lower left-hand corner, and 3 4-40 ¾" screws through the holes, with spacers. As with the Arduino screws, you'll need to tape them at the bottom of the enclosure to hold them in place:

Looks like the Arduino board is slightly askew. No worries, as long as you can plug in the USB cable and power cable, it's OK.



Insert the MOSFET driver shield into the Arduino, taking care to make sure the right pins go in the right holes. Apply pressure evenly when inserting, otherwise you can bend some of the pins during insertion. Make sure the shield is fully inserted into the Arduino.

Plug the CAT4101 board into the MOSFET shield:



Holding the CAT4101 board horizontally, take a drill bit and use it to mark drill hole positions through the two holes on the right side of the PCB. Remove the board, then drill two holes for #6 screws at those marks. Insert two 1" #6 screws, taping them to the bottom of the enclosure to hold them in place. Place a ½" spacer and a 1/8" spacer/thick washer on both screws:

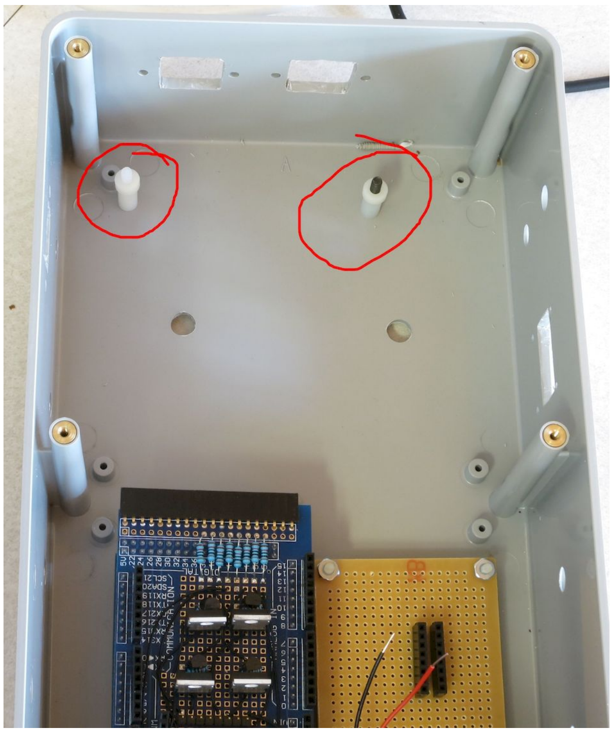
Reinsert the CAT4101 board into the MOSFET shield, inserting the ends of the screws into the holes at the end of the board. Check that the board is close to horizontal; if not, you may need to add/subtract spacers of various thicknesses to make it so. Fasten the board to the screws with washers/nuts.

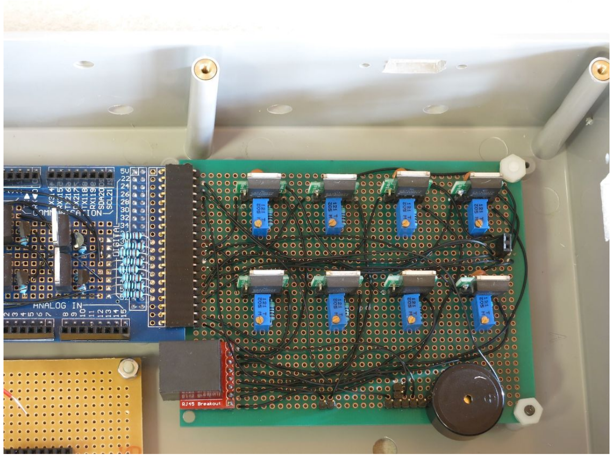
Now is a good time to attach heat sinks. I'm not actually 100% sure you need them on the CAT4101s, and I'm pretty sure you don't need them on the MOSFETs. The former may get a bit warm, but they will work fine up to 150C at which point they shut down automatically. Since each CAT4101 will only be on for a few seconds total during an RTI session, it's unlikely it will get anywhere near that hot.

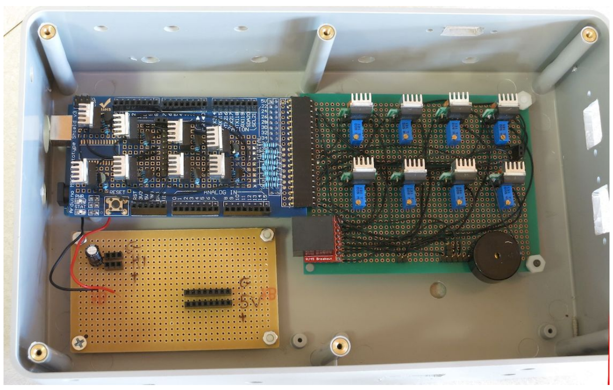
The MOSFETs have even less of a thermal problem – an Rds of around .15 plus a max current of 1 A means they'll have to dissipate about 0.15W of heat max for a few seconds, which shouldn't be an issue.

Me being paranoid, I installed heat sinks on both the CAT4101s (right) and the MOSFETs (left):

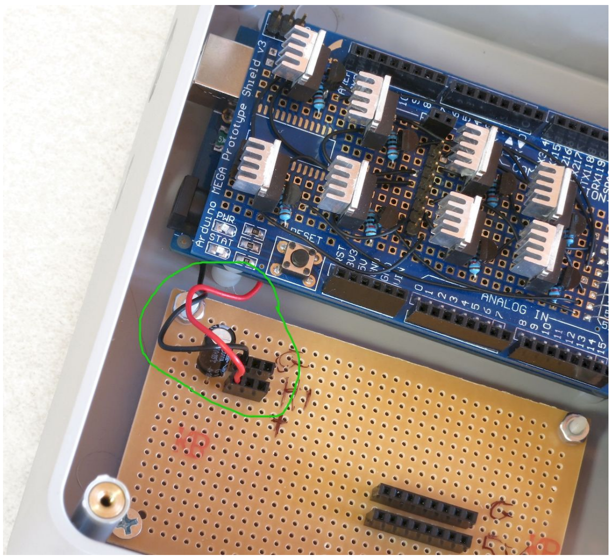
In retrospect, the MOSFET heat sinks were a bad idea. They came close to blocking a couple of key connectors on the MOSFET shield. So I've made an executive decision, determined that you don't need heat sinks on the MOSFETs, and cut the heat sink count on the component list from 16 to 8. Still not convinced you'll need them on the CAT4101s, but there's a better case for them there; with a 12V power input, they may need to handle a few watts of heat for a few seconds.



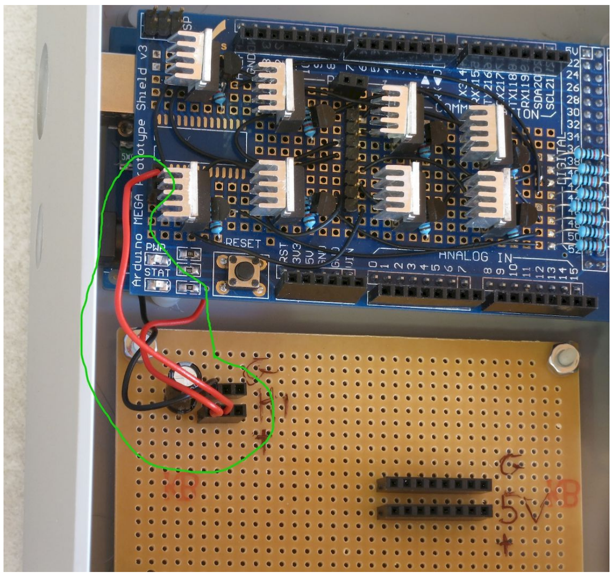




Insert the black and red power wires from the Arduino as shown (circled in green), red (+ voltage) to the bottom strip, black (ground) to the top. You might also want to label the power connectors as shown; upper left is high voltage directly from the power supply, while lower right will get 5V of power and ground directly from the Arduino.



Connect a red jumper wire from the +V header at upper left to the header on the MOSFET driver shield; this will be the main current supply to the LEDs:



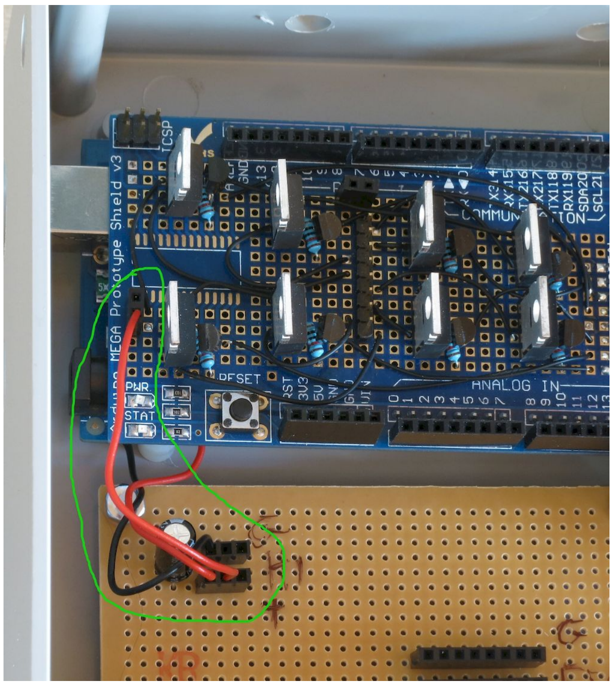
The heat sink blocks a good view of that shield header, but I shot a similar picture before installing the heat sinks:

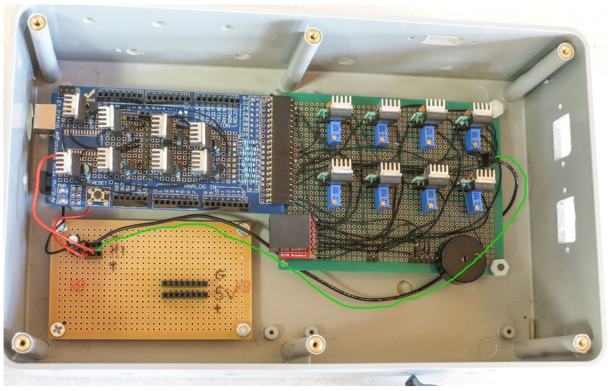
Now run a long black jumper wire from the high voltage ground strip to the double header on the CAT4101 board; this will be the main ground for the LEDs. Try running it underneath the CAT4101 board as shown, to keep it out of the way:

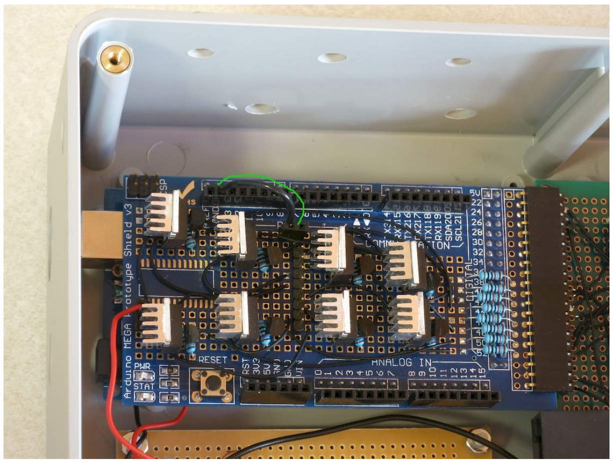
Run a short black wire jumper from the Arduino ground header (upper left on the Arduino board) to the double header nearby on the MOSFET driver shield:

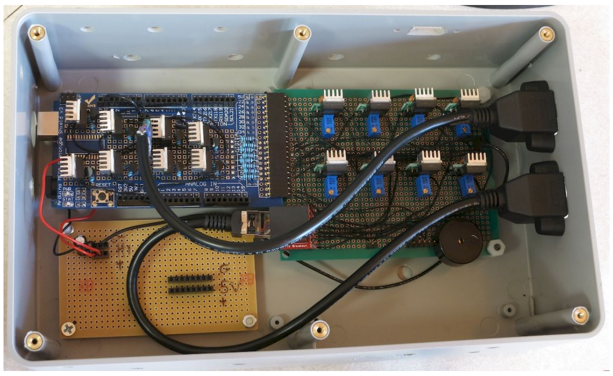
Install the two Ethernet panel jack cables as shown. Make sure you have the connector to the MOSFET board oriented properly, with pin 1 at the top. Put the MOSFET-connected panel jack on the rear side of the right panel, as that's where the red +V cable from the RTI dome will attach (mnemonic: "red to the rear").

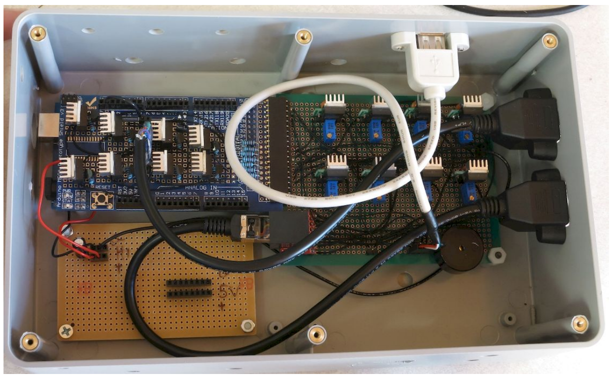
Install the USB panel jack connector. The black ground wires on the 5-pin header should plug into the right side of the 5-pin connector on the CAT4101 board. Install the panel jack with the USB symbol showing on top, so that USB cables will plug in correctly when their USB symbol is on top.



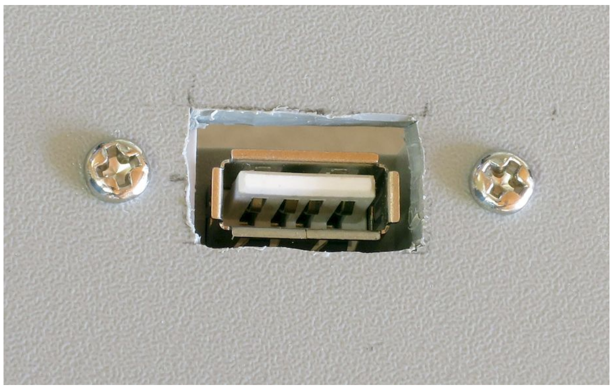








The USB jack on the back should look like this if the panel jack is correctly installed:



Run a red jumper wire from the 5V Arduino header to the + voltage strip on the lower right of the power board; run a black jumper from the nearby ground header to the adjacent ground strip:

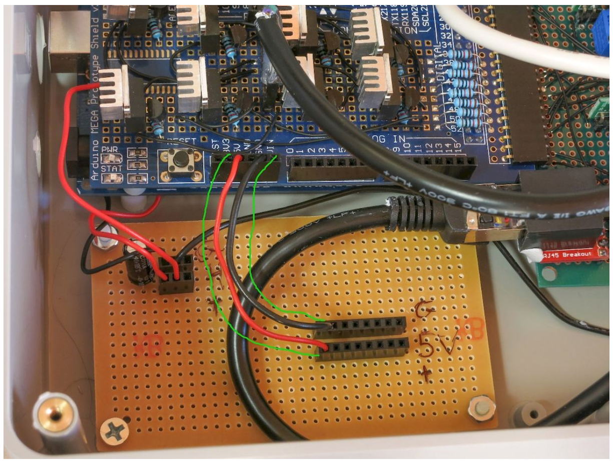
From this point on, the controller box is capable of running the “Dazzler/Serial Test” programs (described at the end of the LED dome wiring instructions) that will light up the wired LEDs in the RTI dome, to check whether all the LED wiring connections are correct.

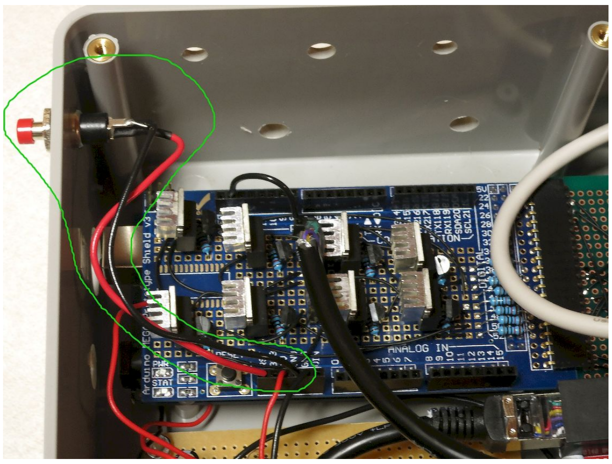
Next, the reset switch. Grab one of the two red pushbutton switches, solder wires to both terminals, put on heat shrink insulation over the exposed leads if necessary, then install it in the small hole in the upper left corner of the left side. Connect the wires to the Reset and GND headers in the lower left Arduino header row, as shown circled in green in the picture below (doesn’t matter which wire is which for this connection):

Apologies for the lack of pictures on this step, but hopefully it’s simple enough to understand from just the one picture.

Take a 3mm red LED, and insert it into the 3mm LED holder. There should be a little plastic insert inside the LED holder with two small holes – make sure the LED leads go through the two small holes separately, which keeps them from shorting out.

Solder a 560R resistor to the longer LED lead (the anode), then solder a red wire to the other end of the resistor,





and a black wire to the shorter LED lead (the cathode):

Trim the excess leads off, and cover the exposed wires with heat shrink tubing to prevent shorting:

Run the two LED leads through the hole in the upper left front panel you drilled earlier; run the nut over the two wires from the inside and secure the LED holder in place. Plug the red wire into the + voltage strip for high voltage, plug the black wire into the ground strip:

If you want to check the LED light, just plug in the power supply to the power jack on the left. The light will only go on if this main power supply is plugged in; if you only plug in the USB cable on the left, the Arduino will start up, but no power will go to either the power LED or the LEDs in the dome. The resistor value was chosen so that the LED would work with any voltage from 7.2 to 12 V DC without being too dim or blowing out.

Time to wire up the two 10K potentiometers. Solder wires to the first one as shown in the picture below (colors matter here):

Then solder the short red and black wires to corresponding terminals on the other 10K pot:

The idea here is that you'll be applying a common 5 volts across both sets of corresponding terminals, so you'll only need to plug one set of wires into voltage/ground terminals.

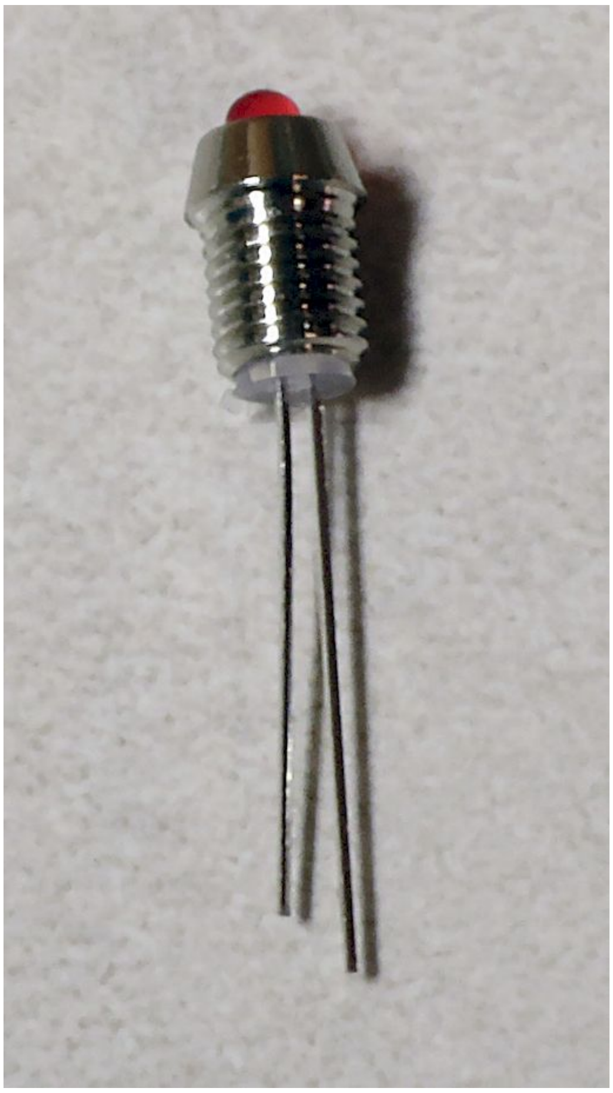
Now solder longer red wires to the center terminals of both potentiometers:

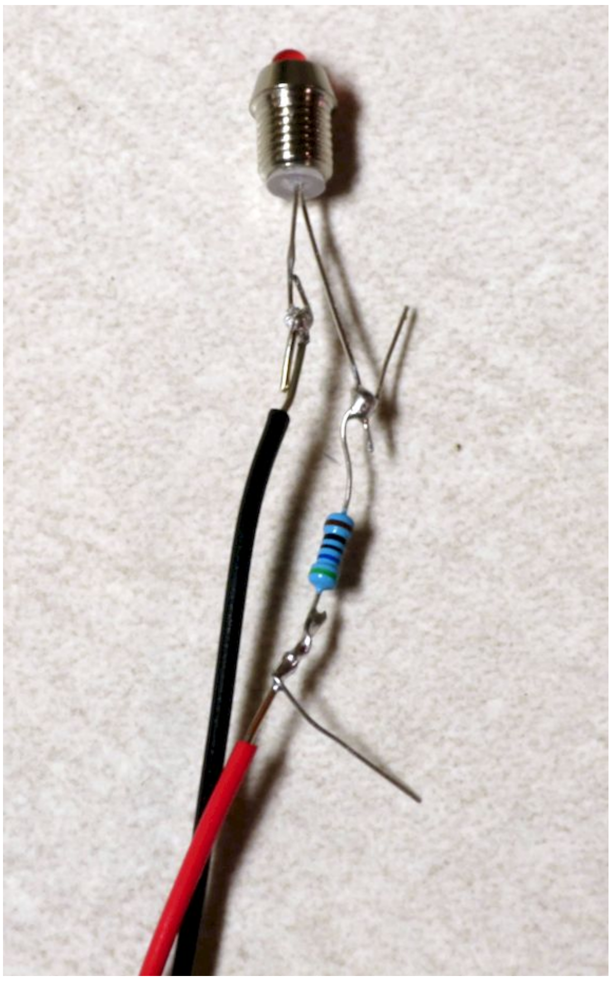
Install the two potentiometers into the left two holes of the control box. Plug the red and black outside terminal wires on the left potentiometer into the ground and 5V strips on the lower right of the power board.

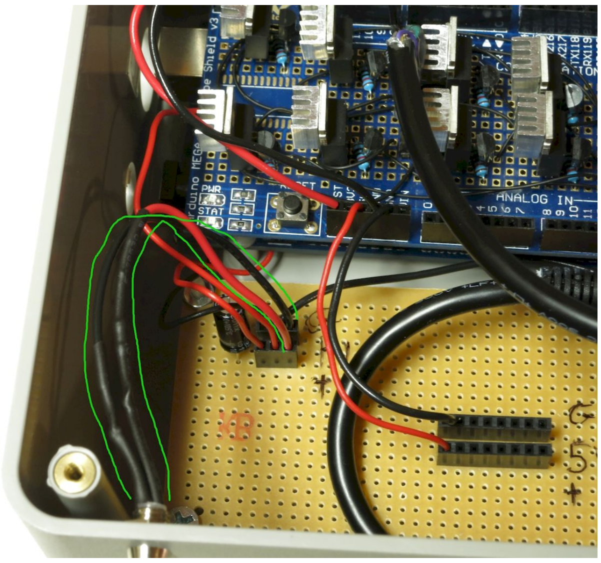
Plug the long red wire from the left pot into Analog In 7, from the right pot to Analog In 8.

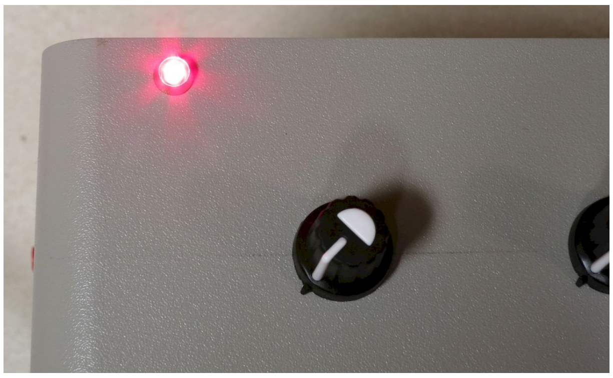
You can put knobs on the pots now:

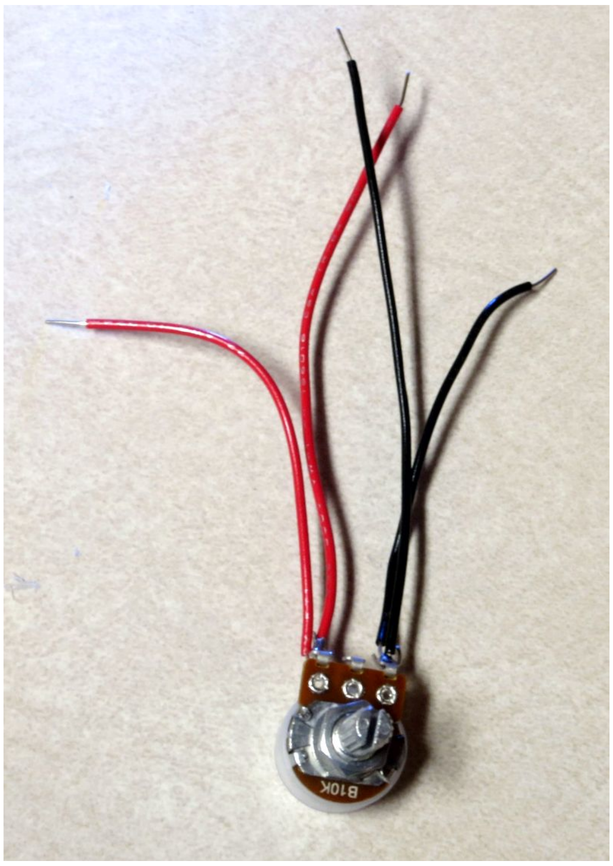
If you're installing the OLED display, re-attach the four connector wires you made earlier as before: black to GND, red to VCC, black to SCL, red to SDA. Make sure you remove the plastic display protector sheet from the

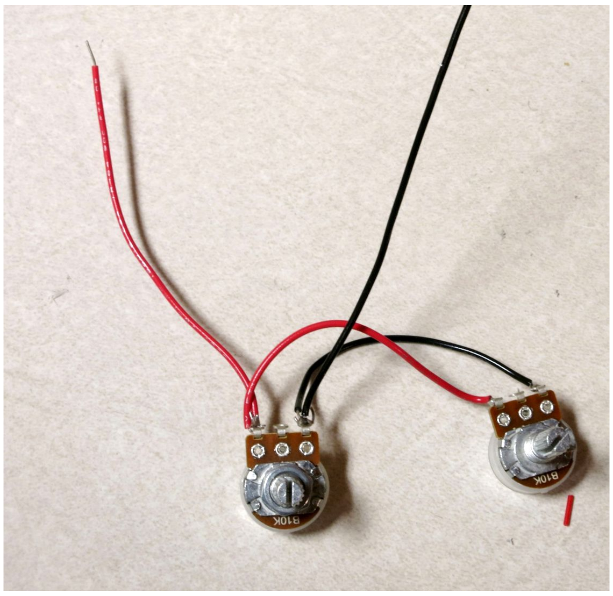


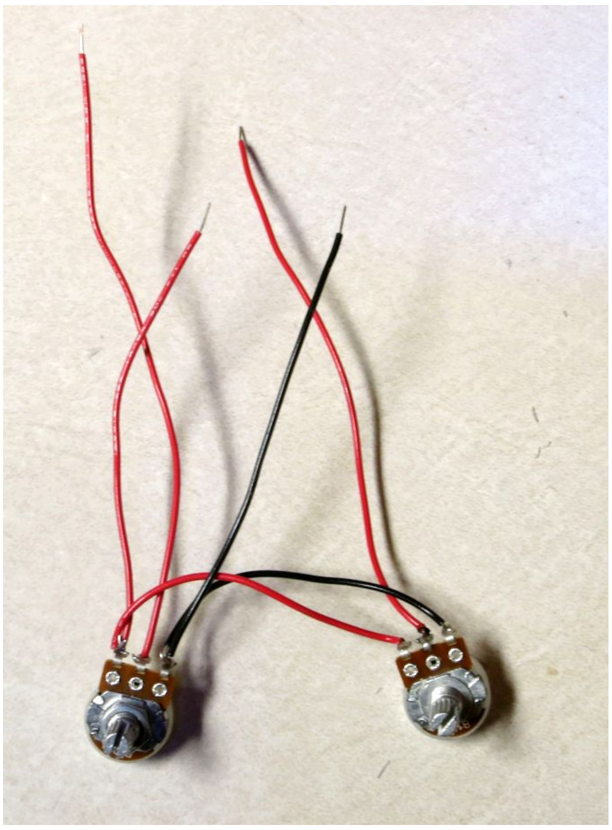


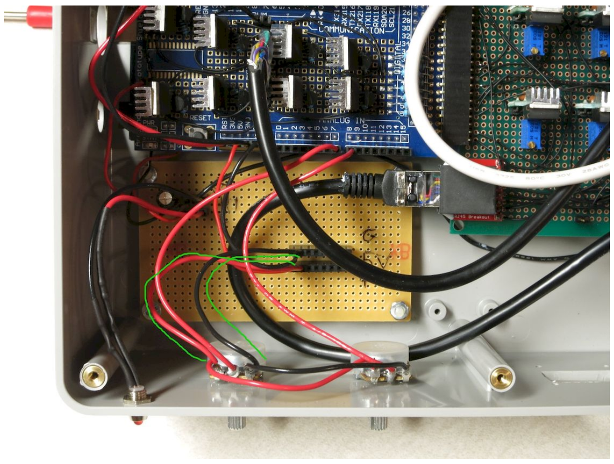


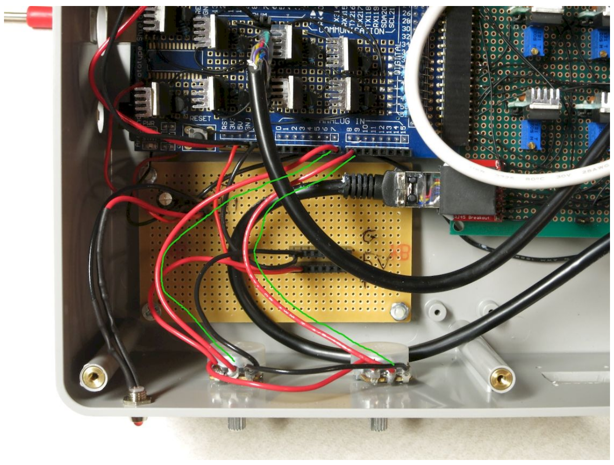




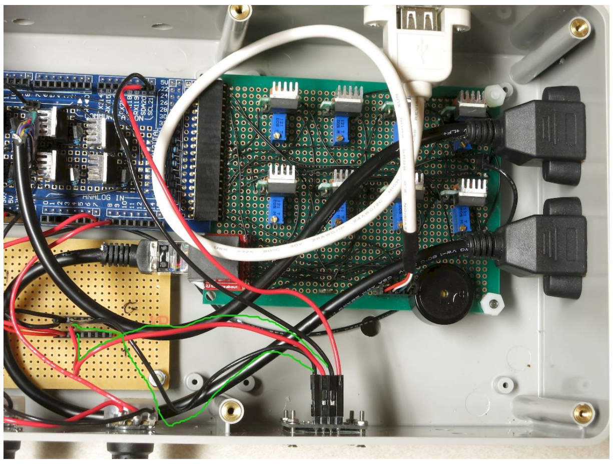

B10K
B10K



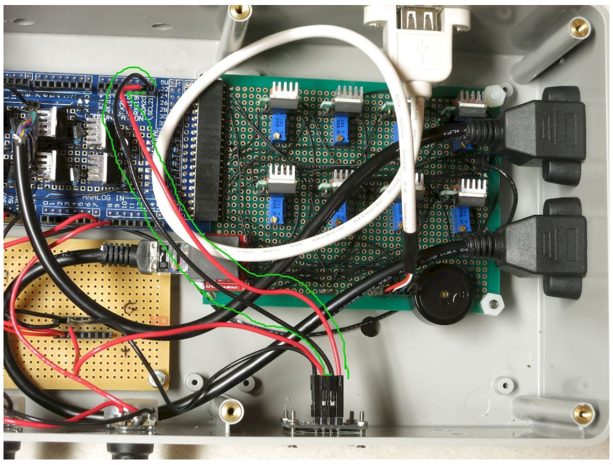




OLED screen. Attach the OLED display to the enclosure using screws and nuts. Connect the red VCC wire and the black GND wire to the +5V and ground strips on the lower right of the power board, respectively:



Connect the SDA and SCL wires to the matching Arduino headers on the top right of the board:



If you want, you can check the installation now with the RTIMage_OLED_Tester program discussed in the previous OLED test section.

Grab a red and black pushbutton switch, and wire them as below:

The install them in the two right holes in the front panel, red button on top:

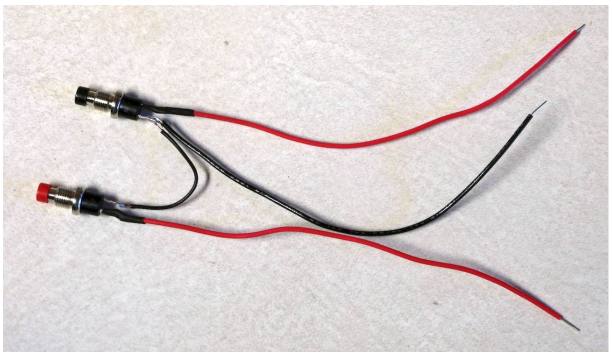
Run the black wire to the ground header on the lower right of the power strip:

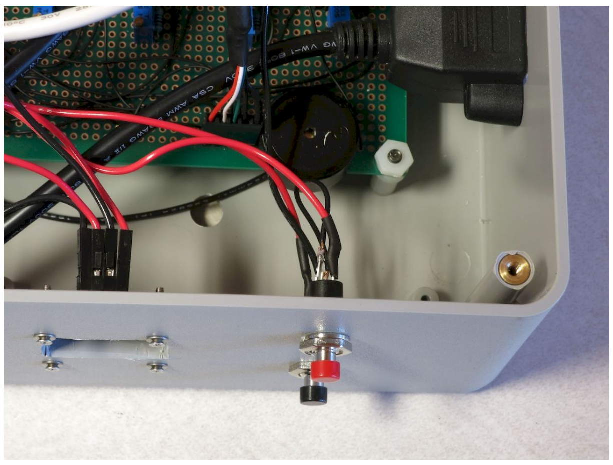
Run the red wire from the red button to ANALOG IN 13, from the black button to ANALOG IN 12.

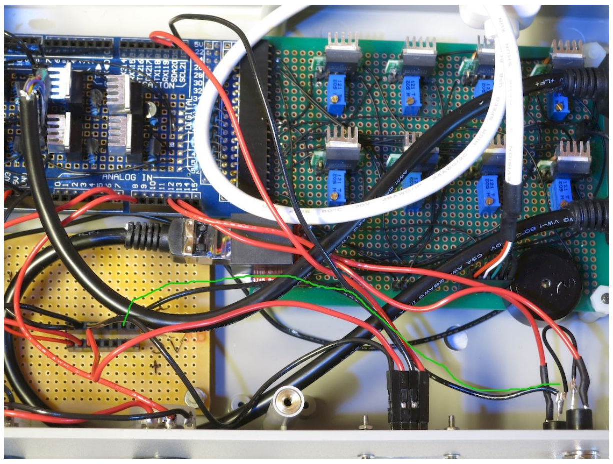
"Huh? Analog?", you might ask if you've worked with Arduino before; switches are normally connected to digital inputs. You can define any of the analog pins as digital inputs, and I do that in the software that runs the control box. Just trying to keep the wiring a bit neater.

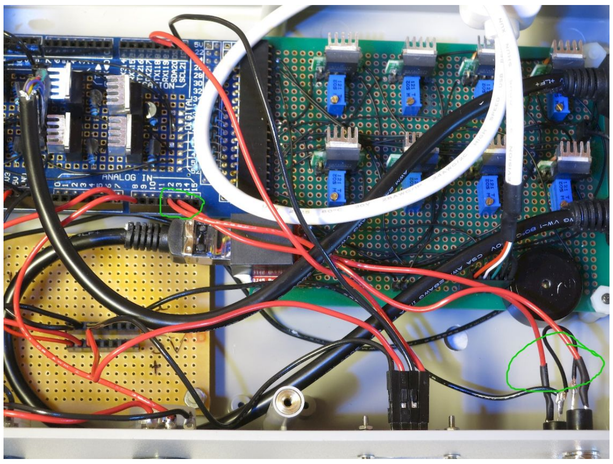
Grab the three toggle switches, and solder up the central terminals with black wire; these will connect to ground:

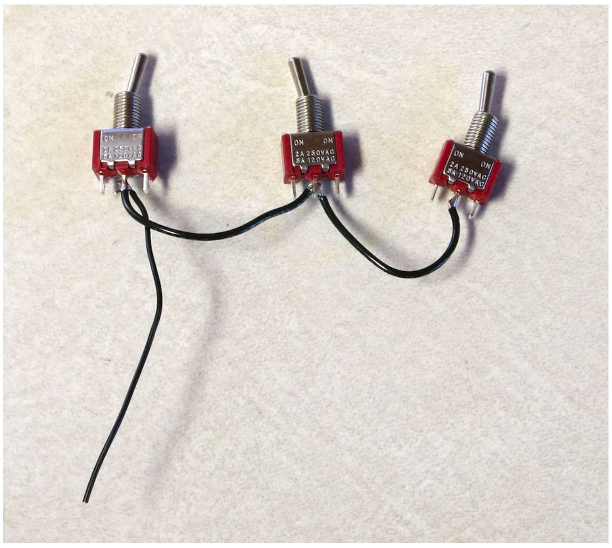
Solder a red wire to the left terminal of each toggle switch:

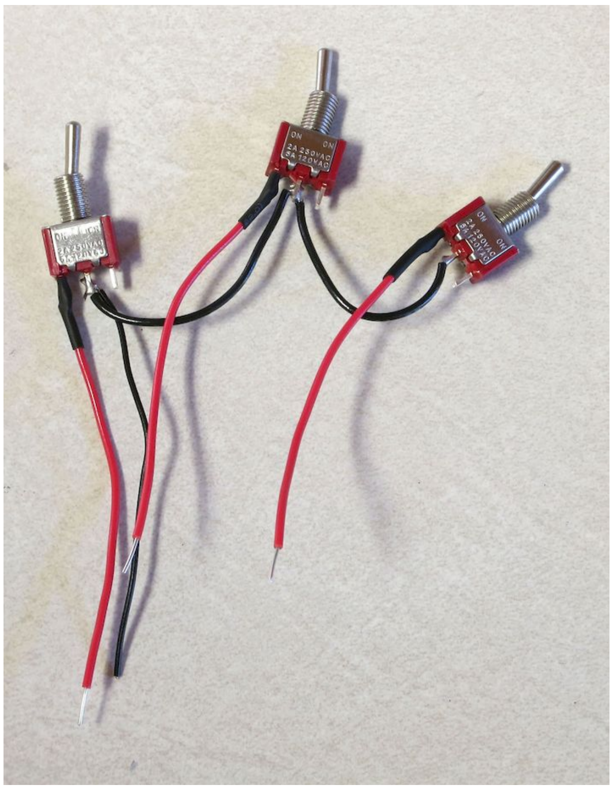




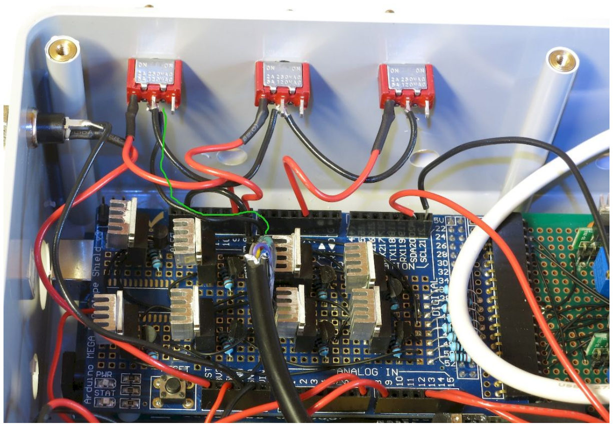






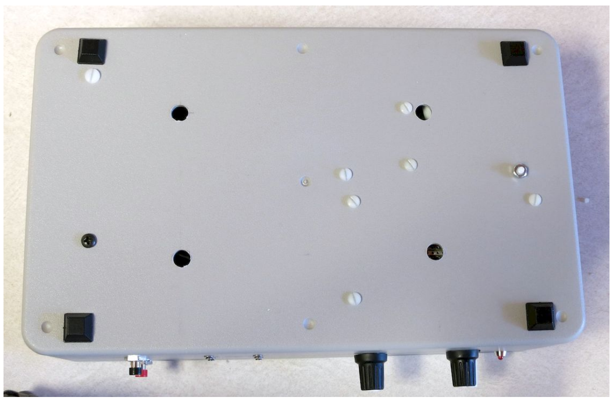


Install the three toggle switches into the holes on the rear panel. Then plug the ground wire into the open hole on the double header on the MOSFET shield:



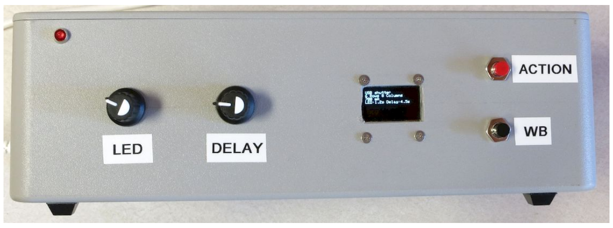
Install the left red wire into Arduino header 9; center red wire into Arduino header 8; right red wire into Arduino header 3.

Last step in the control box assembly is to stick the rubber feet onto the bottom. If a screw sticks out farther than the feet, you'll need to trim it to a shorter length:



Hey – control box is pretty much done! I put labels on the controls and jacks to identify them. I'll go through the functions now as a sneak preview of the operating manual.

## 11.1 Front panel



**LED** Sets the length of time the LED is lit, to allow the camera to take the photograph.

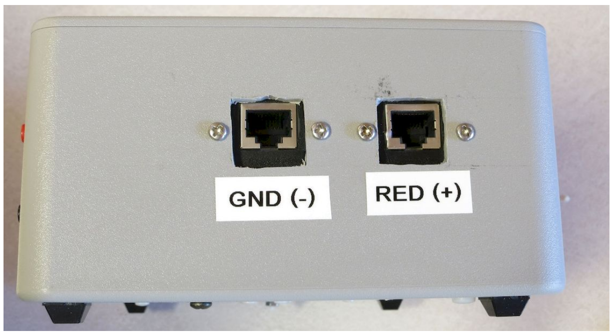
**DELAY** Sets the length of time between the LED going off, then going back on again. This is to allow the camera to process and save the photography.

**OLED display** Shows various relevant pieces of information depending on the mode

**ACTION** Starts the actual process of turning on LEDs for the RTI data acquisition.

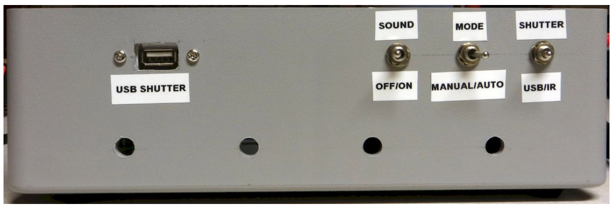
**WB** This turns on the LEDs in the top row in sequence, useful for setting the exposure parameters and the White Balance (hence WB).

## 11.2 Right panel



The Ethernet cables from the RTI dome plug in here, positive (red) on the right, ground (black/white/whatever) on the left.

## 11.3 Rear panel



**USB SHUTTER** This is where you plug in a cable that has either a mini-USB connector (for Canon cameras with CHDK to control the shutter), an IR LED (for cameras that support IR remotes), a custom remote connector (for cameras that support a wired remote), or a servo control cable (for cameras that have no built-in remote capability). These will be described in more detail in an upcoming step.

**SOUND OFF/ON** Controls whether the built-in beeper makes sounds or not.

**MODE MANUAL/AUTO** In Manual mode, you need to manually press the camera shutter, then advance to the next LED light. In Auto mode, the control box automatically fires the camera shutter, then advances to the next LED.

**SHUTTER USB/IR** In USB mode, the shutter is controlled through a USB cable (either CHDK or a custom remote connector). In IR mode, the IR LED plugged into the USB jack controls the shutter. If you're using Bluetooth control (also described in an upcoming step), you will need to have this switch in IR position.

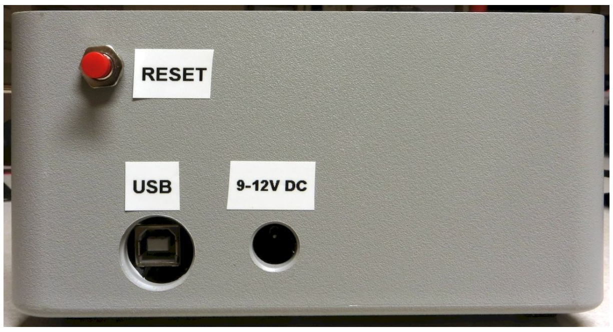
All of these switch functions are set in software, so if you decide you'd rather have a switch do something else, you can modify that in software.

## 11.4 Left panel

**RESET** Resets the Arduino, stopping all running operations, and restarting the software from the beginning. Useful if something goes wrong.

**USB** Connects the USB cable that runs to a computer, for uploading software programs.

**9-12V DC** Main power jack plugs in here. It's a center-positive 2.1 mm ID / 5.5 mm OD jack if you want to get technical.



Now that the control box has been assembled, the big question is, "Does it work?". You can test the basic output to the RTI LED dome by connecting it to the control box, uploading either the "Dazzler" or "Serial_Test" programs, and seeing if the LEDs light up as they should. But neither program relies on any of the control box controls, switches or knobs, to work.

You could upload the main RTI control program, and see if the box works, but that program doesn't have any diagnostics to help you if something isn't working right. So I've written a small program, "RTIMage_Controls_Tester", to help with debugging hardware issues. It's in the Files section of the Hackaday.io page; unzip the folder into your Arduino documents folder then upload it to the Arduino. It will work fine with only power from the USB cable, you don't need the main power plug.

---

**Note:** The first section assumes you've installed the OLED display. If not, don't worry – there's an option for control boxes with no OLED.

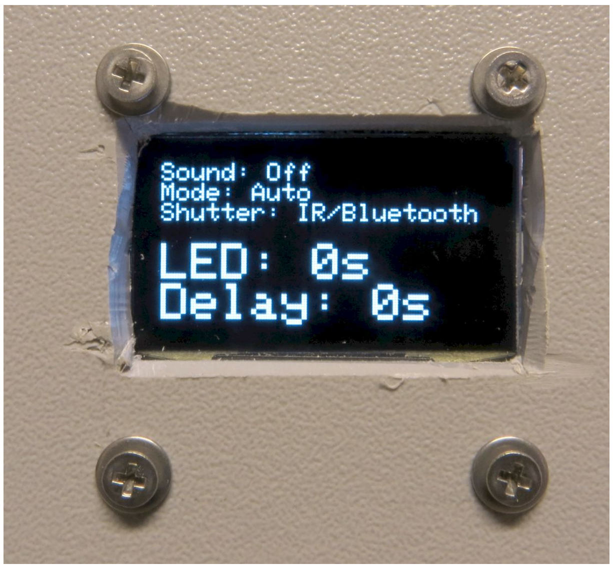
---

You'll get a splash screen with the name of the program:

After a few seconds, you'll get a status screen showing up:

The top three lines indicate the status of the three toggle switches on the back of the control box. As you flip these back and forth, these status lines should change to reflect that. If they don't, check to make sure that the ground wire connected to these three switches is properly connected, and that the red wires attached to each of the switches are in the correct Arduino pin header.

The bottom two lines reflect the status of the two potentiometer knobs, LED (left knob) and Delay (right knob). These have been arbitrarily set in the program to go from 0 (turned all the way to the left) to 100 (all the way to the right). So the screen above indicates that both knobs are turned all the way to the left:


Sound: Off
Mode: Auto
Shutter: IR/Bluetooth
LED: 0s
Delay: 0s

If I turn both knobs a random amount to the right:

The status screen should reflect a change in the two values proportionate with how far you've turned the knobs:

If both knobs have no effect, one likely candidate are the wires from the potentiometers going to the +5V and ground headers on the power board; try checking/changing those first. Also check to make sure the wires are in the correct Arduino pin headers. If one works but the other doesn't, check the connector from the inoperative potentiometer to the Arduino board to make sure it's seated firmly in the correct Arduino pin.

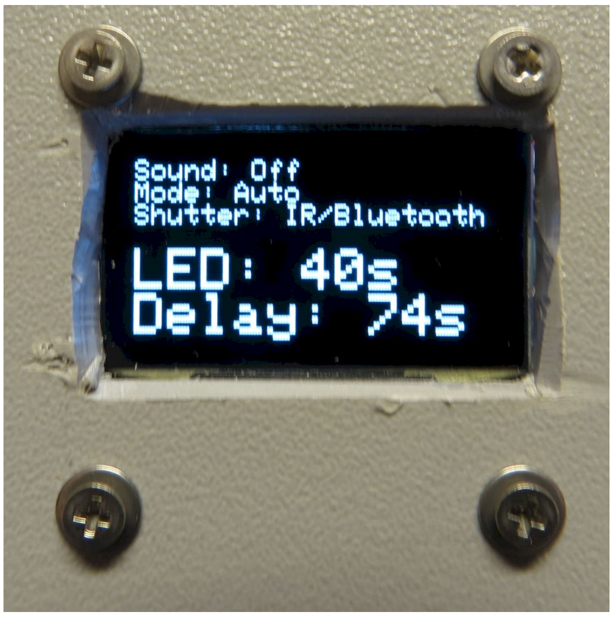
For the two buttons on the right, Action (red) and WB (black), press and hold each one separately. When you depress the Action button, you'll hear a short beep; as long as you keep that button depressed, this message will show up on the display:

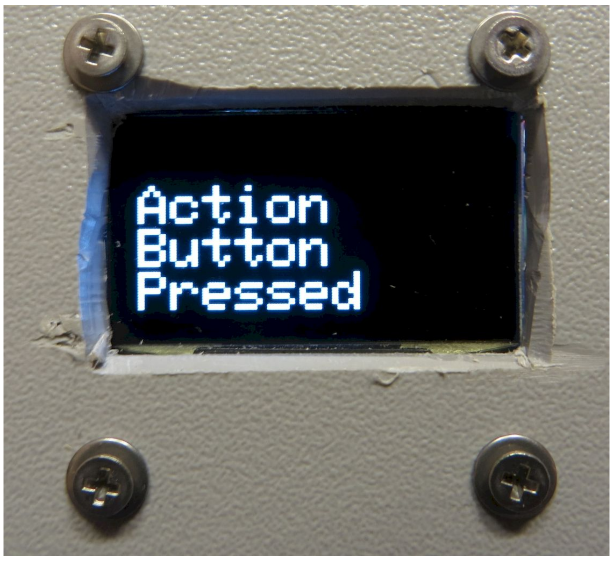
Similarly, for the WB (white balance) button, you'll get this:

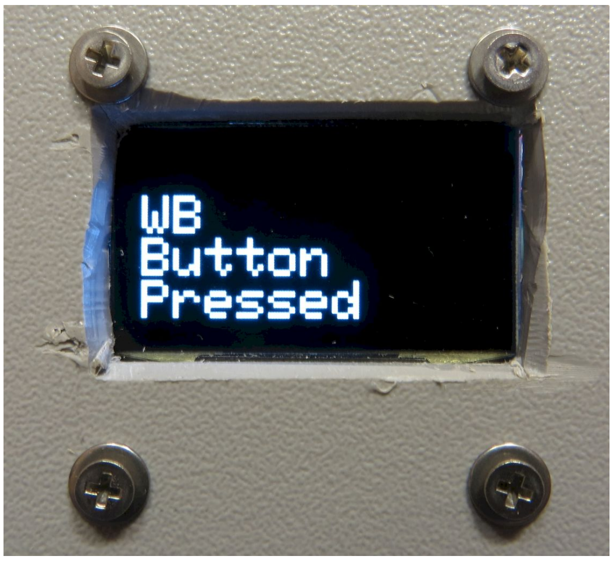
After releasing either button, the screen will go back to the status display. If pressing a button has no effect, check that the wires from the buttons are firmly in their proper connector, both the wires to the Arduino and the wire to the ground header.

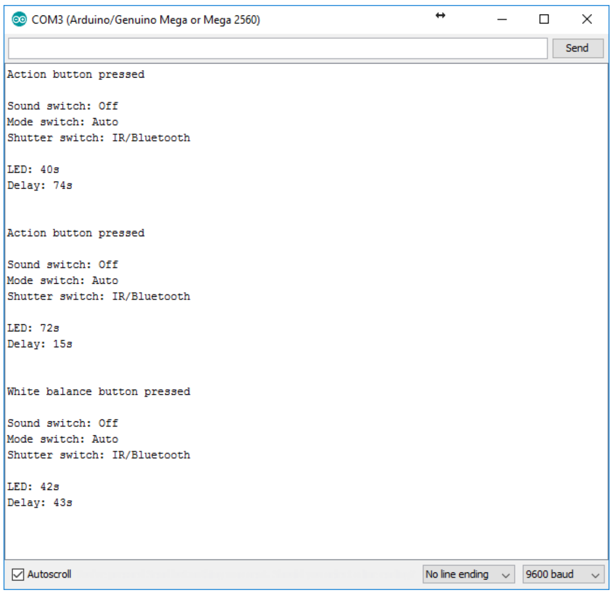
If you don't have an OLED display, there's another way to check the status of the toggle switches, potentiometers, and push button switches. You will need to have the USB cable connected to both a computer and the control box. In the Arduino IDE, enable the Serial Monitor from the Tools menu. When you press either the Action button or WB button, the status of all switches and potentiometers will be printed out:

To check the controls for proper operation, change their setting, then press one of those two buttons again. If you press a button switch and nothing happens, that likely means that one or both of those buttons are working properly, and you need to check their related connections.


Sound: Off
Mode: Auto
Shutter: IR/Bluetooth
LED: 40s
Delay: 74s


Action
Button
Pressed


WB
Button
Pressed

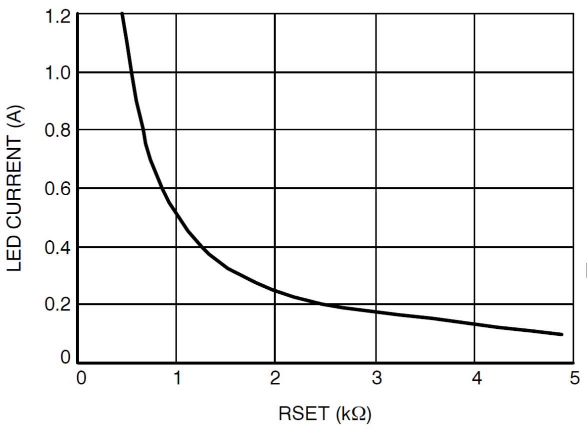


```
Action button pressed

Sound switch: Off
Mode switch: Auto
Shutter switch: IR/Bluetooth

LED: 40s
Delay: 74s


Action button pressed

Sound switch: Off
Mode switch: Auto
Shutter switch: IR/Bluetooth

LED: 72s
Delay: 15s


White balance button pressed

Sound switch: Off
Mode switch: Auto
Shutter switch: IR/Bluetooth

LED: 42s
Delay: 43s
```

One more step, and then you can close up the control box.

When acquiring an RTI dataset with the RTI LED dome, all of the lights need to be the same intensity. This is done by setting all of the output currents controlled by the CAT4101 chips to the same value. This step goes through the process.

The CAT4101 current is controlled by an external resistance Rset that runs between pin 4 and ground. The higher the resistance, the lower the current. The CAT4101 datasheet has this graph and table that shows the relationship between Rset and output current:



The RTIMage control box you've built has two resistances connected in series to every CAT4101 as Rset. There's a fixed resistor of 560 ohms that makes sure you don't exceed the maximum rated current for the CAT4101 of 1 amp. There's also a 5k trim pot connected as a variable resistor, so that you can set the current for any value from about 1 amp down to about 100 mA (0.1 A). Why set a current value less than the maximum?

- There are some cases where you don't need full light intensity. I'm working on using USB microscopes to do microscopic RTI with this system, and full light intensity fully saturates the image sensor on at least two models I've tried; I have to reduce current down to 150 mA before the sensor is no longer saturated.
- If you're going to use the system in portable battery-operated mode, turning down the current will increase battery life.
- The more current you run through an LED, the hotter it gets, and the shorter its lifespan. This is the least critical factor, though. Each LED is only going to be on for a few seconds at most, so it's not going to get very hot. Lifespans for LEDs are measured in thousands of hours of continuous use, during which time it can get quite hot even with a decent heat sink. Even if you assume a comparable lifetime with intermittent

| LED Current [mA] | RSET [Ω] |
|---|---|
| 100 | 4990 |
| 200 | 2490 |
| 300 | 1690 |
| 400 | 1270 |
| 500 | 1050 |
| 600 | 866 |
| 700 | 768 |
| 800 | 680 |
| 900 | 604 |
| 1000 | 549 |

use at cooler temperatures, you're talking about the ability to run millions of RTI data acquisitions before any visible degradation in LED intensity takes place.

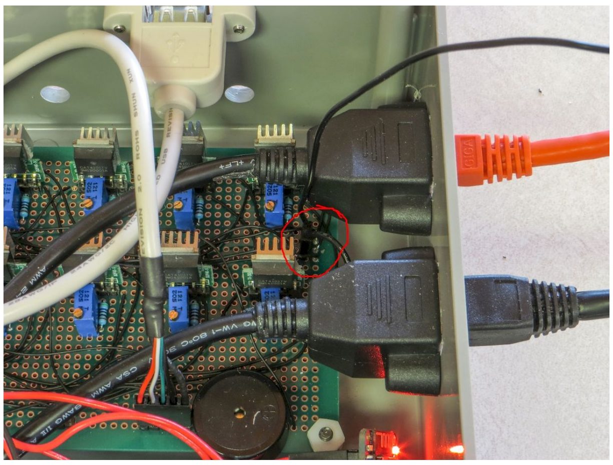
- On the flip side, if you need to stop down the aperture to get maximum depth of field, you'll probably want to increase the current to maximum intensity in order to get the shortest possible exposure time.

I usually set the current level for my domes at 700 mA (unless it requires a lower level). 700 mA is the "nominal" maximum continuous operating current specified by the datasheet for white Cree LEDs with a color temperature less than 5000K, which are the ones used in this dome. This is more than bright enough for short exposure times with both my 12" and 18" diameter domes, and likely would be fine for larger domes as well. However, since the LED operation isn't continuous, currents up to 1 A are fine as long as the LED isn't on for more than a few seconds; it just won't get hot enough to degrade. I've used LEDs running at 1 A for many runs, and had no problems at all.

One way to choose the current level for your dome is to use the lowest current that gives you reasonably short exposure times, but no greater than 1 A. Keep in mind that the camera will be in a fixed, rigid position, not moving at all. If the camera isn't moving, exposures of up to a second or more can be considered "short", since there isn't any motion blur with a fixed, rigid camera.

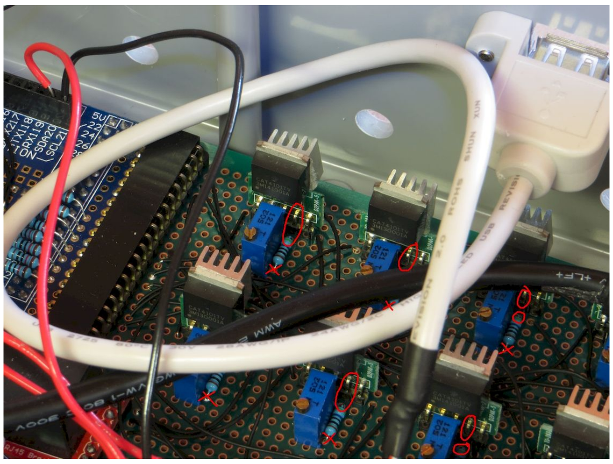
Okay, say you want 700 mA of current output – how do you set the RTIMage control box to output that? I've written an Arduino program that will help with that, and you can use that program immediately, but there's a step you can take that will speed up the process enormously.

First off, decide what current you want. In my case, 700 mA. Use the curve or the table above to determine what the correct value of Rset should be. I used the curve, and estimated that the Rset value should be about 800R. On the CAT4101 board in the control box, on the right side, there's a two-pin header that has one socket occupied by a ground lead running from the high voltage power supply. Cut a piece of black wire, trim off the insulation on both ends, and stick one end into the free socket in the header (circled in red):

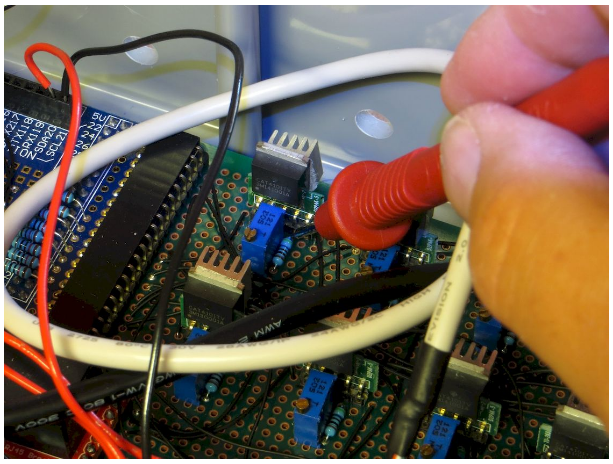


Grab your multimeter (set in resistance measurement mode), and connect the ground probe to that black wire.

To measure the Rset for each CAT4101 chip, you'll need to touch the red positive multimeter probe to either pin 4 on each CAT4101, or to the 560R resistor lead at the base of the CAT4101. In the picture, I've circled in red either a pin 4 connection on the CAT4101, or the resistor lead at the base of that pin; the Xes indicate resistor leads that you shouldn't use, since any resistance measurement there will only measure the potentiometer resistance, not the combined 560R-potentiometer resistance:



Here I am touching the resistor lead at the base of this CAT4101 chip, and getting an initial resistance reading of 3.1 kilo-ohms (3100R) on my multimeter, meaning I need to reduce the potentiometer resistance substantially. Set the range on your multimeter so that it shows a real value; in this case, I started on the 20K range, and increased the sensitivity when I got down below 2K ohms.



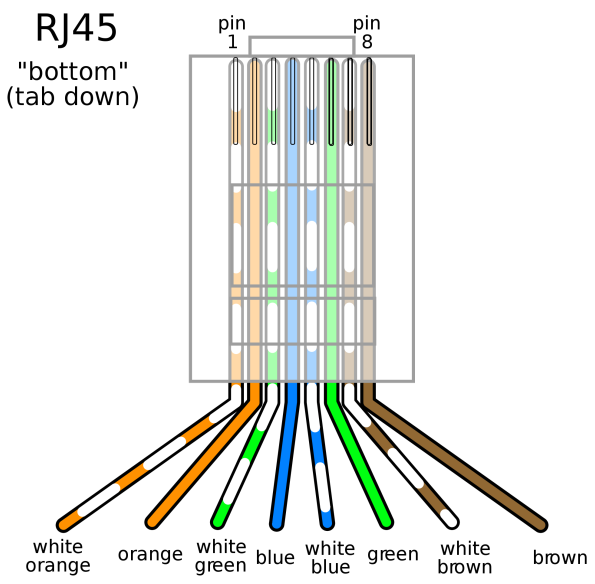
If I couldn't reach that resistor lead, I could touch the pin lead directly above it for the same purpose. It's important that you not touch any other leads/connections, just the one connected to Rset. It's also helpful to have two people on this operation, one to control the probe, the other to adjust the resistance. I've done it by myself, but it requires some contortions.

To adjust the potentiometer resistance, turn the little screw on the blue trim pot next to the CAT4101 counter-clockwise to reduce resistance, clockwise to increase it; use the multimeter reading to determine when you've gotten to the right value. It doesn't have to be an exact match to your target resistance, since you'll almost certainly have to adjust it further using the true current value. Even if you're way off, don't make a large number of turns on the trim pot screw without monitoring the resistance. It is possible to ruin a trim pot by turning it too many times past either the minimum or maximum level. And that would be bad. Minimum level for Rset is 560R, the fixed 560R resistor plus 0 from the trim pot; maximum is 560R plus 5K from the trim pot, 5560R. They may be a bit higher or lower than these two values, depending on the accuracy tolerances of the resistor and trim pot.

So I reduced the Rset value for this CAT4101 down to roughly 800R; I then repeated the process for the other 7 CAT4101 chips. The output currents are likely very close to each other with just this basic adjustment, but for the best matching of currents from different chips, they need all to be adjusted to have the same current value.

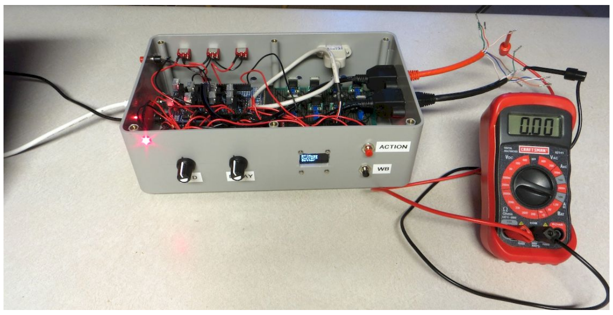
I've written a program called RTIMage_Current_Setter that will aid you in setting output currents. Upload this program to the Arduino in the control box. To set currents, you will need to have the main power supply plugged into the control box – the USB cable will not supply output power. You will also need to plug the two Ethernet test cables (which you made in a previous step) into the Ethernet connectors on the right side ("red to the rear", "close to the ground").

You will also need to set your multimeter in DC current measurement mode, have a current range set that can measure your target current value, and have the jacks in the correct positions. With my multimeter, the standard jack positions are only good for the 200 mA current range; I had to move the positive probe to a separate jack marked for 10 A, and set the current range to 10 A. You will be attaching the positive red probe to a single wire on the red Ethernet connector; with the black ground probe, you will be attaching it in order to each of the 8 wires coming from the Ethernet ground probe. You'll probably be following the standard color code for Ethernet wires to determine pin 1, pin 2 . . . pin 8 in order.



If you have an oddball cable color scheme, you should follow that instead, doing pins 1 to 8 in the correct order for that cable.

Here's a picture of my setup. I have the red positive multimeter cable running to the white/orange wire on the red (positive) Ethernet test cable (pin 1), and it will stay there for the whole process. The black ground multimeter cable is attached to the orange/white wire (pin 1) on the black (ground) Ethernet cable for the first current setting, but will be moved to pins 2-8 in succession to set the current value for all CAT4101s.

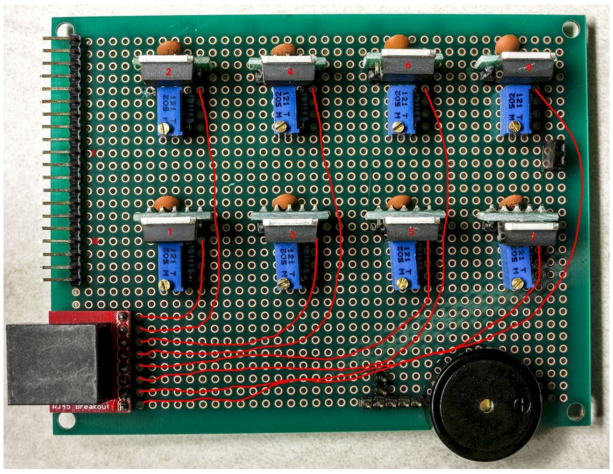


How do I know which CAT4101 corresponds to which pin number? I pulled up this handy picture from a previous assembly step, that labels the CAT4101s on the board with their matching pin number:

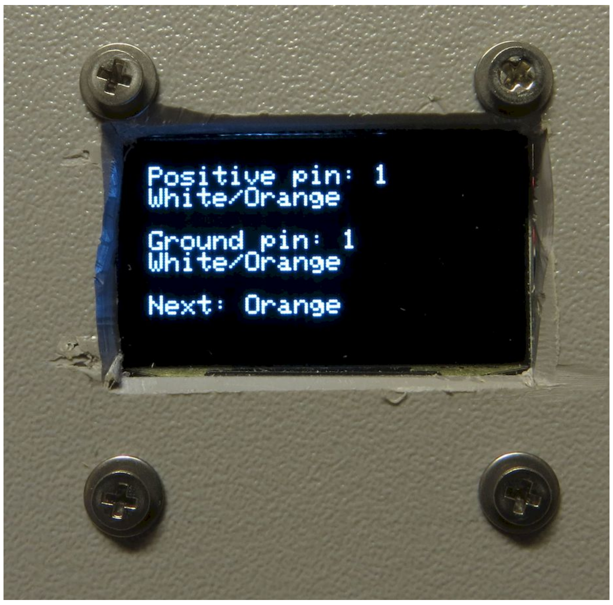
The immediately-following steps assume you have an OLED display installed. If you don't, there are related instructions at the end, but read this section first anyway.

When you load in the RTIMage_Current_Setter program, you'll see this splash screen on the OLED display:

To turn on the current to ground pin 1, press the red button; you'll see the screen change to:



```
RTIMage
Current
Setter
```



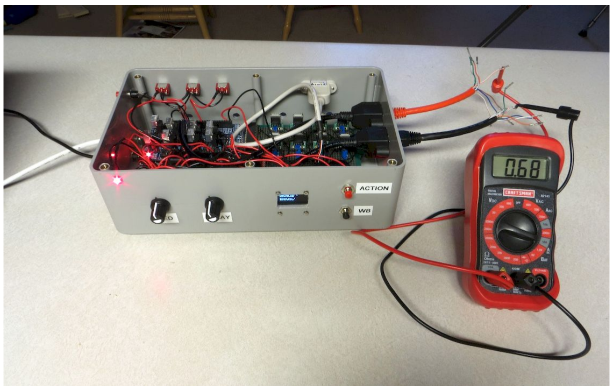
```
Positive pin: 1
White/Orange

Ground pin: 1
White/Orange

Next: Orange
```

The top lines indicate which pin / wire color you should have the positive multimeter lead connected to. This will always be the same. The next set of lines indicate which pin / wire color you should have the ground multimeter lead connected to. This will change as you go from the first pin to the last pin. The color of the next pin wire will be indicated at the bottom.

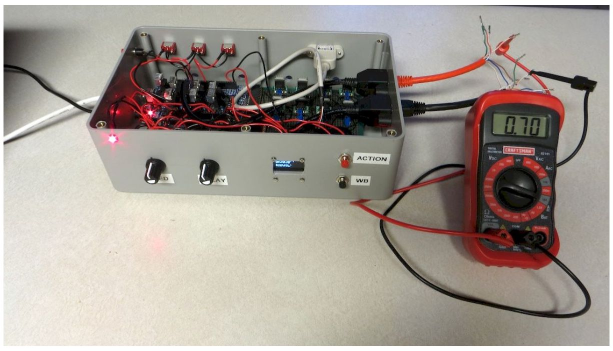
Pressing the red button will also turn on the current for 3 seconds, then turn it off. It's set to turn off to prevent possible overheating issues with your multimeter; my model says that you can only measure 10 A of current for a maximum of 30 seconds before you have to let the multimeter cool off for 15 minutes.

These systems typically measure current by running it through a standard resistor and measuring the voltage; since power goes as the square of the current in this situation, 1 A of current will create 1/100th the power dissipation of 10 A, so overheating shouldn't be a problem. However, keeping the flowing current down to a minimum is still a good idea.

So, I pressed the red button, and the current flowing through ground pin 1 showed up on the multimeter (in amps):



Hey, pretty close to the desired value of 700 mA (0.7A). While the current turns off after 3 seconds, you can turn it back on for the same pin by pressing the black button. I turned the trim pot down a touch to reduce the resistance / increase the current, then pressed the black button:

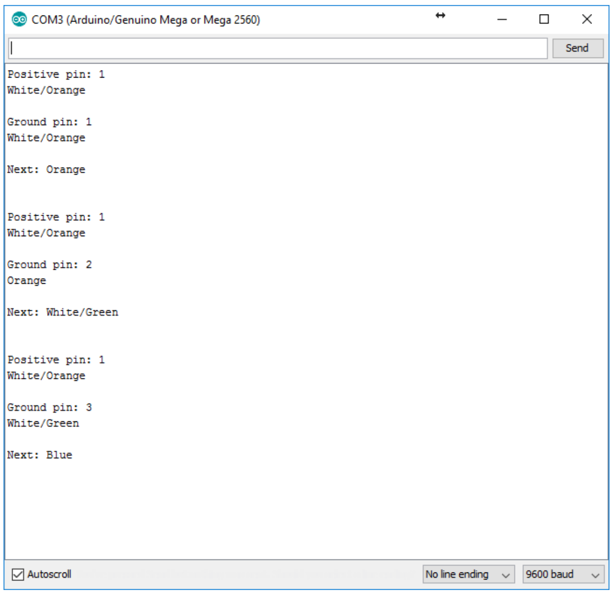


Looks like the desired value. Since the multimeter only reads to two decimal places, the current could conceivably be anywhere between 695 mA and 705 mA, which is +/- 0.5%, pretty decent. If I wanted to get closer, I could keep the current on continuously by pressing the black button continuously, and then tweak the current from below until it just changes from 0.69 on the multimeter to 0.70.

When you've got the current set for one pin, time to move on to the next one. The status screen will show you the next color wire you need to hook the black multimeter cable to. Do that, and press the red button; the status screen will advance to the next pin / color wire, and turn on the current for that pin (as well as showing the color wire of the next pin). Repeat the current setting process until you get to the last pin/wire; the status screen will let you know when you're there. When you're done with the last wire, pressing the red button will take you back to the splash screen; if you need to, you can start the current setting process over again by pressing the red button.

If you don't have an OLED display installed, turn on the Serial Monitor from the Tools menu in the Arduino IDE; you'll need to have the USB cable connected to the Arduino. Each time you press the red button, you'll get messages like the one below that tell you which is the current pin / wire color, and what's the next wire color:

When all the pins are set to the same current, you can put the lid on the control box and screw it down. Write down the current value you set somewhere; you'll want to have it handy when uploading the main control software to the control box.



```
Positive pin: 1
White/Orange

Ground pin: 1
White/Orange

Next: Orange


Positive pin: 1
White/Orange

Ground pin: 2
Orange

Next: White/Green


Positive pin: 1
White/Orange

Ground pin: 3
White/Green

Next: Blue
```

CHAPTER 12

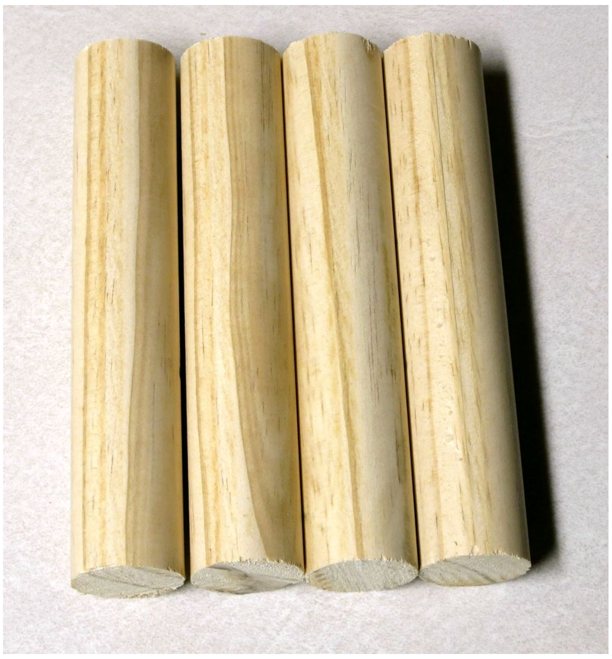
# Necessary hardware

## 12.1 Simple Stand

While there are ways you could use the RTI dome by lifting it up and placing it over the object you want to photograph, it makes more sense to have the dome/camera stationary, and have a way to easily swap multiple objects in and out of the system for easy data acquisition. One solution is put the dome on an elevated stand, and I'll first be describing the simplest possible stand design I can think of.

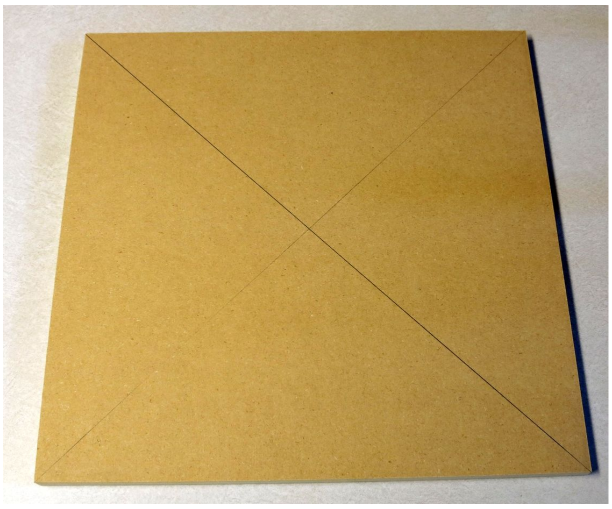
There are reasons you might want to modify this setup having to do with how you mount the camera, and I'll talk about that in the next step. But even there, these instructions will help you with that process. There's a more advanced/flexible stand design that will be described later.

Grab a wooden dowel about 1" or so in diameter, preferably a soft wood like pine. Cut it into 4 6" long segments for the stand legs:

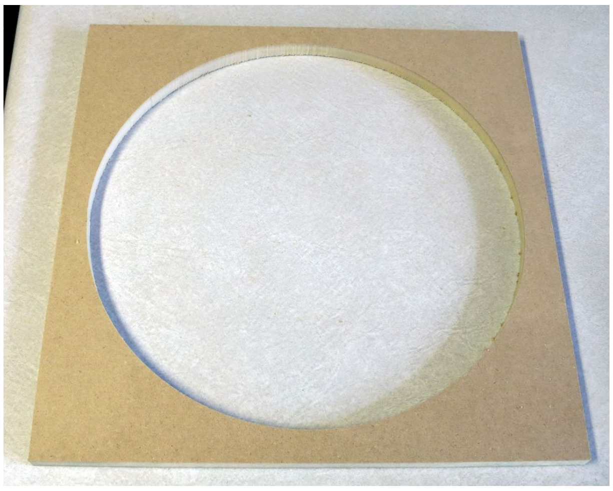


Cut a square board with dimensions equal to or slightly larger than the outside diameter of the RTI dome, including the rim. My dome has an OD of 12", plus a rim of 0.75", for an outside diameter of 13.5"; I added 1/8" for luck, and had a square board cut with dimensions of 13 5/8" by 13 5/8". My first stands were made out of wood, but I've had problems finding wood flat enough for this purpose. I have switched over to using ½" thickness medium-density fiber board (MDF); it cuts and works like wood, but even large pieces are usually flatter than wood panels. Just don't get it wet (or paint/seal it if it will be anywhere it might get wet).

Draw diagonal lines from the corners of the board to find the center:



Measure and cut out a hole centered on the middle of the board, with a diameter roughly equal to the inside base diameter of the dome; doesn't have to be exact:

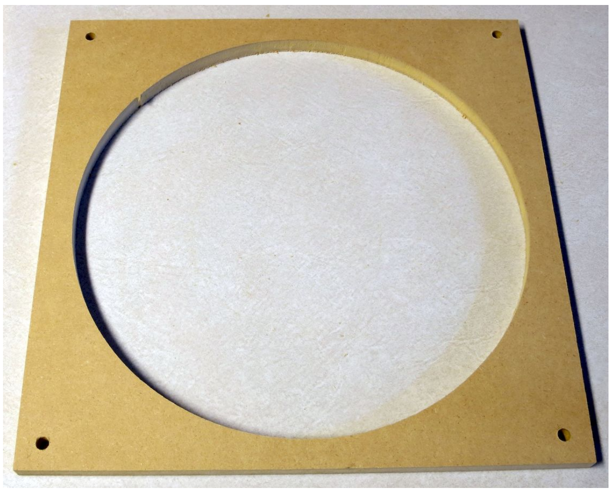


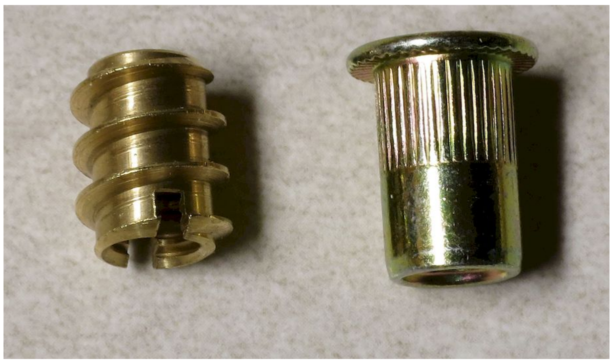
You could attach the dowel legs right now with glue, screws or nails, if you don't plan on transporting the stand. However, I like to make the dowel legs removable, so that you can break down the stand to a smaller size. Drill ¼" holes near the corners; I measure about 5/8" along the edges, and drill at the intersection:

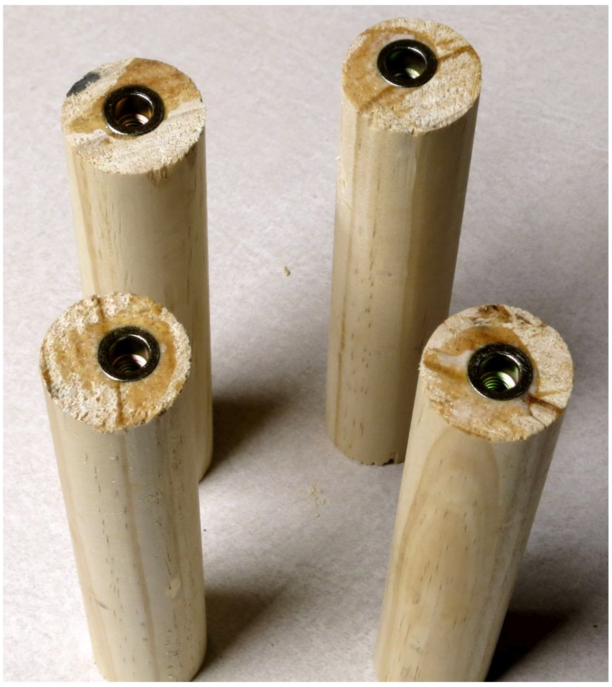
You now want to put ¼-20 threaded inserts into the top of the dowels. There are two varieties of these: screw in (left) and press fit (right):

With screw-ins, you drill a hole of a specified size (5/16" for ¼-20 inserts), then screw the insert into the hole. Tip: the slotted end, which you might think is used with a screwdriver to screw in the insert, actually goes in first. With press-fit, you drill a hole just a small bit smaller than the insert, then hammer it in. Screw-ins are more expensive than press-fits, and I had a few press-fits left over from a previous project, so I decided to use those. Turns out I didn't have a drill bit slightly smaller than the press-fits, so I drilled a slightly larger hole and then glued them in place. Hammered them down before the glue dried so that they would be flush with the top of the dowel:

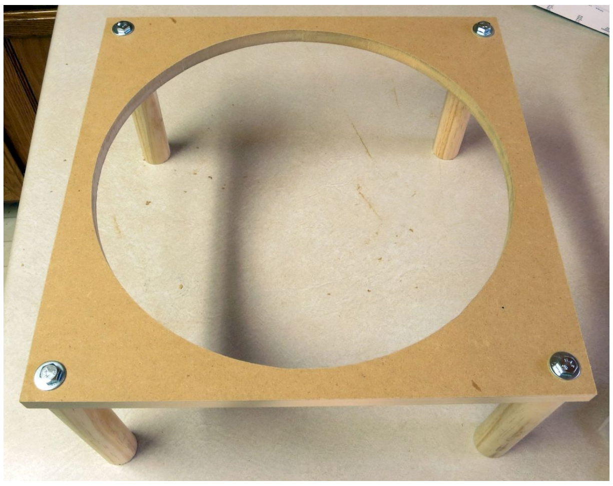
Also did a crappy job of putting the hole in the center with a drill. Tip: use a nail or screw to make a small starter hole in the exact position you want the hole, and first drill a pilot hole with a smaller diameter bit than the ultimate hole size.







With threaded inserts, you can run a 1” ¼-20 bolt (with washer) through the main bolt, and screw in the dowel legs:



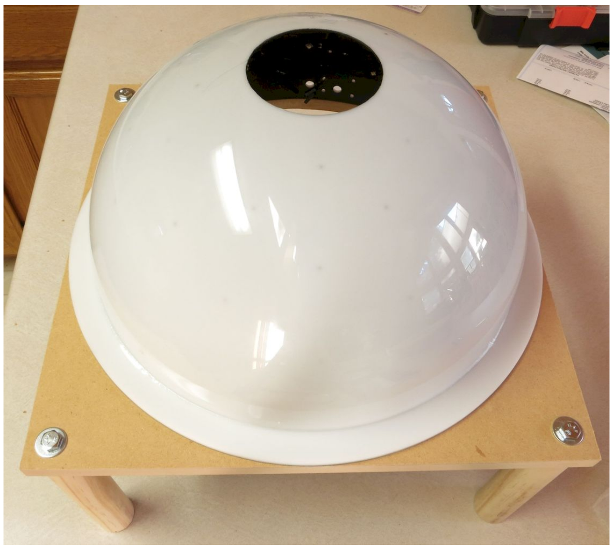
Put the dome on, and it fits:

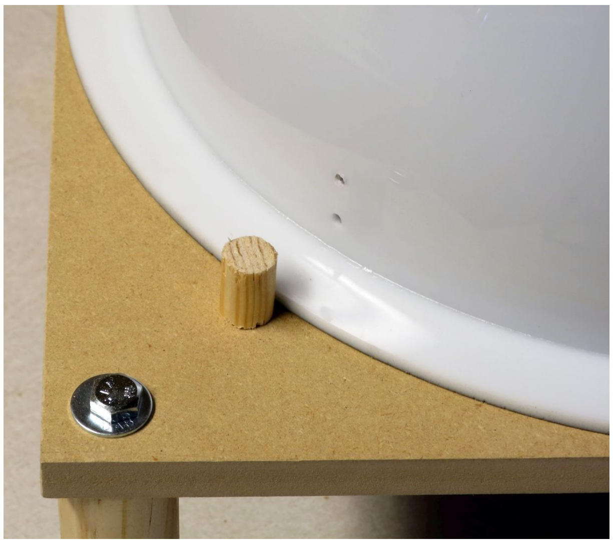
But it’s too easy to knock the dome off, or to the side. So, cut some ½” dowel pieces about an inch long, and glue them in next to the dome (be careful not to use too much glue that will ooze out and glue the dome in place as well):

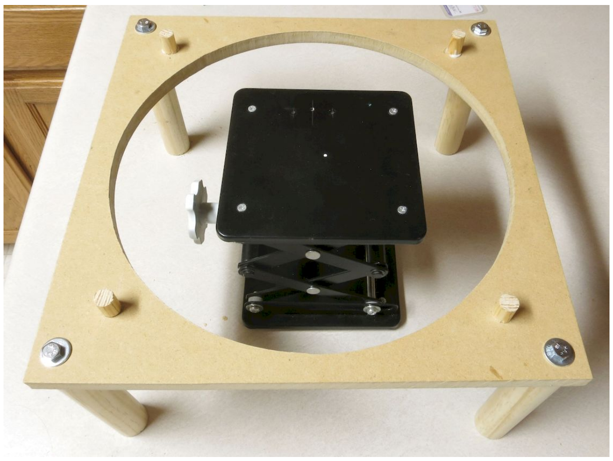
In the past, I’ve reinforced these with screws from the bottom, but I don’t really believe that’s necessary (and it runs the risk of splitting the pin). If the pin gets knocked off, it can just be glued back on.

For placing the surface of the object you want to photograph at the correct height, the base of the dome, you could build a box or stand out of wood/MDF, or even stack books up to the correct height. I use a lab jack, which lets you easily adjust the height to any required level:

This is a just a basic, simple design. You can leave it as is, or paint it in some festive color (I chose white). After painting, I added some rubber pads to the bottom of the dowel legs for skid and vibration resistance; I used some computer soundproofing foam I had lying around. For some options for camera mounting, you may need a larger version of this (see an upcoming section for more info).







## 12.2 Advanced Stand

The previous stand instructions let you build a simple stand that will work for basic functions, and is satisfactory for larger domes. However, for small domes (12”-15” in diameter), I’ve come up with an alternate advanced design that can be used in several different useful configurations:

1. Portable mode. Fits into a backpack (12” dome) or Pelican case, sits flat on a surface.
2. Stand mode. Dowel legs raise it above the ground, but an opening in the bottom allows for easy placement of low-profile objects.
3. Micrometer mode. A small microscope micrometer stage allows for accurate positioning of objects when doing macro or microscopic photography.
4. Open mode. The center piece is removed, and the object is positioned using a lab jack or similar stand. This allows the surfaces of thicker objects to be imaged.
5. Vertical mode. The dome is held upright, allowing the surfaces of larger object to be imaged (like pots, paintings, etc.).

Here’s a video that demonstrates the use modes of this stand.

This stand design requires a bit more work to construct than the simple one, but the extra effort is worthwhile because of the expanded flexibility. If you know someone with woodworking skills and tools, I’d suggest buying them a six-pack to help you build it; as you’ll see, my lack of those skills resulted in some less than optimal hacks to put it together (but it still works).

You will need the following:

- 2 flat square pieces of wood or MDF, at least the outer dimension of your dome including the flange. The dome I’m using here is a 12” diameter plus 0.75” flange around the edge, for a total outer dimension of 13.5”. I used 13.625” square MDF pieces, just to play it safe. 0.5” thickness.
- 4 wooden dowel pieces 6” in length, 1” to 1.25” in diameter. Soft wood (e.g. pine) is preferable.
- Vibration-dampening material (for the bottom of the dowel pieces.
- 2 wooden pieces 0.75” to 1” square by approx. 3” to 4” in length; exact length will depend on the size of the dome (see instructions).
- 8 ¼-20 threaded brass inserts.
- 8 ¼-20 1” hex bolts
- 4 ¼-20 metal washers (use with brace bolts)
- 3 2” long mending plates
- 2 1.5” utility hinges
- 8 3/8” #6 wood screws
- 3 ½” #6 wood screws
- 3 #8 ½” machine screws
- 3 #8 threaded brass inserts
- 4 #10 machine screws or bolts, 1” long
- 4 #10 brass knurled nuts
- 8 #10 metal washers
- 4 #10 neoprene/rubber washers
- Micrometer stage (optional)
- 1.5” or longer #4 machine screw, with nut/washer (optional for micrometer stage)

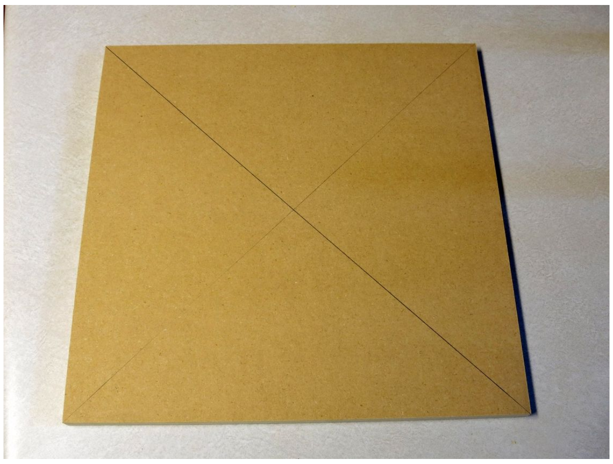
Optional pieces to secure stand for transport:

- 4 1.5” ¼-20 bolts
- 8 ¼-20 washers
- 4 ¼-20 wing nuts

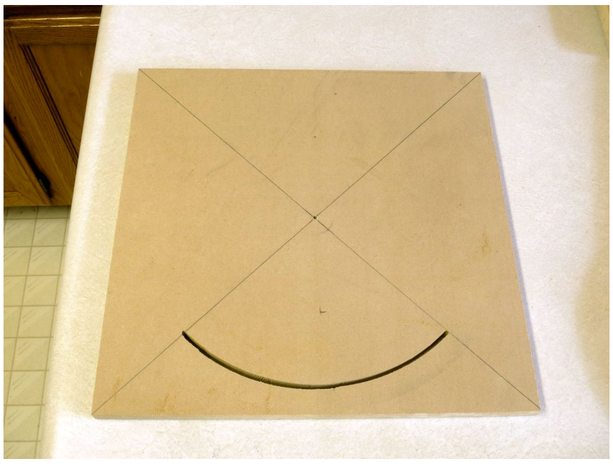
If you find I’ve forgotten something from the instructions, let me know.

**Note:** Before I begin, a quick note. I made a number of screw-ups in constructing the dome, even though I had a previous version in front of me to act as a model. And, as an added bonus, I also didn’t take pictures at every step. So, if the text and the picture conflict, follow the text. Also, read the directions all the way through (and watch the video), and use that as a guide for what you’re going to build. If something doesn’t make sense right away, read on, hopefully all will become clear.

Start with one of the large wood/MDF squares:



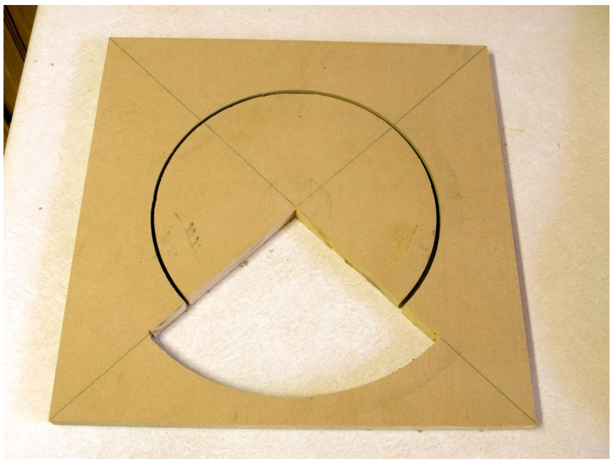
Draw lines between opposite corners to find the center point at the intersection. Draw an arc with radius equal to the inner dome diameter between two of the diagonals, not including the flange, and cut along that arc; for my 12” diameter dome, that’s a 6” radius:

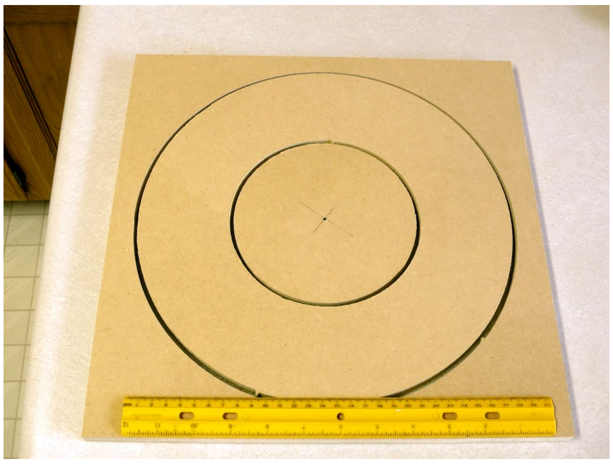


Now draw a circle with a radius 1.5” less than the inner dome diameter (4.5” in this case), and cut out a circle from the center. Then, cut out a ¼ radial slice from both the circle, and the remaining board:

The reason for this slice is to create an opening to allow you to put samples under the dome, and also manipulate the optional micrometer stage; more on this later. This is the bottom plate and the base of the center plate.

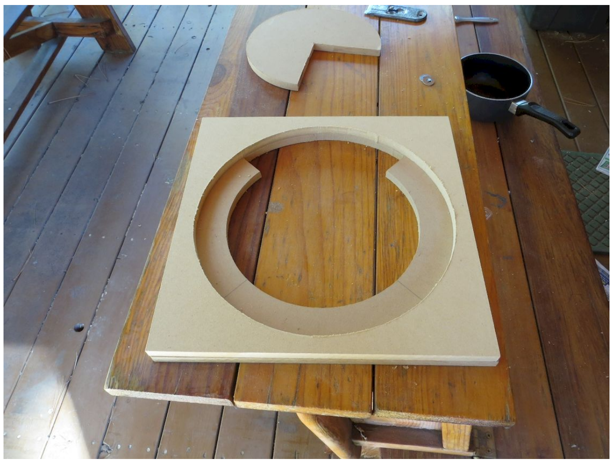
With the other square wood/MDF piece, find the center as before, draw a circle all the way around with the inner dome radius (6” for my 12” dome), and cut the circle out of the wood piece. Then, from the circular piece you’ve just cut out, cut out a circle ½ the diameter of your dome (a 6” circle for me, 3” radius).





Keep the square (the top plate) and the inner circle; the remaining round ring you can throw out, or find some other use for. This inner circle will be glued to the center plate later.

So you have two cut squares, top and bottom plates:

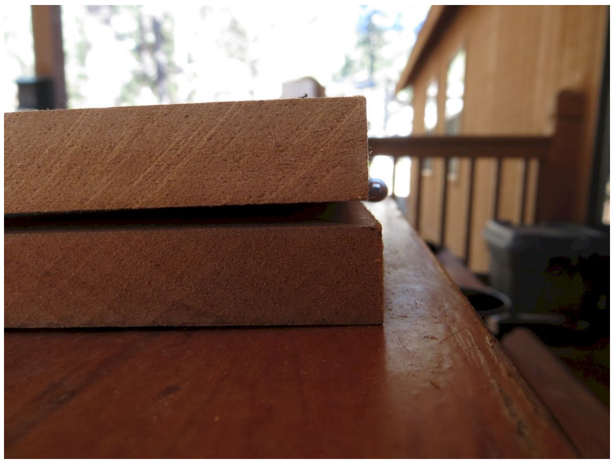


The dome will be attached to the top piece, and the top and bottom pieces will be connected with the two hinges. Place the dome on top of the top piece above, and mark the positions where the hinges should go so that they're as far from the edge as possible, but won't interfere with the dome:

Here are the two hinges laid in place in their initial positions on the bottom plate:

You can attach the hinges directly here, but that would cause a small problem. The thickness of the hinges will keep the top and bottom pieces from being parallel to each other; they will be slightly angled, with a gap at the hinged end and no gap at the far end:

Here’s the hinge end:



And here’s the opposite end:

If you decide to install the hinges this way, you’ll need to add spacers at the far end to compensate for this gap, and keep the top and bottom plates parallel. The alternative, which I’ve been doing, is to mill recesses in the two plates that the hinges will fit into flush. First I use the dome to draw an arc on the bottom plate, to make sure the hinges will fit:

Then I trace outlines around the edges of the hinges, to mark the places where I need to mill out a recess (reasonably close to the dome):

Now use these marks as a guide to creating similar recess marks on the top piece.

I used a Dremel with a mill attachment, along with a depth control guide, to mill out the recesses to the thickness of the hinge plates. Any woodworking hobbyist should have better tools and expertise to do this with. Here are the bottom and top plates, with the hinges in position in the milled recesses:

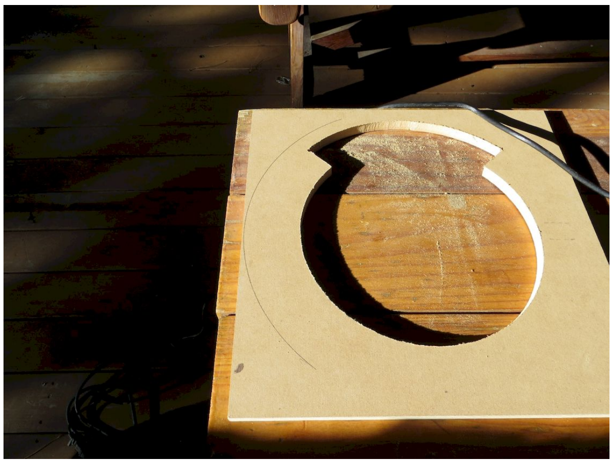
Now use 3/8” #6 wood screws to attach the hinges to the two plates. Do one plate first, then the other one; try and line the plates up so that the hinge edges are parallel to each other. It can also be helpful to drill a shallow starter hole for the screws.

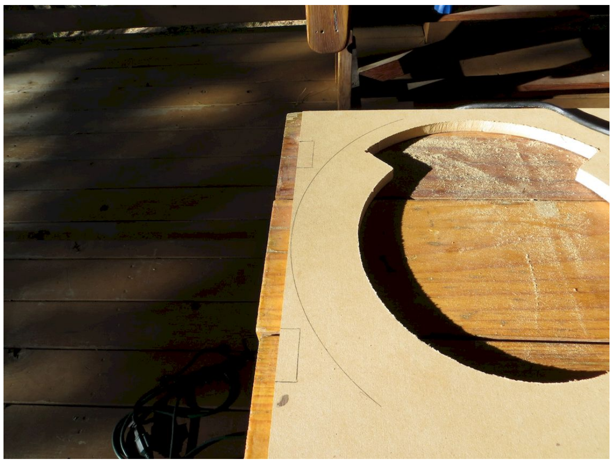
Flip the plates closed, and check to make sure the plates are flush along their entire length (this is the hinge end, which would have a clear gap if the hinges weren’t installed flush with the surface):

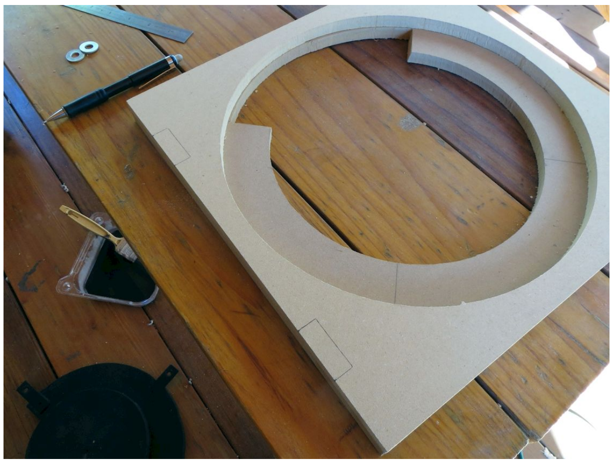
Now you need to drill four holes for the stand dowel legs. The two on the far end away from the hinges should be drilled about 1” away from each side, as marked here; you will drill a ¼” hole through both plates:

The next picture shows the four dowel holes drilled through both plates.

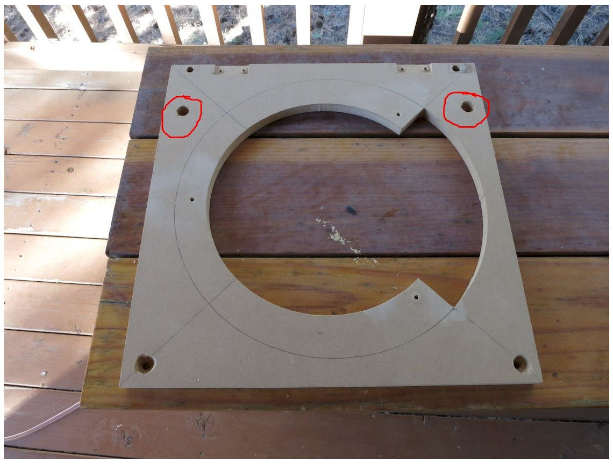
The holes furthest from the hinges are in the correct positions; the holes nearest the hinges were drilled incorrectly. You want these near-hinge holes to be about 1” from the side, but about 2.5” from the back edge. This is to give







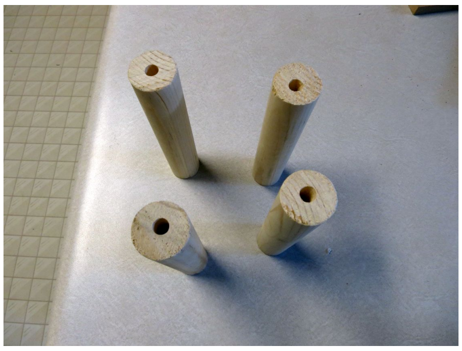
room for the square braces necessary to hold the top plate vertically in one mode, and create a hole to hold the square brace (see the last part of the video for an example of this). Here's the bottom plate with the circled near-hinge holes in the correct position (ignore the other holes for now):

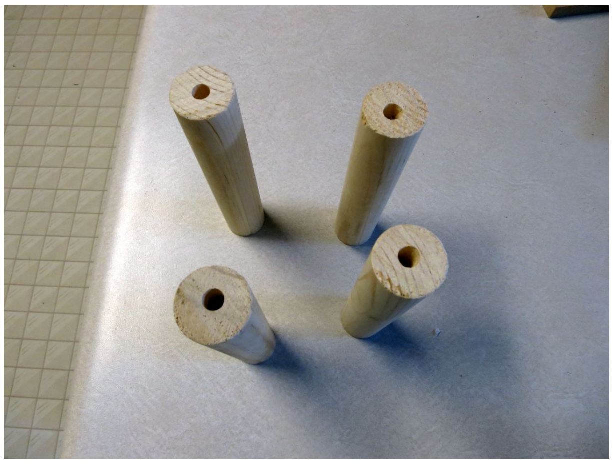


Once you've drilled these holes, unscrew the bottom plate from the hinges; you can leave them attached to the top plate. Now take the 4 6"-long dowel pieces, and drill a hole in the center of one end for the threaded brass insert. For these dowels, I used the brass insert type on the left (externally threaded), which requires a 5/16" hole in the dowels; you can also use press-fits, like the one on the right, which I used for the simple stand:

It's best to drill a starter hole first, then work your way with increasingly larger drill bits to the 5/16" size:

The slotted end of the threaded insert goes down into the dowel hole first. Do a Google search for recommendations on the best way to install these. The way I used was to take a 1.5" ¼-20 bolt, thread a nut on it first, then screw

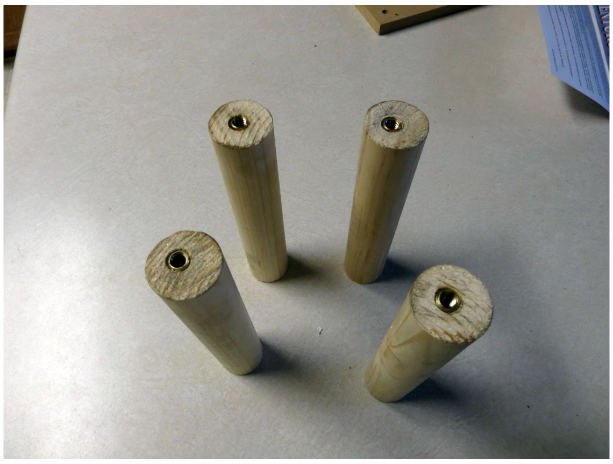




the insert on the end of the bolt until it touches the nut. Clamp the dowel securely, then use a socket driver to start threading the insert into the hole, using firm downward pressure. Once you get it started, use a wrench on the nut to continue screwing the insert into the dowel. Once the insert is fully in the dowel, hold the bolt steady with a socket driver, and use a wrench to loosen the nut. You should then be able to unscrew the bolt/nut with the socket wrench.

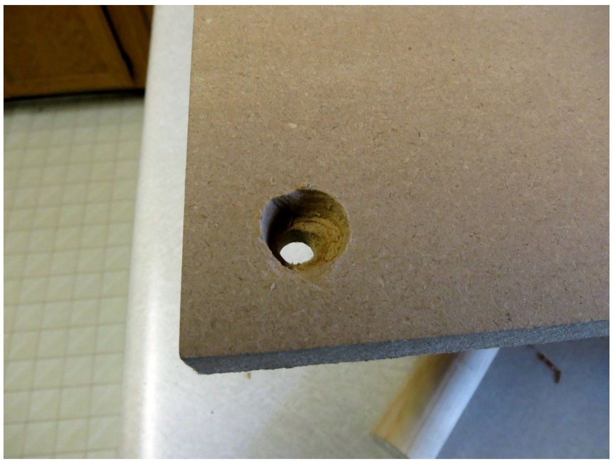
If you only use a bolt, no nut, what will most likely happen is that you will screw the insert in all the way, but it will get stuck on the insert; when you try to unscrew the bolt, the insert will come out of the dowel, and you'll find it difficult to impossible to remove the insert from the bolt.

So here are the four dowels with the inserts in them:

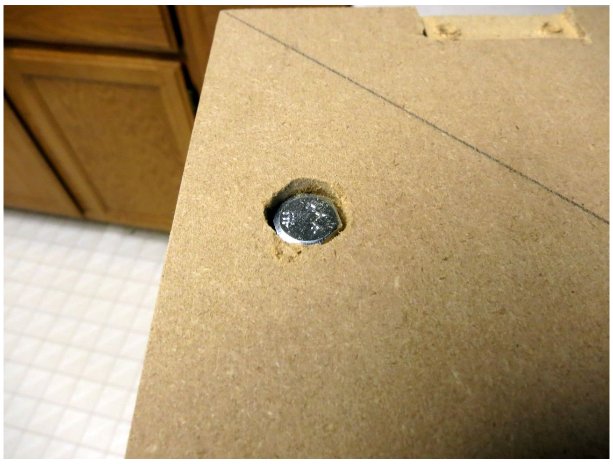


You can attach the dowels legs now to the bottom plate with ¼-20 bolts, as with the simple stand, but the heads of the bolts will stick out and keep the top plate from being flat against the bottom plate. To get around this, drill ½" countersink holes in the bottom plate that the bolt heads will fit into:

I used a ½" drill bit to create these, which is why they look so horrible. You can buy ½" countersink bits, or you



can ask your friendly neighborhood woodworking expert to make them for you. The ¼-20 bolt heads should now fit inside these countersinks, flush with or below the surface of the bottom plate:



Repeat the process with all four holes in the bottom plate, and check that the dowels fit:

Next, you'll be drilling the holes needed to attach the dome to the top plate. Start by marking the positions for four holes on the top plate; you'll want them spaced about halfway between the edge of the plate, and the edge of the hole (shaded a bit towards the outside of the plate). Measure carefully to make sure they're centered relative to the edges of the plate:

**Note:** The dowel holes near the hinges are in the wrong position here – they got filled in later on, and the correct holes drilled).

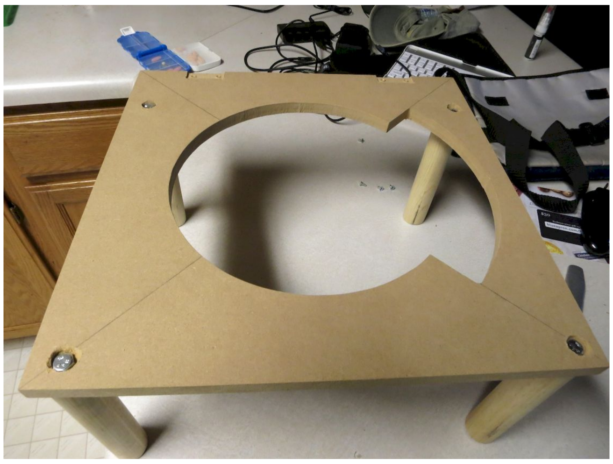
Now drill holes at the marked position for #10 machine screws/bolts:

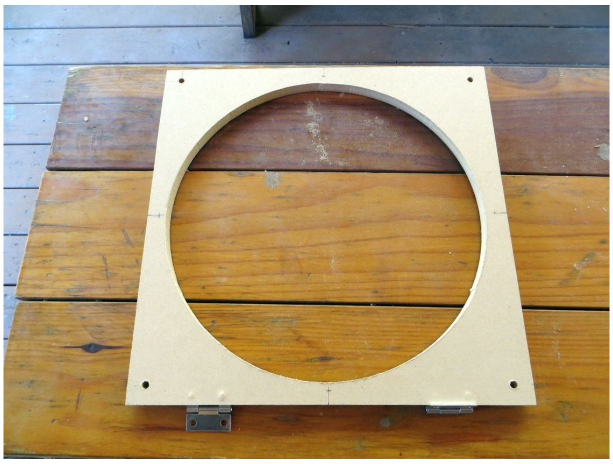
Now mark a spot on the flange of the dome so that's it centered with the camera mount holes at the top, near the cable end, as shown below. It needs to be in this location to minimize the movement of the cables as you tilt this top plate up and down. It will also make lining a camera up in the correct orientation easier in some use cases, e.g. when you use a camera on a tripod:

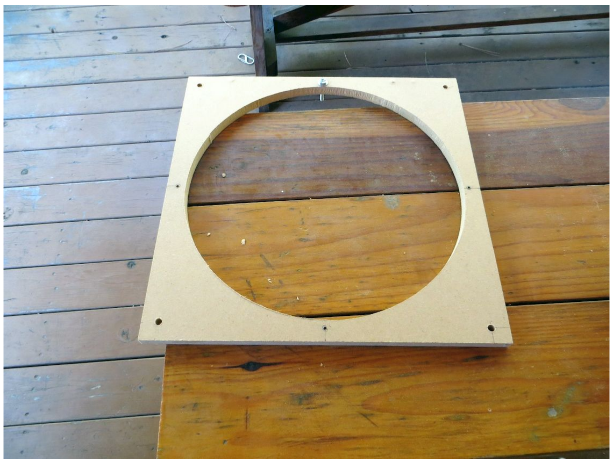
You want to make sure to position the marked hole so that when that hole lines up with a hole in the top plate, the dome can be correctly centered on the plate, with no part of it overlapping the edge. Wouldn't be a disaster, but wouldn't look good.

Drill a hole at the single marked position for the #10 machine screw/bolt, using first your 1/8" acrylic drill, then a step drill:

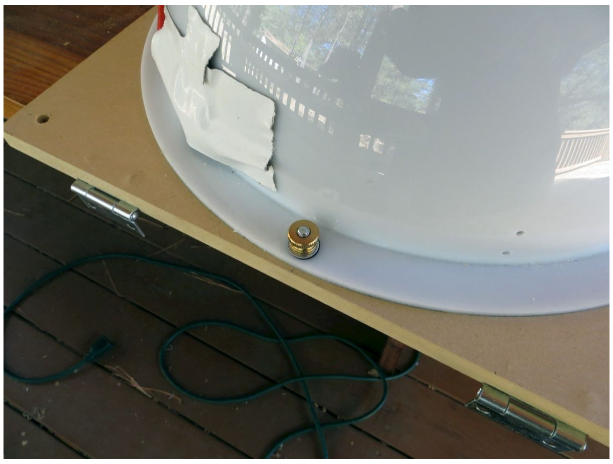
Take a 1" #10 machine bolt, run it through the hole in the top plate on the hinge side, and put the dome on the plate with the bolt going through the drilled hole; attach it with a knurled brass nut, on top of a neoprene rubber







washer and metal washer. Attach the dome securely with the nut, but don't tighten it too much; acrylic doesn't like that:



The dome should be centered on the top plate, with no part of it going over the edge of the top plate. If it does overlap the edges a bit, you can drill out the dome hole slightly larger to allow you to position it correctly. The knurled nut is visible on the left, on the hinge side:

Tape the dome in place with masking tape, to hold it securely:

Push and twist a drill bit through the three remaining holes to mark the positions on the dome flange where you need to drill holes:

Drill holes in these position for the #10 screws, then insert the remaining bolts/knurled nuts to see if everything lines up. If not, you can drill the holes a bit larger:

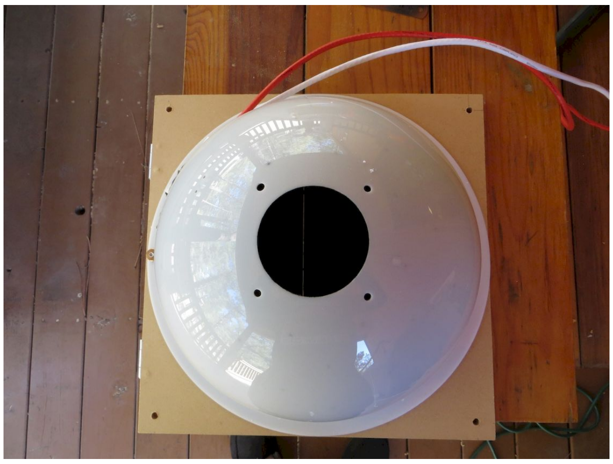
For mounting the dome vertically, it can be useful to have the dome oriented 90 degrees to the right of the current configuration. Try rotating the dome 90 degrees and seeing if the screws still line up; if not, you can drill them a bit larger:

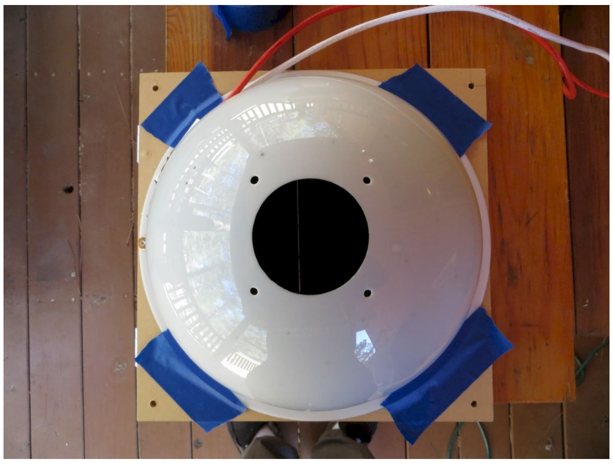
Finally, the #10 screw/bolt heads on the bottom of the top plate will keep it from sitting flat on the bottom plate. So, like the dowel bolts, you'll need to drill countersinks on the bottom of the top plate to accommodate the heads:

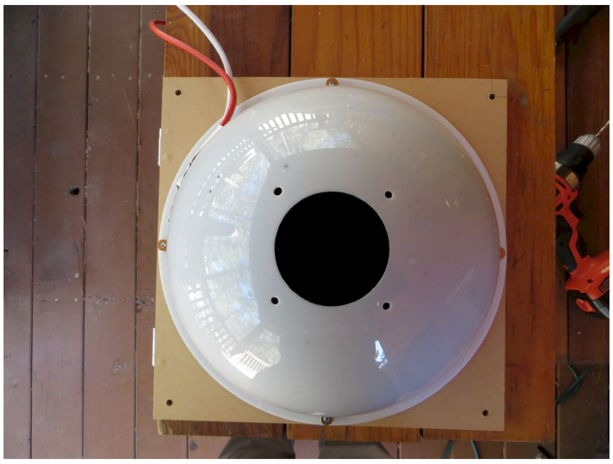
Next is the center plate. Grab the bottom plate, the round disk, and the center piece with the radial notch cut out. Glue the round disk to the center of the notched piece:

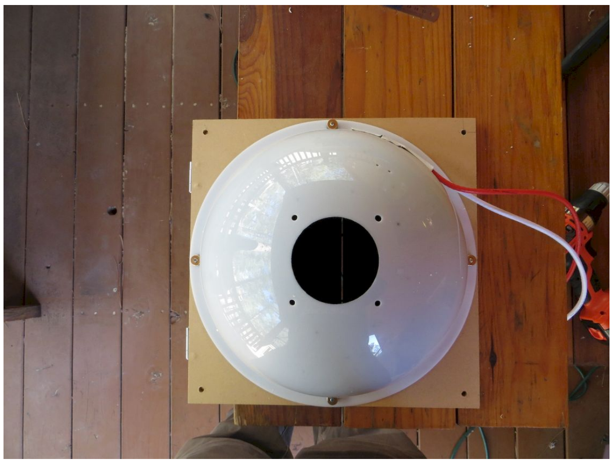
Take the three mending plates, and attach them to the center plate at the three positions seen below with 3/8" #6 wood screws (pre-drilling the holes will help):

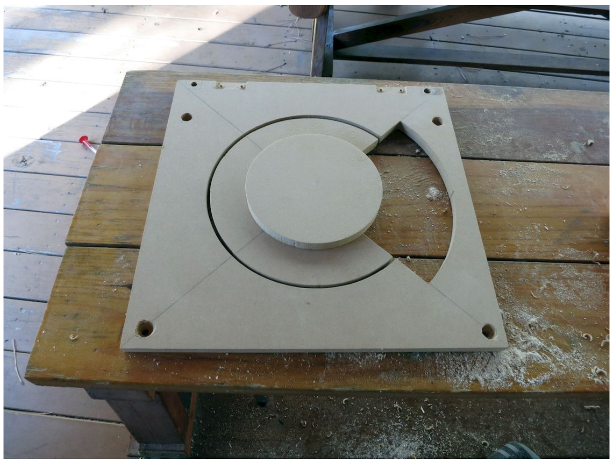
Mark a hole position for one of the three mending plates, then drill a hole for a #8 threaded brass insert in the bottom plate. Install the insert (in a similar manner to the dowel inserts), then attach the mending plate to the bottom plate using a ½" #8 machine screw:

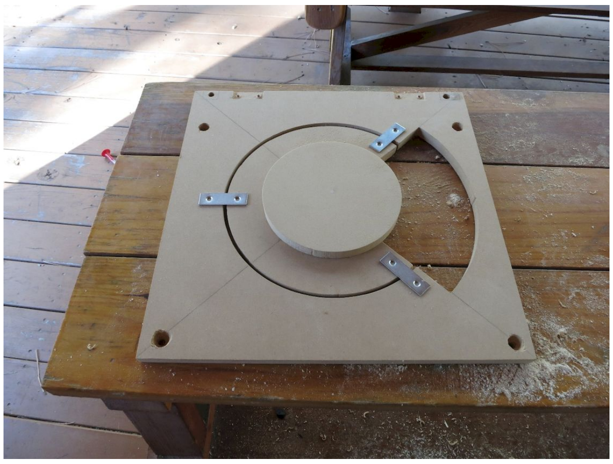


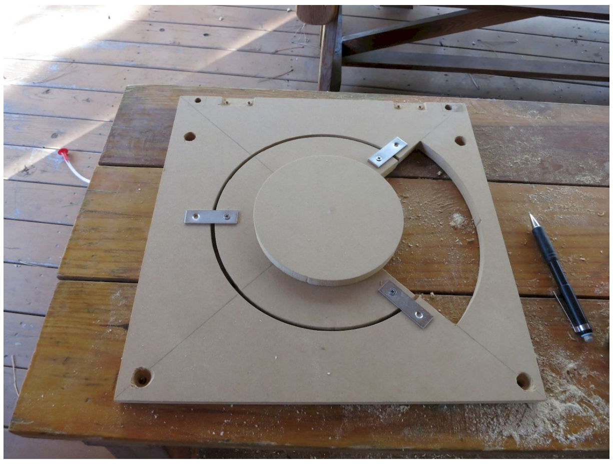


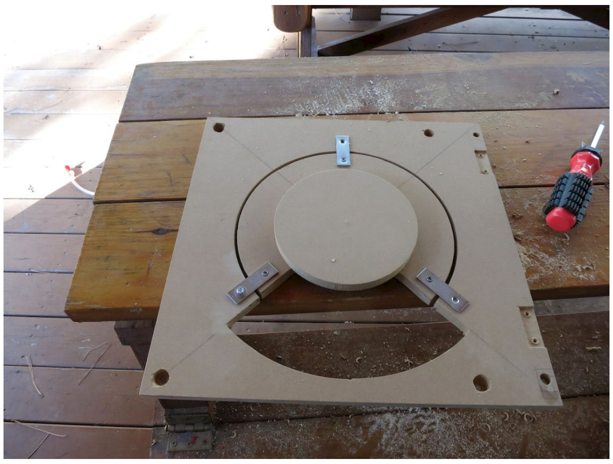




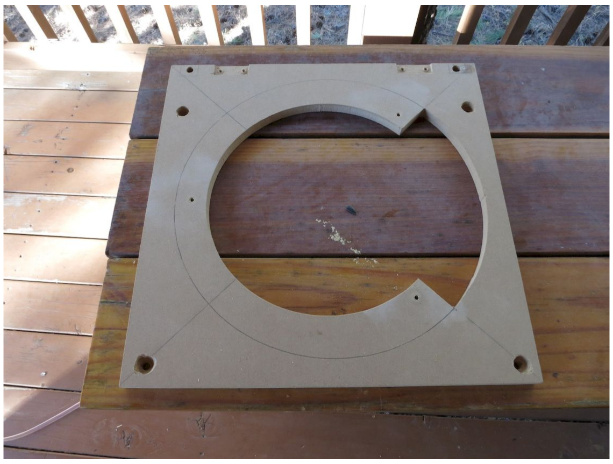








Center the plate as best as possible, tighten the screw, then mark hole positions at the other two mending plates. Drill holes and install threaded brass inserts at the other two holes:



Then replace the center plate and install all three screws to make sure they fit, and hold the center plate securely:

The next section describes installing the braces that allow use of the dome in a vertical orientation. Even if you don't think you'll need to use it that way, I still recommend setting up the dome to allow vertical use – you never know what your system might ultimately be used for.

First, re-attach the bottom plate to the top plate at the hinges:

Note here that the dowel holes are in the correct position (and you can see where the incorrect holes were filled in with wood putty).

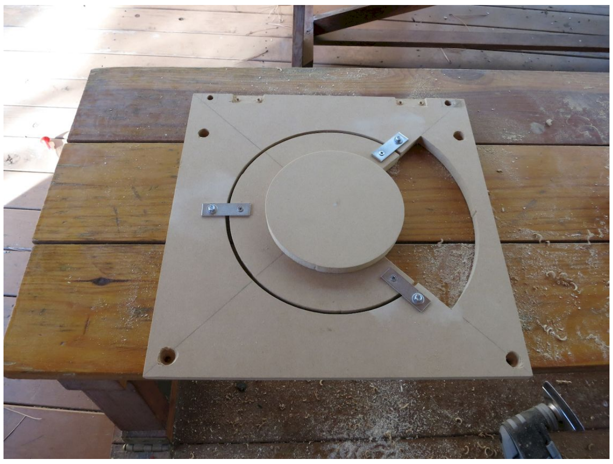
Now you'll need to position and size two square wooden braces (3/4" to 1" square) to hold the top plate (with the dome attached) vertically. Here I have two braces positioned:

In the picture above, I have the bottom of the braces flush with the edge of the bottom plate. This was wrong – it should be lined up in the center of the gap between the top and bottom plates. As a result, the braces were too long, and the dome wouldn't fit until I ground off one of the edges. You want the brace to be long enough to overlap the dome hole on the top plate plus a bit more, but not so long as to block the dome. About 2-3/4 or 2-7/8" is about right. You can check this by placing the dome on the top plate, lining it up with its mounting holes, then making sure the braces clear.

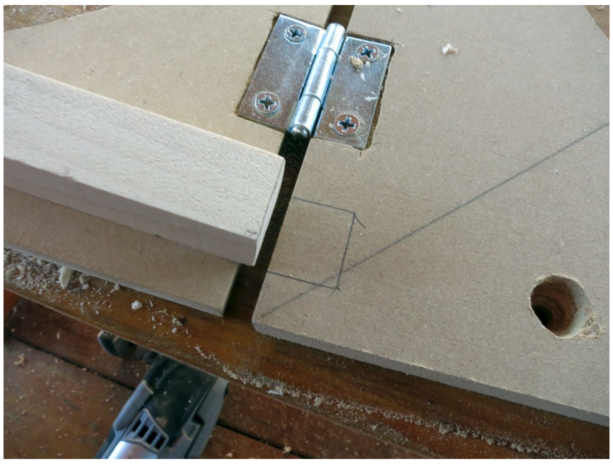
Take one brace, and put it in position relative to the top plate held vertically, and the dowel hole drilled in the top plate; clamp it in place:

Drill a 5/16" hole through the dowel hole in the top plate, extending all the way through the brace piece on the other side:

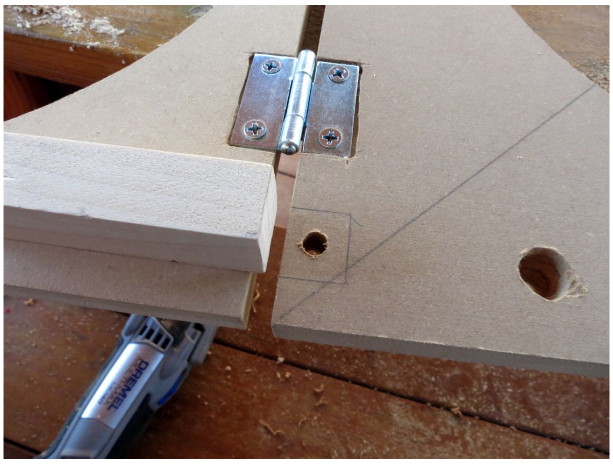
Insert a ¼-20 threaded brass insert into the hole in the brace, then bolt the brace into place:



If it doesn't fit neatly in place (or even if it does), you can drill out the hole in the top plate to 3/8" diameter to allow you more wiggle room in positioning the brace. Now trace the outline of the brace on the bottom plate:



Drill a 5/16" hole in the bottom plate, roughly where the center of the brace will be:



Flip the top plate up until the bottom of the brace is against the bottom plate, then drill a 5/16" hole into the bottom of the brace using the hole in the bottom plate as a guide:

Install a ¼-20 threaded brass insert into the hole at the bottom of the brace. Bolt it back in place on the top and bottom plates to make sure it fits; you can drill out the bottom plate hole to 3/8" to give you more wiggle room in positioning the brace vertically:

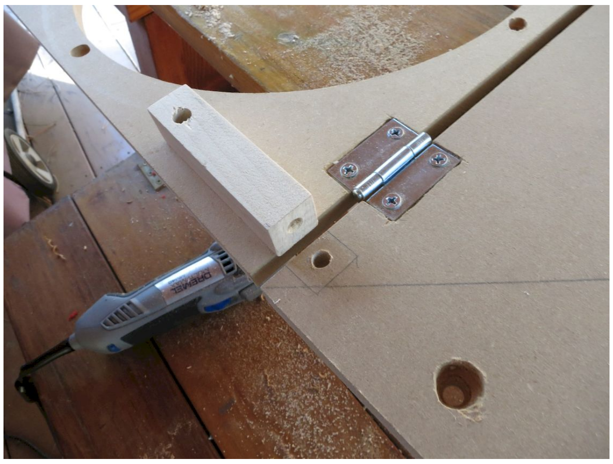
Mark the bottom of the brace to identify its position (I put "L" to mark this one as the left brace).

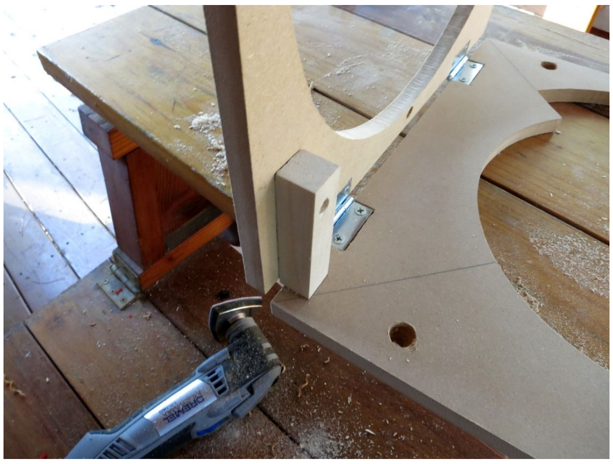
Repeat the same procedure to install the brace on the other side.

Almost all of the parts assembly is done; now comes painting. Disassemble all parts, including the braces and hinges. The only parts that absolutely need to be painted are the bottom and center plates; these should be painted flat/matte black to reduce light scattering inside the dome. However, I recommend painting the entire stand, because it looks better, and also protects it from moisture. You can choose whatever colors you feel are appropriate. I used black for the entire bottom/center plate assembly, since that made it easier. For the rest of the dome stand, I used white, since it's intended for both indoor and outdoor use, and white will help keep it cooler in the sunlight. If you only plan to use it indoors, use whatever colors you want. Once the paint dries, attach dense rubber padding to the bottom of the dowel stand legs, for vibration reduction and to keep the system from sliding around. Sorbothane is awesome, but expensive; Google for other vibration dampening options. Your system should always be used in a vibration-minimized environment, e.g. a sturdy heavy table sitting on a concrete floor.

Now re-assembly your stand:

1. Attach the hinges to the top and bottom plates.
2. Attach the dowel stand legs with 1" ¼-20 bolts.





3. Attach the center plate to the bottom plate with #8 screws.
4. Attach the dome to the top plate with #10 screws, neoprene and metal washers, and brass knurled nuts.

For vertical mode:

1. Remove the dome from the top plate.
2. Mount the top plate vertically with ¼-20 bolts and washers.
3. Mount the dome on the back side of the top plate.
4. While the system will be fairly stable in vertical configuration, I recommend keeping the center plate in place, and putting large books or other heavy weights in place to make it more stable.

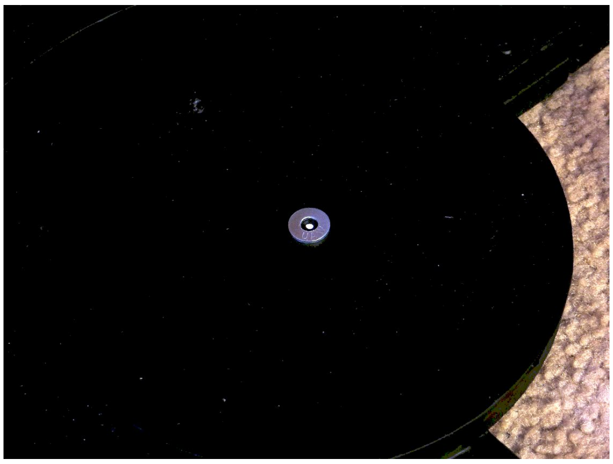
If any part of assembly is unclear, use the video as a guide to figure out where/how everything fits together.

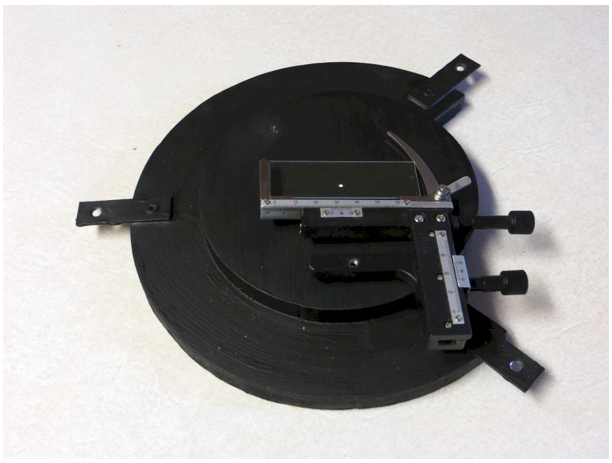
You will want to mark the center of the plate to aid in positioning samples..

Use the camera installed on the top of the dome. I put a small washer on the center plate, and centered the washer in the camera view. I then marked the center of the hole of the washer with a bit of white paint dabbed with a toothpick:

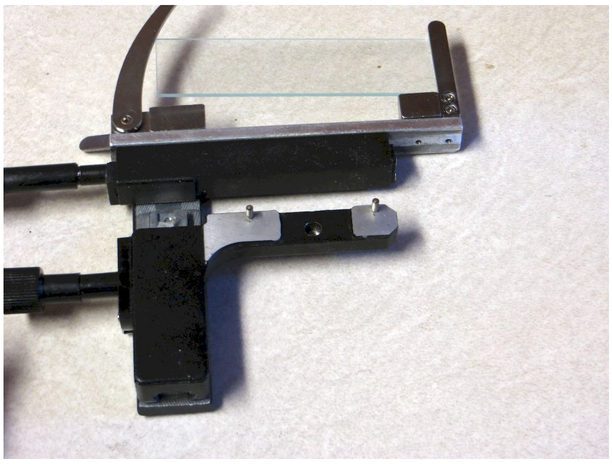
You can place the samples directly on the wooden disk, but I usually put some kind of smoother black/gray material on the disk to serve as a less-textured background.

One final optional step. If you plan to use either a macro lens or a USB microscope with this system, you'll want some way to accurately position your sample in place. I use an inexpensive micrometer stage intended for use with lab microscopes for this. Position the micrometer X-Y screws so that the stage is in the center position, put a glass slide into its holder, and place it on the center plate so that the center of the glass slide is at the center of the center plate:





Mark the position of the center screw hole, and drill a hole for a #4 machine screw. On the underside of the micrometer stage, there are two positioning pins:

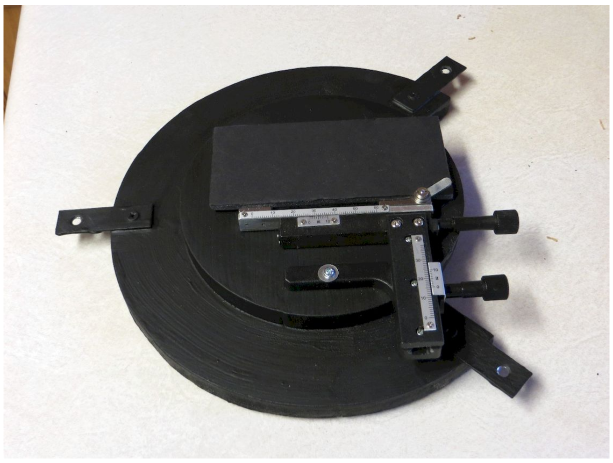


You'll need to drill holes in the center disk to accommodate those. The simplest way is to attach the micrometer stage with the screw to the center plate, lining it up as in the second picture above. Then press on the micrometer stage so that the pins make minor indentations in the center plate. You can then drill holes at the indentations that the pins can drop into, and install the micrometer stage:

While you can attach small samples directly to a glass slide in the micrometer stage, I use a piece of black posterboard with a glass slide glued underneath to make a larger sample stage:

The glass slide fits in the micrometer stage, and the entire posterboard piece moves with it.

Finally, the 1.5" ¼-20 bolts with washers and wing nuts can be fastened through the holes running through both



the top and bottom plates to keep the system secure during transportation.

## 12.3 Mounting the camera

When acquiring RTI data, you need to have the camera pointing straight down at the object being photographed, through the hole at the top of the dome. Which brings up the question, how do you mount the camera above the dome so that it's fixed, rigid, and pointing straight down? Here are some possible ways, some of which I've tried, some of which I haven't (but which could work). One of these may work for you, but you may also have to come up with your own unique camera mounting system.

### 12.3.1 Solution 1 - a tripod

Use a tripod that lets you mount the head on the bottom. Like this model, and many others:

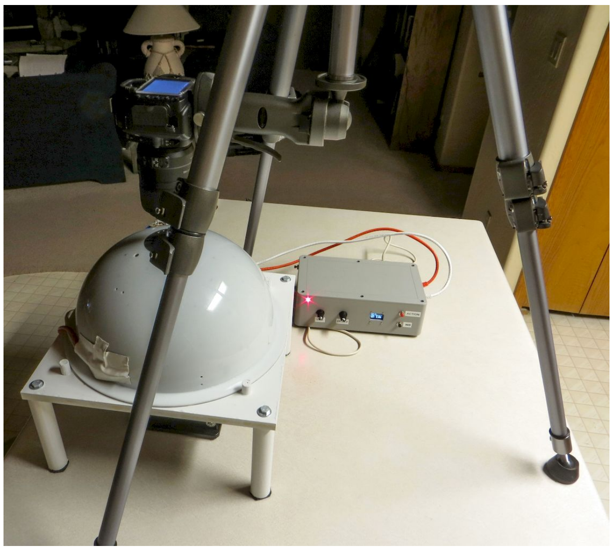
Orient the camera so that it's facing straight down, and position it so that it's looking through hole at the top of the RTI dome. I use this method with my smaller domes when I have to shoot with a larger camera like a DSLR:

Works fine: video clip.

Although the pistol grip on this tripod is pretty weak – it's just barely able to hold up this Nikon D90. If I buy another tripod like this one, I'd choose a standard tripod head that should be more stable (as long as I can adjust it to let the camera point straight down.

Advantages:

- You may already have a tripod that can do this.



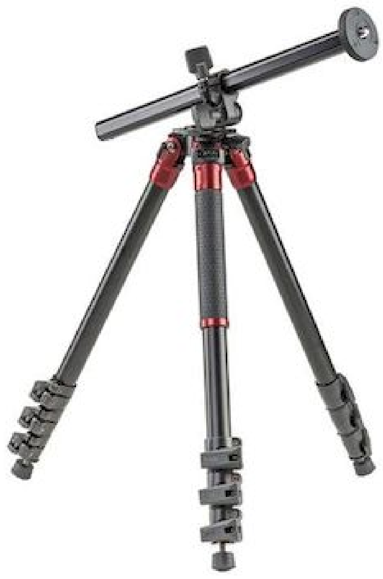
- Strong enough to support large cameras.
- Flexible enough to support a variety of camera sizes (compact to DSLR).

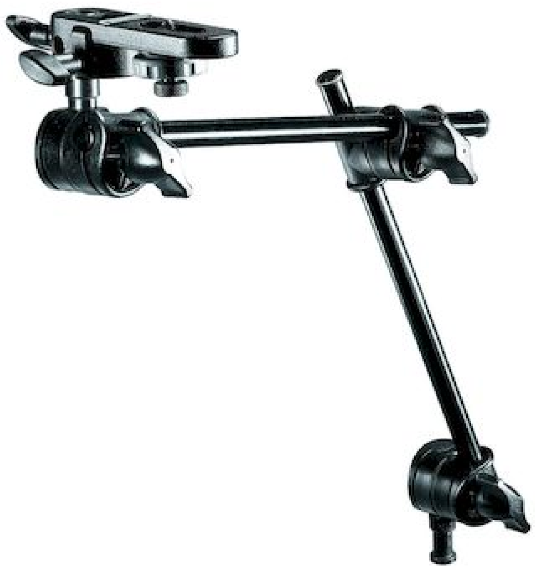
Disadvantages:

- If you don't have a tripod like this, you'll have to buy one.
- Requires a fair amount of table space even for a small dome, and more for a larger dome.
- Not practical for very large domes (> 24" in diameter)

### 12.3.2 Solution 2 - a pivot tripod

Use a pivot tripod or tripod extender like these guys:

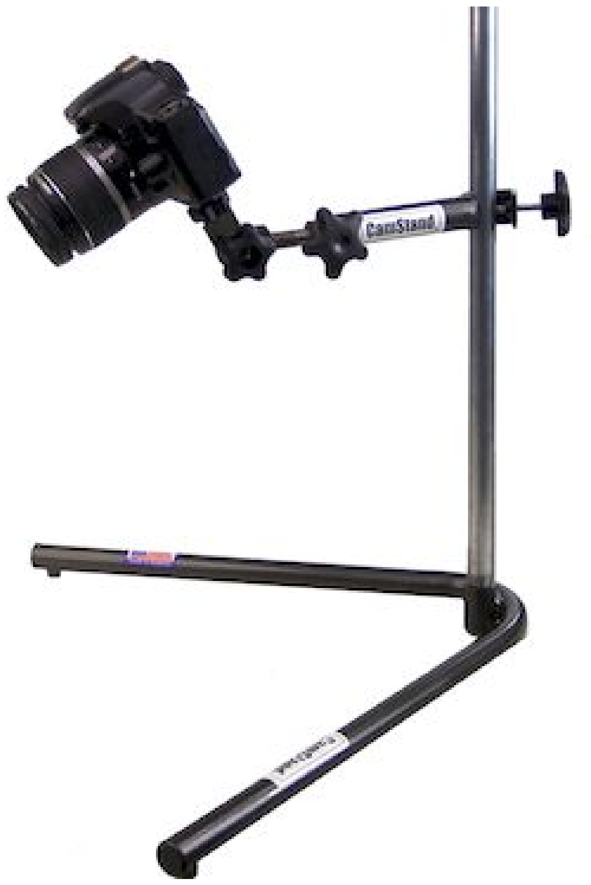




Haven't used either of these, so can't comment too much on advantages/disadvantages. Seems to me that the extender might be susceptible to vibrations that might blur the image, and the pivot tripod might have problems supporting a camera if it's extended too far (unless you can add a counterweight).

### 12.3.3 Solution 3 - a camera stand

Here's one you can buy off the shelf:



Haven't tried this one directly, although a friend used a similar concept on a dome I built for him. Seems like a decent solution, although you may need to modify your dome stand to get it to fit. Another option would be to adopt the "pole" part of this design and attach it directly to the stand.

### 12.3.4 Solution 4 - PVC stand

½" PVC pipe – is there anything it can't do? Here's one of the earliest camera stands I made for an 18" dome, out of PVC pipe glued together (with foam rubbed on the bottom to prevent movement and vibration). The camera is held in place with a ¼-20 screw in the back of the T-junction. The white unpainted pieces of PVC are not glued in place, so that I can rotate the camera into position.

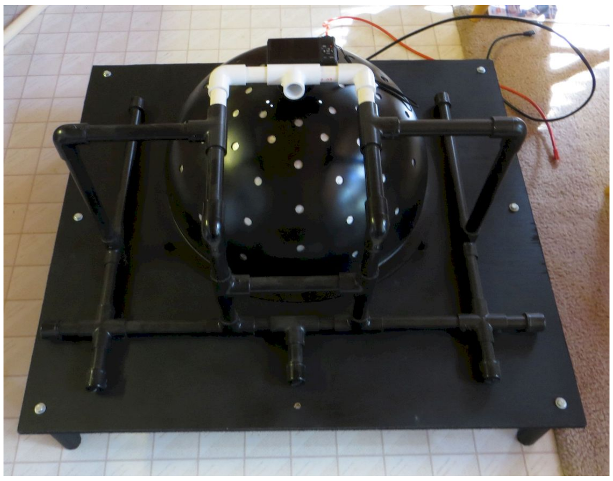
Advantages:

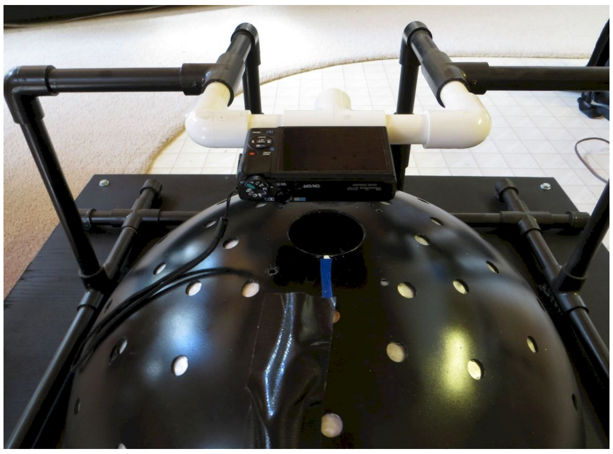
- Cheap, parts readily available.
- Strong enough to hold cameras up to DSLR sizes

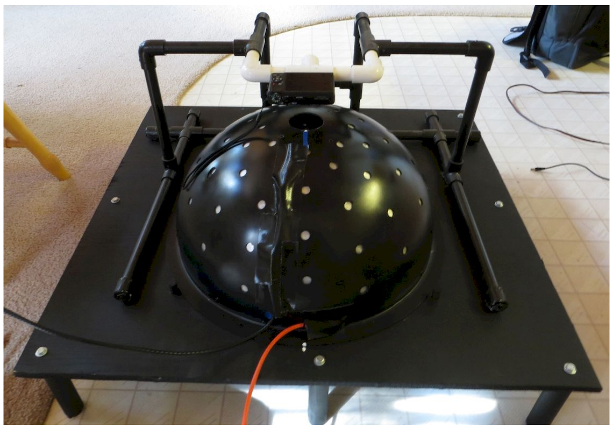
Disadvantages:

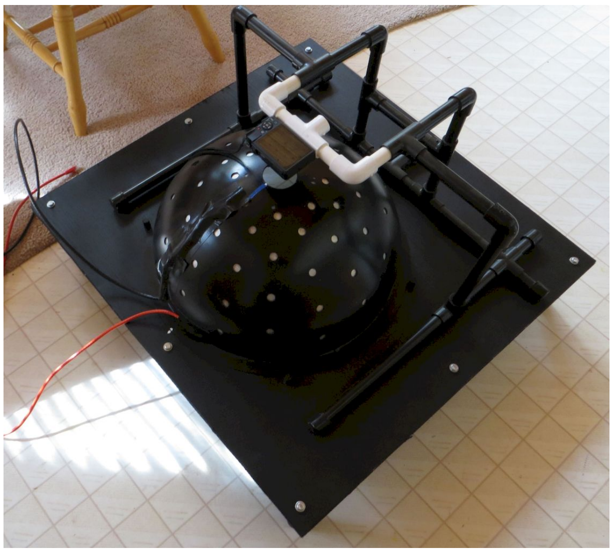
- Requires a much larger stand to accommodate both the dome and the camera holder
- While reasonably steady, you'll definitely need a stable, vibration-free surface to put it on
- Only accommodates one camera size at a time. You can raise it with supports, I suppose, but not sure how stable that would be.

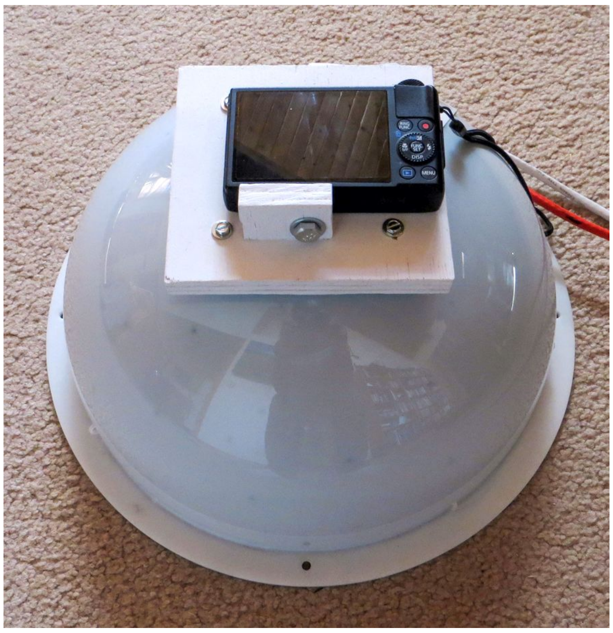
These are just a few ideas – you may have to come up with your own design to accommodate your camera. For most of the photography I do, I use a compact camera, and I've come up with a simple, stable holder design that should work well with any dome size:











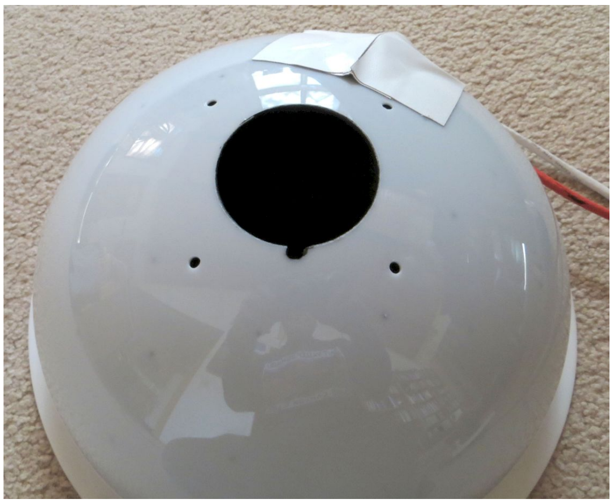
Small, neat, secure, easy to install – perfect for the right camera. What is the right camera? It needs to be a reasonably small camera that will lay flat/parallel on the wooden board when you insert the lens assembly into the hole drilled in the board. Here's my Canon S110 from the front:

Nice flat front, lays very flat as long as the hole is big enough to contain the lens and the ring outside it.

Here's my Canon SX260:

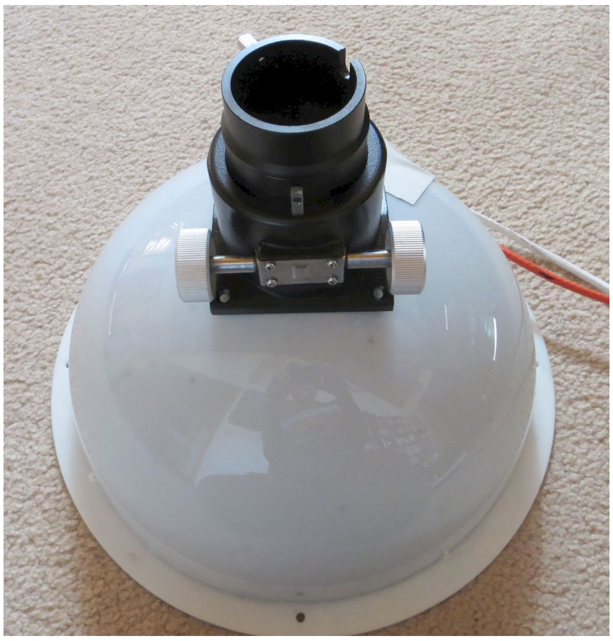
Curved front, which you think might disqualify it. And yet, if I lay it on a board with a hole cutout just big enough for the lens assembly to fit, it also lays parallel to the board, and works great as a camera for the RTI system. So the moral is, you have to try the camera before determining whether it will work with this style of holder or not.

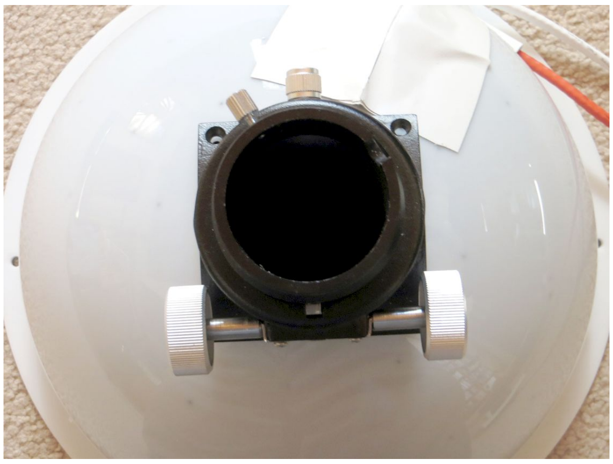
Remove the holder, and the holes that hold it in place become visible:



This hole is 2.5" in diameter, specified so that a telescope eyepiece focuser could fit exactly (the notch at the bottom of the hole was cut for the focuser gear train):

Why am I mentioning this? Because I used the eyepiece focuser as a template for drilling holes into both the dome, and into the wooden camera holder. The focuser is intended for use with a USB microscope, for high-

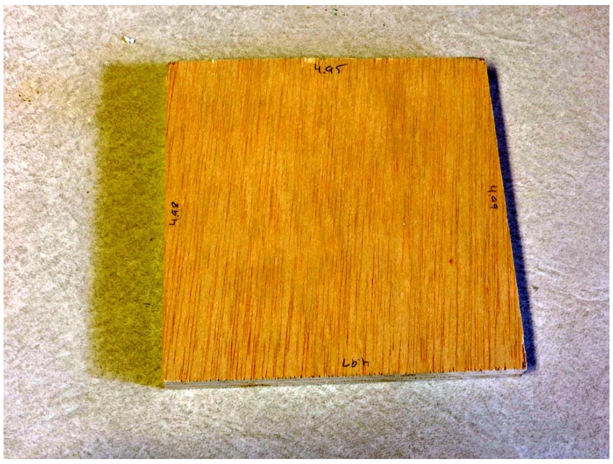




magnification RTI work. What if you don't have a focuser? Or, in my case, what if you had the hole drilled to a size other than 2.5" in diameter, like the 3" for the dome I'm using for this build? How do you position the holes to be drilled on the dome, and how do you get the holes on the wooden board to match up?

The following is my solution. If you think of a better one, let me know.

Start by getting a board cut to square dimensions, large enough to hold the camera when the lens assembly is centered on the middle of the board. I have a board already cut to about 5" x 5" (a bit less):



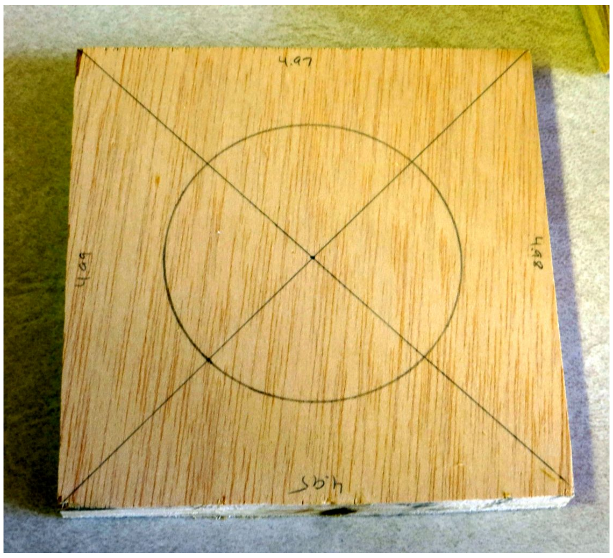
and my Canon S110 will fit on that:

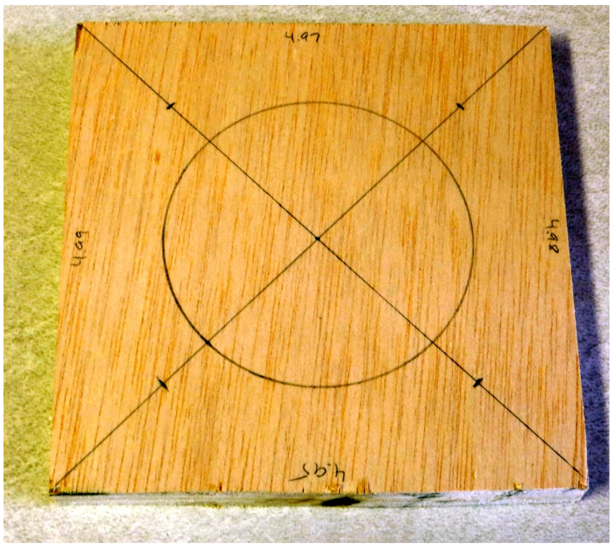
Draw lines between opposite diagonal corners, and the center will be where they intersect. Draw a circle around the center that has the same diameter as the hole in your dome, 3" in this case:

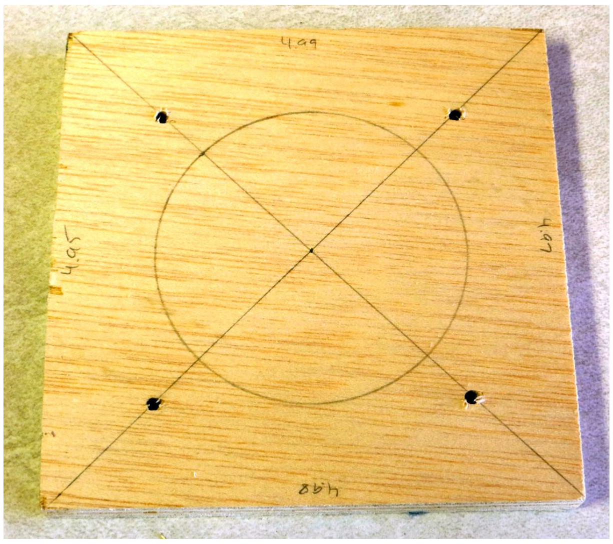
Now figure out where to space/place the holes. In my case, I want to space the holes with the same distances as the screw holes for the telescope focuser, so that I have the option of using the focuser with this new dome. The holes are spaced about 74.5mm apart on the focuser, so using Pythagorean geometry, I calculate that the holes should be about 52.7mm away from the center of the block (sqrt(2*74.5^2)/2). I mark drill hole positions at those distances from the center:

In the likely case that you're not worried about the option of installing a telescope focuser, you can pick any radius outside the hole and inside the board to mark the positions. But be careful not to put them in a location where you might unintentionally drill through an LED or some wiring in the dome. The distance I'm using should be pretty safe for most domes, but check to make sure. Use a nail or small drill bit to make a precise starter hole, then drill a 1/8" hole through each of the drill marks:

If you have a hole saw the same size as the hole in your dome, you can now use it to drill out a hole in this board. Aligning the hole in the board with the hole in the dome, and taping the board in place (see later on in this

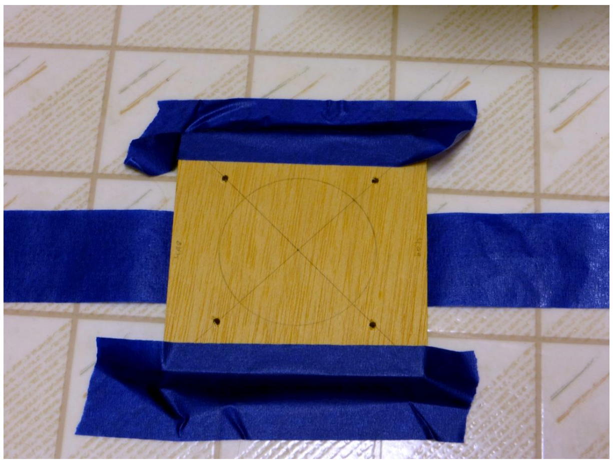

4.97
4.99
4.98
4.95


4.97
4.99
4.98
4.95

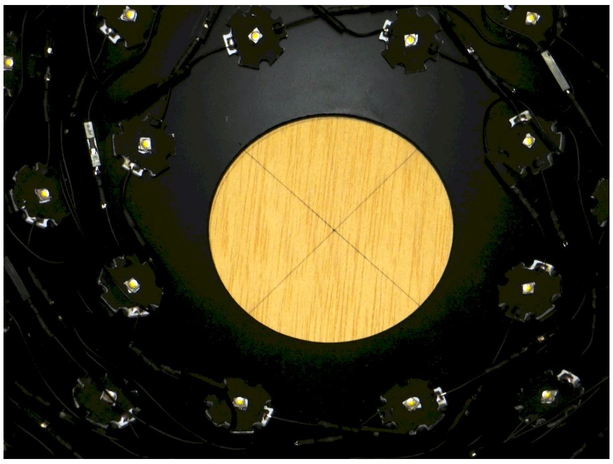

4.99
4.95
4.97
4.98

instruction step), you can use this board as a jig for drilling holes in the dome.

But, if like me, you don't have a hole saw big enough, you have to go through a few more steps. First, lay some blue painters tape on the floor sticky side up, then place the board on that tape. Use more tape to fix the board to the floor:



Now place the RTI LED dome upside down on top of the board, so that the board is visible through the hole. Move the dome around on the board until the drawn circle is aligned evenly with the edge of the dome hole:



In the above picture, there's still a bit of the drawn circle visible on the top left, but none on the bottom right, so it's not perfectly aligned yet.

Once aligned, put tape on the inside to help hold the board in place:

Use the first piece of tape, laid sticky side up on the floor, to further fix the board to the dome. Then remove the tape holding the board to the floor, and flip it right side up.

This may look good, but I wanted the board a bit more clockwise, so that the center line between the holes lined up just to the right of the Gorilla tape at the base of the dome. So I repeated the process, and got this:

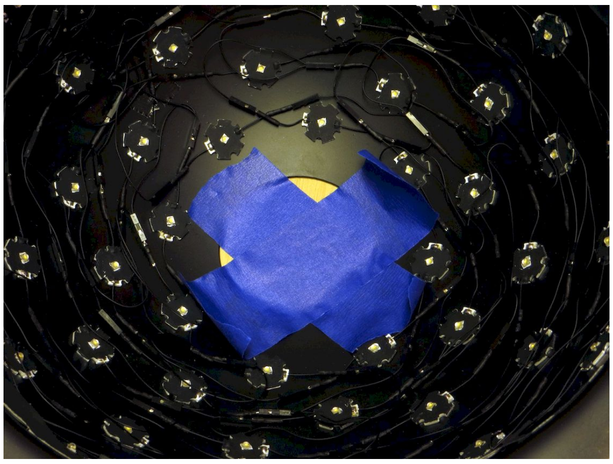
Exactly where I wanted it, and a useful orientation for a more advanced stand I'll be documenting further on.

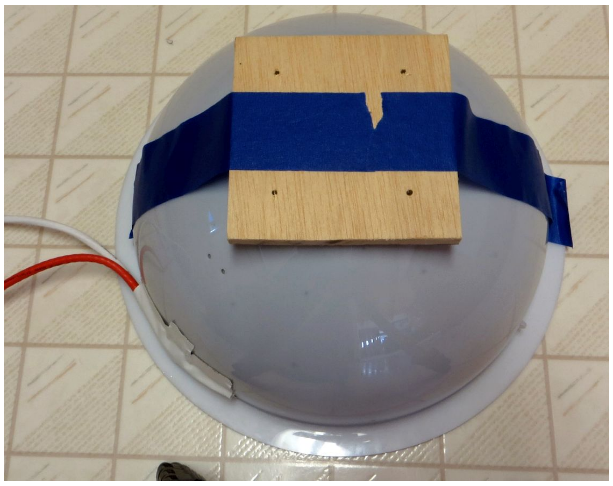
Put more tape on the board to hold it firmly in place, much more than I did below:

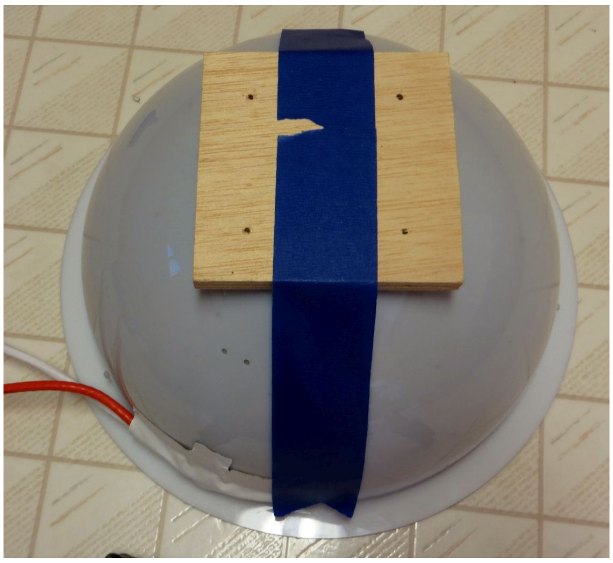
Double-check one last time to make sure the holes aren't above LEDs or wiring inside the dome. Then using your 1/8" acrylic drill bit, use the holes in the board as a guide to drill 4 holes into the top of the dome:

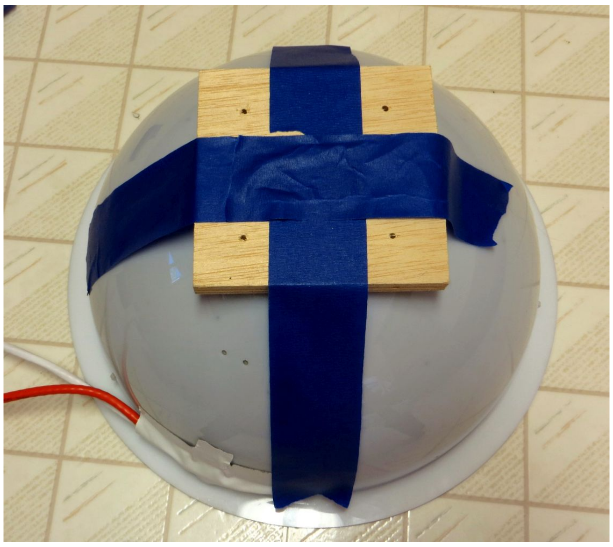
See that double hole at lower left? That's because I didn't have enough tape holding the board in place while I was drilling, and it slipped on the last hole. I had to re-align the board and re-drill the hole in the proper location. I then used a step bit to clean out that double hole:

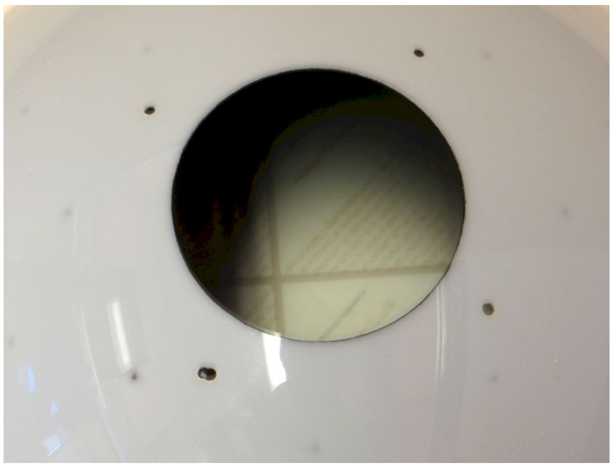
Acid test is, does the telescope focuser fit? Putting it inside the main hole, and putting screws through the focuser and the dome:

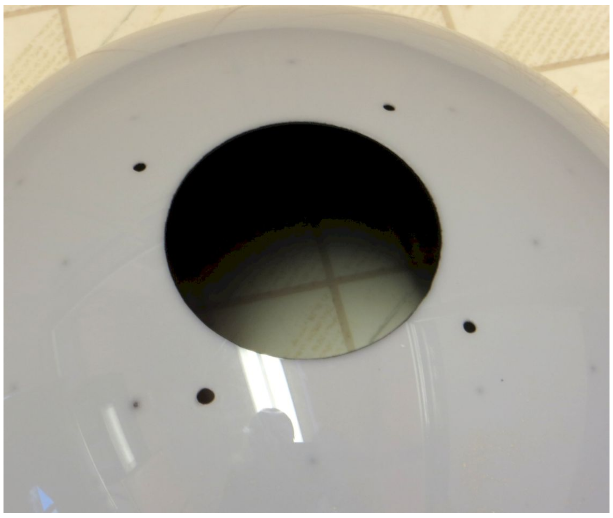


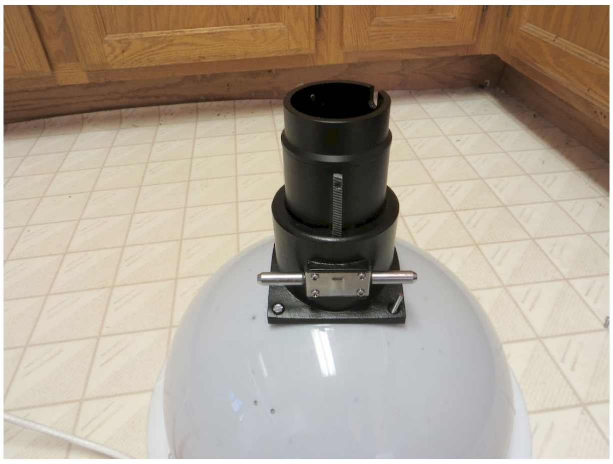






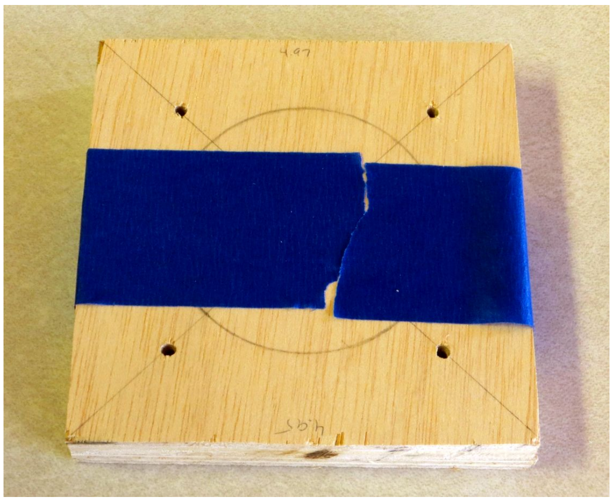






Three of the four screws went in perfectly; the fourth one (visible on the right) was a bit off. Fixing that is as simple as enlarging that hole in the dome a bit, which I did.

You should now mark the wooden board you used for aligning these holes as a permanent template, and only use it for creating new boards. Here I've taped the template on top of another board, using it to mark a fresh set of drill holes:



Use a 1/8" drill bit through the template holes to mark the drill positions underneath, making small pilot marks.

Draw diagonal lines between opposite holes to find a central point.

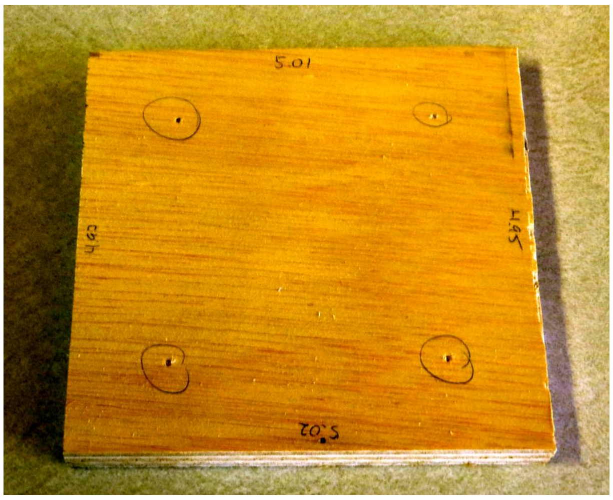
Next, take the camera and measure the size of the hole you'll need to drill in this board for the lens assembly to fit through. I used a pair of calipers, but a ruler should get you close enough:

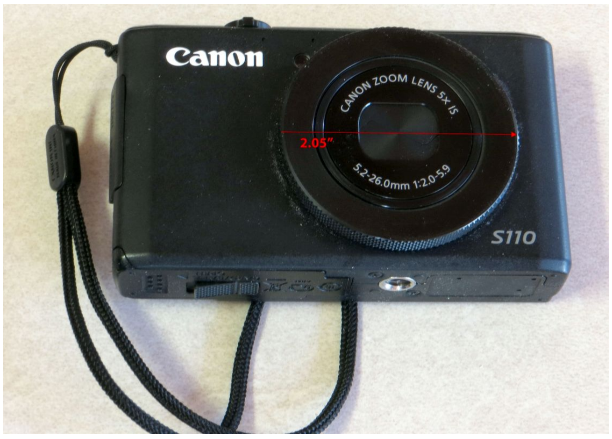
I measured a hole diameter of 2.05", so that's the minimum diameter hole I need to cut in the wooden board for it to fit through. Don't worry if it's a bit big – you'll have additional control over its final position. Use whatever cutting implement you want. I had a hole saw 2" in diameter, and used that; if the hole was a bit small, I planned on sanding it out until it fit.

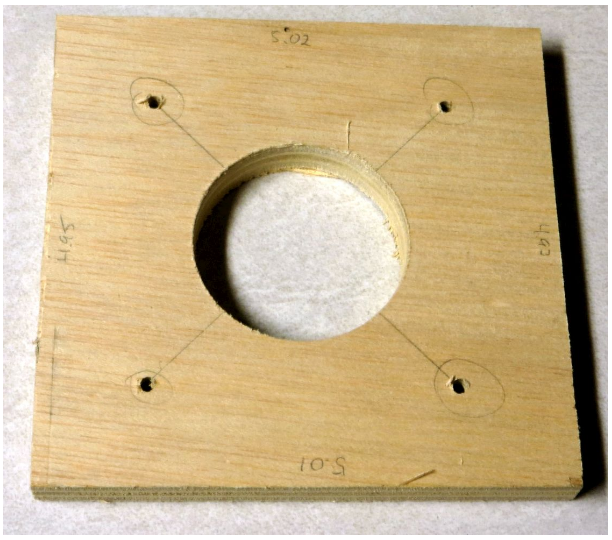
**Note:** When using a hole saw, remove it frequently to blow out the sawdust. If the saw teeth fill with sawdust, it won't cut any further.

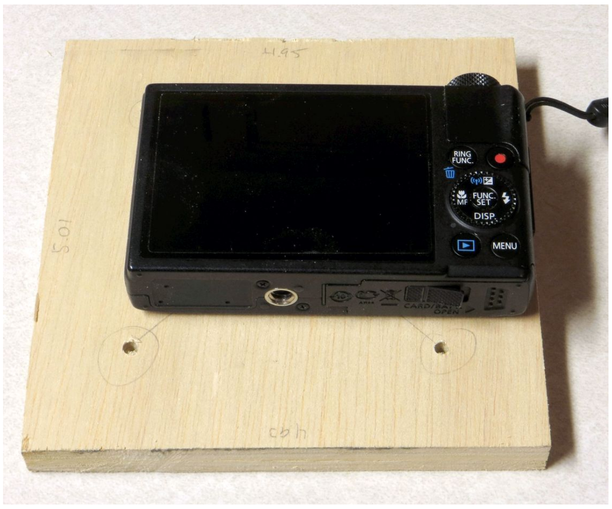
As it happened, the hole saw had enough cutting loss that the camera lens assembly fit perfectly into it without further trimming.

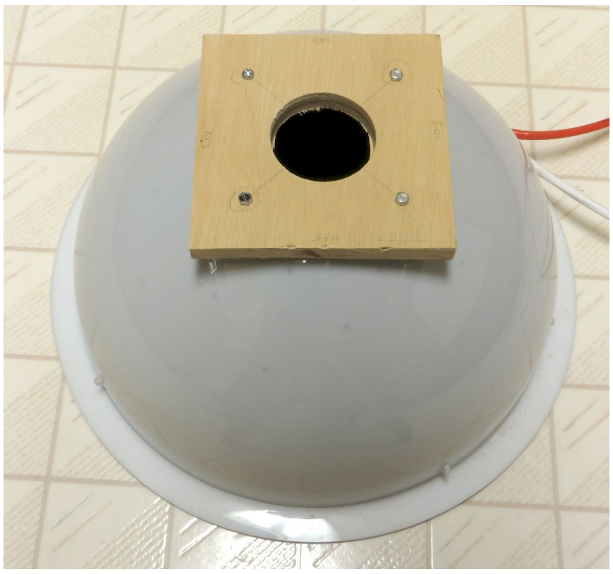
And checking the fit of the board into the dome holes using #6 screws, it fit perfectly as well – no need to enlarge any of the holes:


5.01
4.95
4.93
5.02

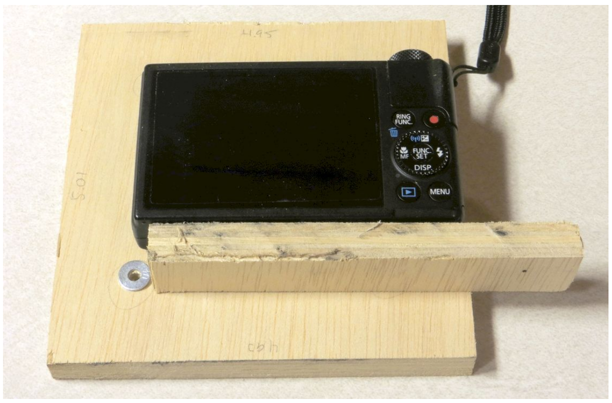

Canon
CANON ZOOM LENS 5x IS
2.05"
5.2-26.0mm 1:2.0-5.9
S110


5.02
4.95
4.93
5.01

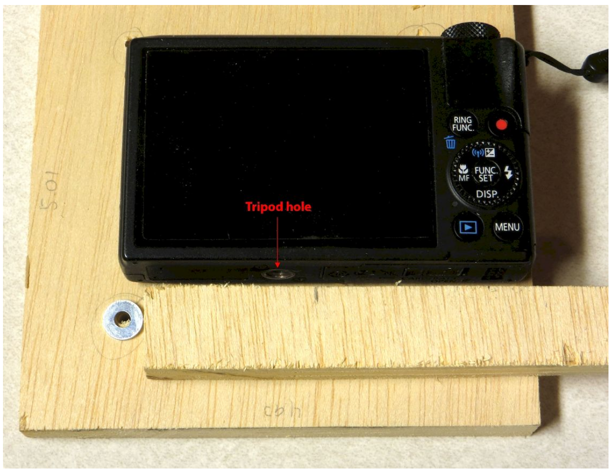




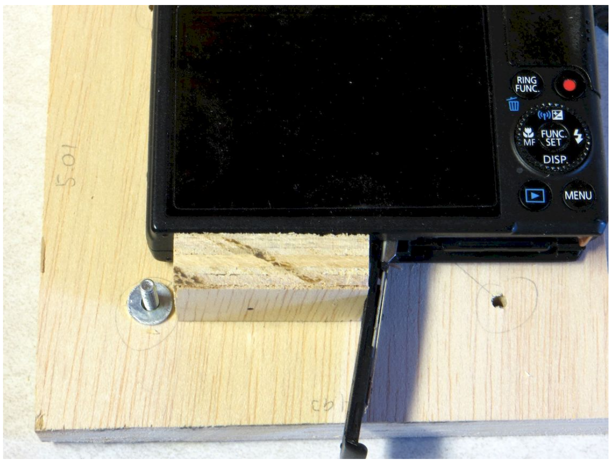
To attach the camera securely to the board, and align it correctly, this holder uses the ¼-20 tripod mounting hole on the base of the camera. Cut a ½”-thick rectangular piece of wood the height of the camera when inserted into the hole (2 cm in this case):



On the left, you want it to clear the washer the screw will go through. On the right, you want it to go at least as far as the tripod hole plus a bit more:

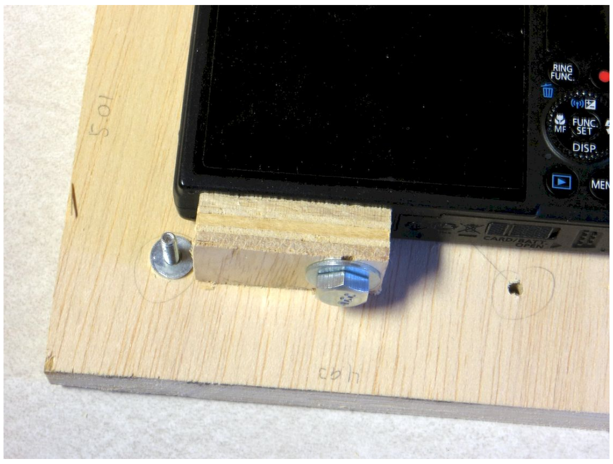


Pencil mark above shows where I cut this piece. Ideally, the piece will allow you to open the base door of the camera to replace the battery or SD card:



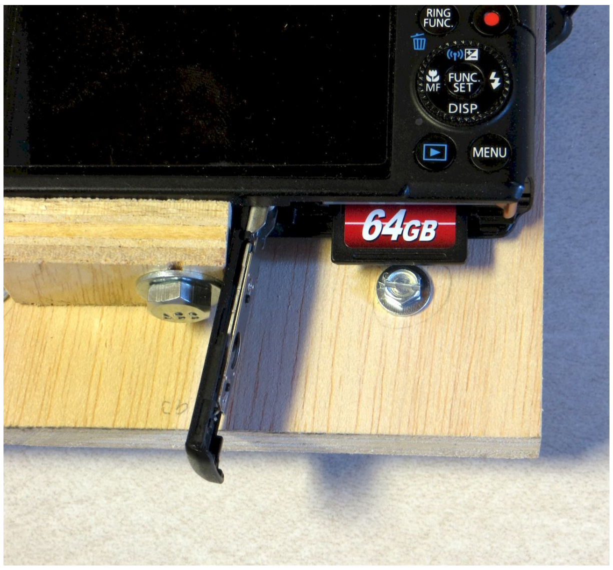
Possible on my Canon S110, but not possible on every camera; my Canon SX260 didn’t allow for this.

In the small block you’ve just cut, measure where the center of the tripod hole is on the camera, mark it on the block, and drill a 5/16” hole there. You can then insert a 1” ¼-20 bolt with washer through the hole, screwing it into the camera’s tripod hole.



Making sure the camera lens is fully inside the central hole, loosen the bolt, press the wood piece until it's flat against the board, then tighten the bolt. Remove the camera/wood piece assembly, put a small amount of general purpose glue (Elmer's, Weldbond) on the bottom of the wood piece, then put the camera/wood assembly back so that the camera is lined up as evenly with the drill holes as possible. Press firmly down, then let the glue dry before unbolting the camera. Not too much glue – you don't want it to ooze out and glue the camera to the board as well, though general purpose glue should peel off the camera in case that happens.

Put the mounting screws in, and check to make sure that the battery/SD card door still opens with the screw in place. For this camera, there's enough clearance:



If it didn't clear, I could always countersink the screw head to lower a bit. In fact, I did that with another screw on the board that slightly blocked access to the zoom control on the top of the camera:

Putting the camera holder back on the dome to check the fit again:

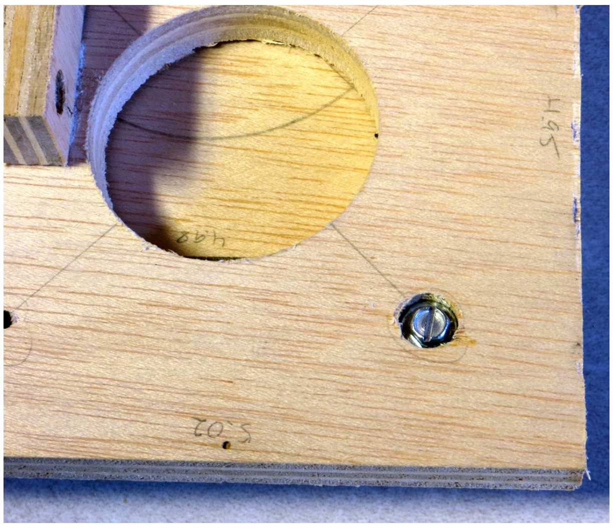
Looks good. The fit is pretty snug with just the screws through the holes, but if I need extra security, I use a neoprene washer, steel washer, then brass knurled nuts to attach it internally to the dome:

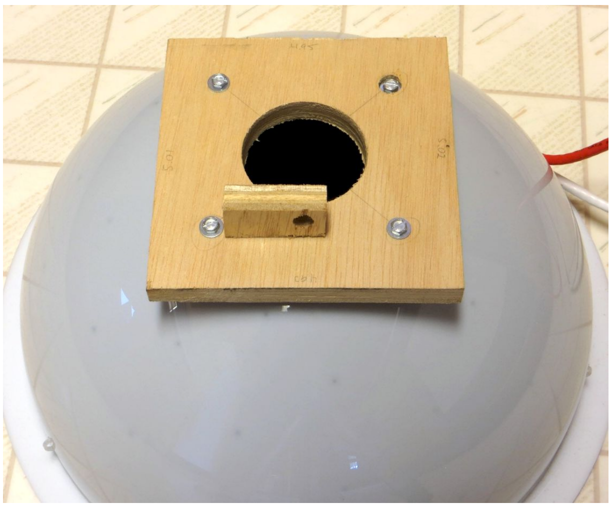
Don't tighten the knurled nuts too much – just enough to make them snug.

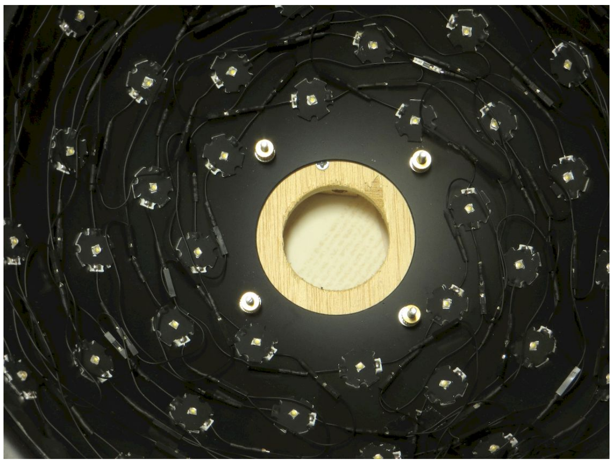
And the camera still fits:

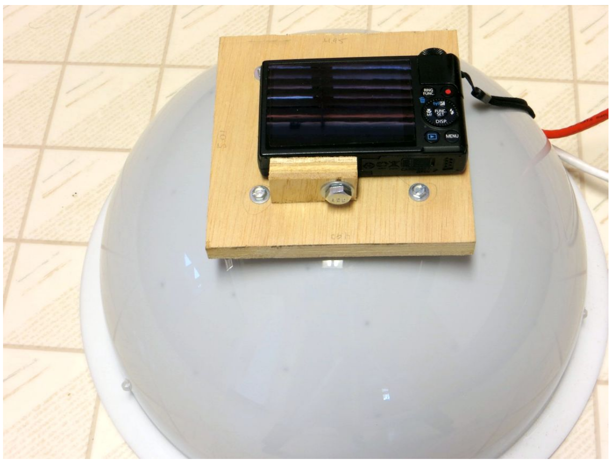
I painted the camera holder later on – white on top to match the dome, black on the bottom. I found that painting the bottom white caused some problems with very reflective objects – they reflected the image of the bottom. Painting the bottom black fixed that problem.

This is obviously only a practical solution for smaller cameras like point-and-shoots; DSLRs can't be attached this


495
490
50.5







way. For whatever camera you want to use, you may have to find/build your own unique holder.

## 12.4 Controlling the camera shutter

For creating an RTI dataset, you first need to acquire a set of photos taken at different lighting angles – that's what the RTI-Mage system is for. It lets you take up to 64 photos at different lighting angles, which can be processed into an RTI data file. There are a number of ways that the RTI-Mage allows synchronization of the camera shutter with each individual LED light at a different angle:

**Manual** For cameras that don't have any remote capability supported by the RTI-Mage. The LED lights and shutter are advanced manually.

**Automatic** A number of modes support automatic shutter triggering in synchronization with the LED lights, including

**CHDK** RTI-Mage supports cameras that can run the Canon Hacker's Development Kit through a USB cable connection, using the Remote Parameters settings.

**IR Remote** RTI-Mage supports Canon, Nikon, Sony, Pentax, Olympus and Minolta cameras that have an IR remote capability, using a custom-built cable.

**Wired remote** RTI-Mage can support some cameras with wired remote connectors, using a custom built cable.

**Bluetooth** Using the Adafruit Bluetooth HID module, RTI-Mage can support cameras that are controlled by computer software. Examples would include USB microscopes and Canon/Nikon cameras that use manufacturer's custom software.

**Servo** For cameras that don't support any other remote mode, a small servo can be used to mechanically depress the shutter automatically. **NOT YET SUPPORTED – I'M WORKING ON IT**.

Here's some more information about each mode.

### 12.4.1 Manual

This mode is for those whose camera does not support any of the automatic modes listed below; the Mode switch will need to be in the manual position. In this mode, an LED is lit manually with a press of the Action button, and stays on long enough to allow you to press the camera shutter button manually to take the picture. If the light goes off before you press the shutter, you can turn it back on with the WB button. When the photo is shot, press the Action button to advance to the next LED in the sequence. For good results, the camera and dome will need to be stably/firmly mounted in place, so that neither moves when the shutter is pressed. Otherwise, the RTI imagery may be a bit blurred, though it may be difficult to see that.

### 12.4.2 Automatic

There are a number of automatic mode options for the RTI-Mage system; which one you use depends on which kind of camera you have. Some of them require either special software installed on the camera or a computer, or a special cable to be used; these section will describe how to configure each camera/cable as required.

### 12.4.3 CHDK

CHDK (the Canon Hackers Development Kit) is custom firmware for many Canon cameras that can be temporarily loaded from the SD card to the camera, and allows remote triggering of the shutter through the standard mini-USB connector. You can find more information about CHDK at its wiki page, including a list of compatible cameras and installation info. Note that the wiki has many sections that are out of date. Also note that many recent Canon cameras do not work with CHDK – check the list of compatible cameras at the wiki to see if yours is. eBay or the Canon refurb store are good sources for older, CHDK-compatible Canon cameras.

The simplest way to install CHDK on your Canon camera's SD card is to use the STICK utility. Note that STICK can't reformat cards in exFAT format to the required FAT32 format, and Windows won't let you format cards larger than 32 GB in FAT32. However, the STICK page does link to several utilities that can re-format 64 GB and larger cards to FAT32. Once you've installed CHDK on the SD card on your computer, set it in write-protect format, insert it into your camera, then turn the power on. You should see a boot message for CHDK on your camera's screen. Use the buttons for your camera model to access the Main Menu for CHDK- these may differ from model to model.

On my Canon S110 and SX260, pressing the Play button briefly, then the Menu button, takes you to the Main Menu for CHDK. Once there, go to CHDK Settings, then select Remote Parameters. Turn on Enable Remote, set Switch Type to OnePush, and Control Mode to Quick. Exit out of the CHDK menu, and your camera is now set to fire the shutter when it receives a voltage pulse over a USB cable plugged into the USB Shutter jack on the RTI-Mage control box on one end, and the mini-USB jack plugged into the camera at the other end. The RTI-Mage camera mode will need to be set to Auto, and the Shutter mode set to USB. Once that's done, starting the photography cycle in Auto mode will turn on one LED, fire the camera shutter, then turn off the LED and advance to the next LED for the same cycle.

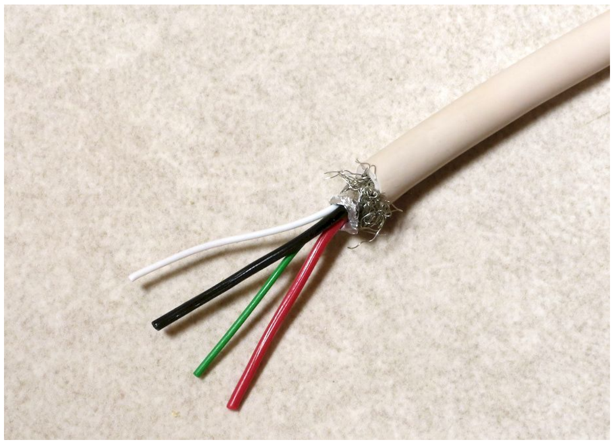
This is as good a place as any to mention a possible quirk with Canon cameras. When I upgraded my Canon camera from an older 8 megapixel model to a 12 megapixel one, I was surprised to find that some of my RTI imagery with the new camera looked blurrier than with the old camera. What's more, it wasn't consistent – sometimes it would be pin-sharp, other times not so much. Drove me crazy, but I finally figured it out. When you first turn on the Canon camera and set a zoom level, successive pictures show a slight "drift" in the pixel location from picture to picture, a maximum distance of about 7 pixels. You don't see that when you take regular photographs because it's a trivial amount, and you have no reference for it. RTI combines results from multiple photographs, though, so such a small drift can have a visible impact on the results. This appears to be hardware issue, as I've seen it with two different Canon cameras, but not a third. There's no solid fix, but there is a workaround. The pixel drift seems to settle out after about 30 pictures, so I shoot a set of dummy pictures first; subsequent picture sets show no signs of drift. Keep in mind that if you change the zoom level, or turn the camera off then on again, you'll have to shoot another set of dummy pictures to fix this issue.

### 12.4.4 IR-remote

For the IR remote, the RTI-Mage controller supports Sebastian Setz's Arduino library for camera IR remote control.

This supports a number of camera makes with IR remote capability, including Canon, Nikon, Sony, Pentax, Olympus and Minolta. To use this mode, you will need to build an IR remote cable to plug into the USB Shutter jack on the RTI-Mage control box (instructions below); you will also need to enable support for your camera model in the RTI-Mage software (more on this in another instruction step). The IR remote cable is simplicity itself. The positive and ground wires from a USB cable are connected to a 940 nm IR LED, with a 100R resistor in series to limit the current to 31 mA at 5V. To build the IR remote cable, grab a spare USB cable, and cut off the end that isn't the USB A male plug (see here):

Strip the insulation off the cut wire end, revealing four wires: red, black, green and white:



Cut off the green and white wires – you won't need them.

Saw a small section of protoboard off the larger piece listed in the components section. Sawing seems to be the best option, as cutting tends to make the board explode into little bits. Read somewhere someone suggesting drilling out the holes in the board to make a cut, but haven't tried that. Take the IR LED and stick it into one side of the board, long lead (positive/anode) on the left. Bend the IR LED until the leads are flat against the protoboard:

Insert a 100R resistor into the protoboard next to the left LED lead (anode), and bend the leads as shown:

Strip insulation off the red and black wires of the USB cable. Insert the red wire into the hole adjacent to the resistor; insert the black wire into the hole adjacent to the right LED lead (ground/cathode). Solder all adjacent connections and trim off excess leads; attach the main USB cable to the protoboard with a bit of wire:

Didn't take a picture, but I globbed hot glue over all the components on the protoboard to hold them securely.

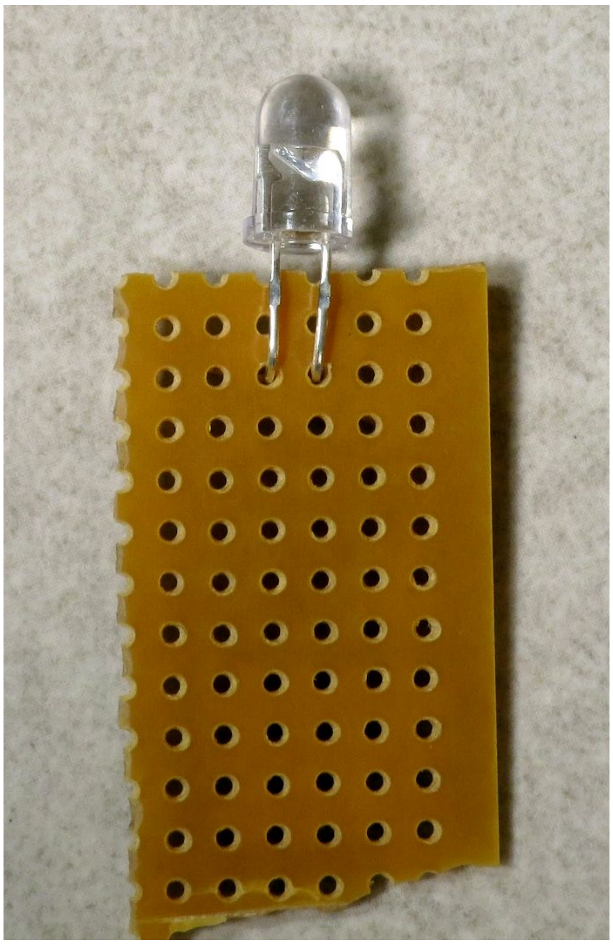
It's hard to test whether this LED is working correctly, even when plugged in, because the IR light is invisible to the naked eye. Camera sensors can detect IR light, which is why they normally have an IR filter in front to block out IR. However, this IR LED is bright enough that a camera can pick it up.

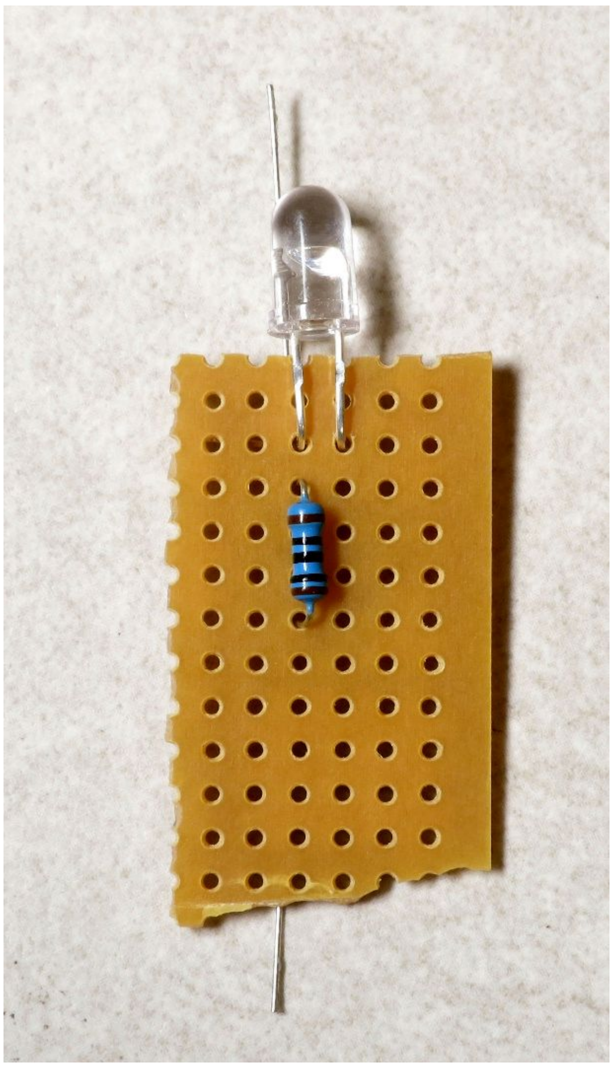
Grab your digital camera, set your ISO to the highest level you have, set the mode to Auto, then turn out the lights. Plug the IR LED cable you just made into a 5V USB power supply (your computer will work fine), and point the LED straight into the lens of the camera. In the display, you should see a faint purplish glow – that will tell you that you've assembled the IR LED cable successfully.

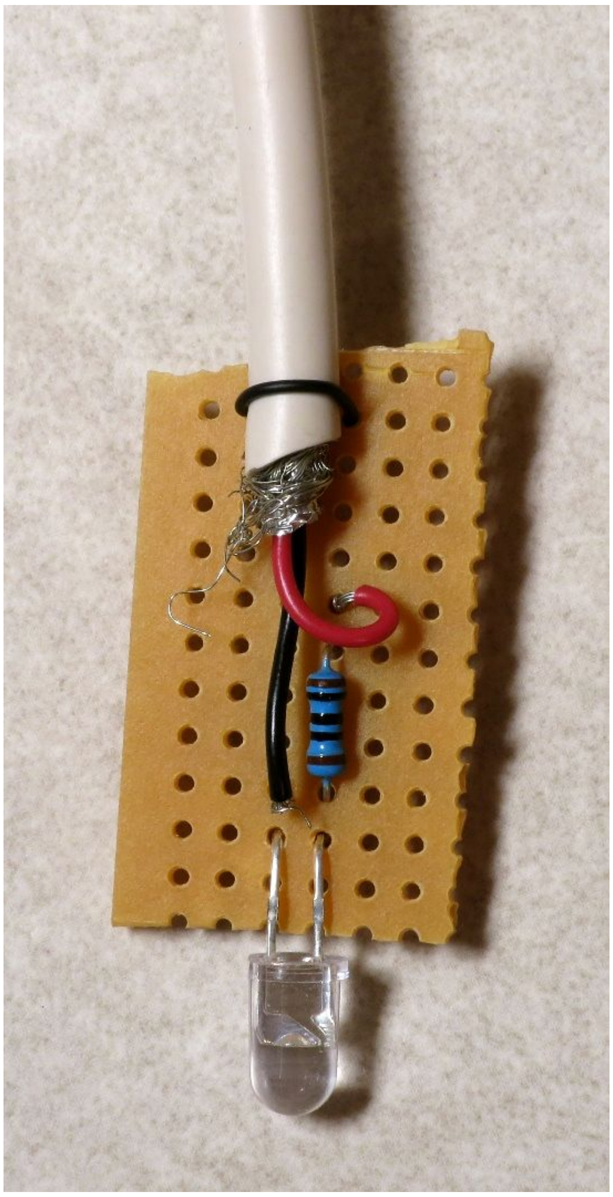
Once I saw the IR LED was working, I wrapped it up in black Gorilla tape for the final packaging. Feel free to pursue a more elegant packaging solution.

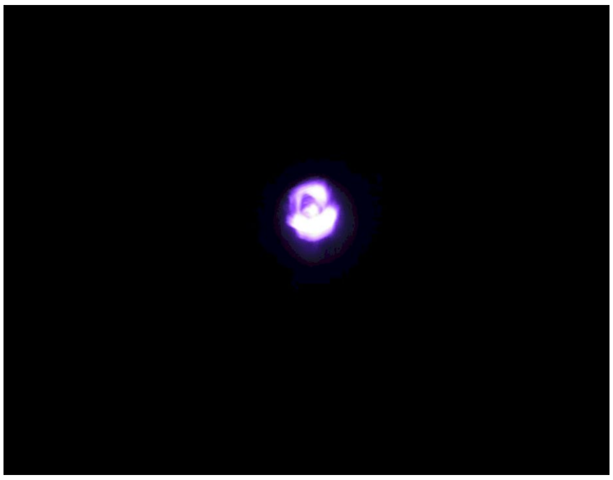
To use this IR remote with a compatible camera and the RTI-Mage controller:

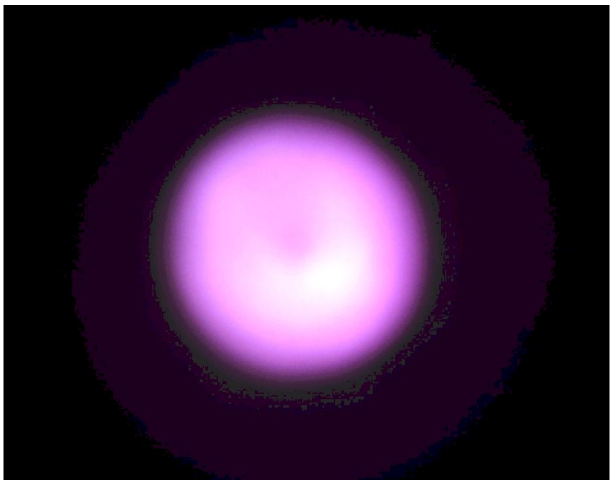
1. Modify the RTI-Mage software to enable your camera make (more on this in the upcoming section on the RTI-Mage control software).
2. Plug the IR remote cable into the USB Shutter jack.
3. Set the Mode switch to Auto, and the Shutter mode to IR/Bluetooth.
4. Point the IR at the camera's IR sensor (check the manual for its location), and then fix it in place (tape, clamp, wire, whatever).
5. Check your owner's manual for how to set the camera to respond to an IR signal. Typically, this involves one of the timer-related control settings on your camera.











That should be it. Once that's done, starting the photography cycle in Auto mode will turn on one LED, fire the camera shutter, then turn off the LED and advance to the next LED for the same cycle.

### 12.4.5 Wired remote connection

A wired remote connection can be created for those cameras that support it. However, unless the camera doesn't support IR remote capability, I would recommend you use the IR option. A single IR cable can support multiple camera modes, but unfortunately just about every camera maker has a proprietary plug/jack for their wired remote. There's a full set of pictures and pin-outs at the Camera Remote Release Pinout list, which will give you a feeling for how complicated it can be.

However, for many of the cameras, the mechanism is quite simple – if you complete a circuit across one/two camera remote connections and ground, the shutter will fire. There are many ways this could be done mechanically, with relays/electromechanical switches, but the reading I did suggested that there are problems with switch "bounce" with these. Bounce means that the mechanical switch hits the other contact, bounces off, then hits again for a double-contact; this means the potential for a double shutter release. So I'm using a commonly-employed circuit that closes a circuit electronically using a chip called an optocoupler (or optoisolator).

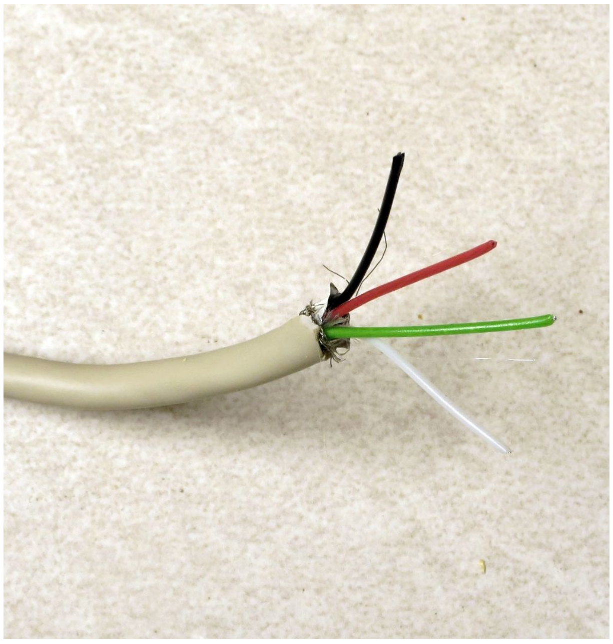
This chip has an LED and a phototransistor. When the LED is off, the phototransistor doesn't allow any current to flow, leaving the circuit open; when the LED is on, the phototransistor lets current flow, closing the circuit. The optocoupler chip I use is the 4N35, cheap and readily available, but there are many other models you could use.

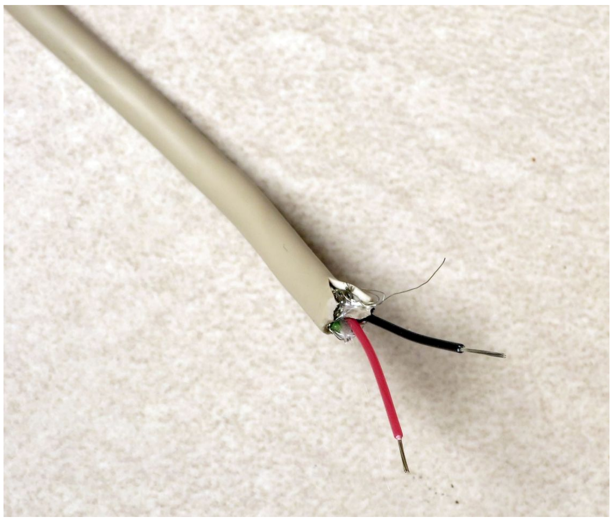
As with the IR remote, grab a USB cable and cut off the non-USB A end; strip the cable insulation off:

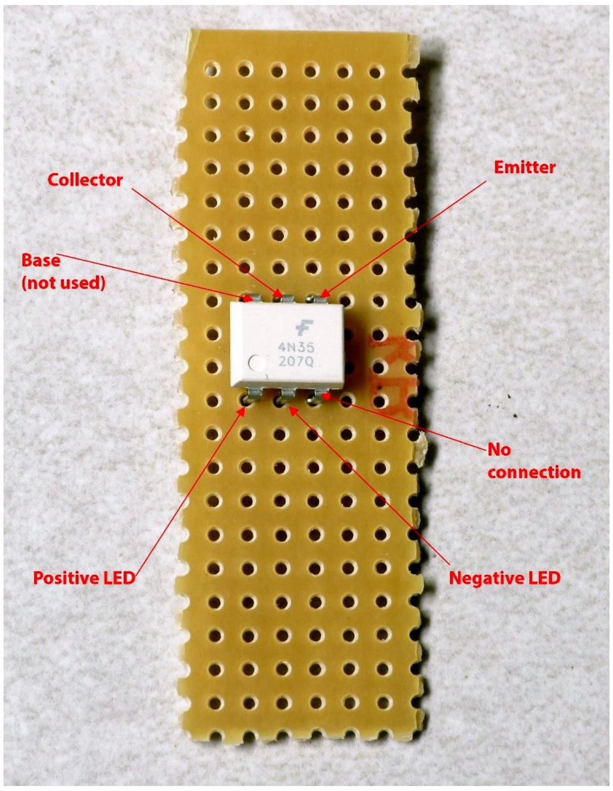
Cut off the unneeded white and green wires, and strip insulation off the other two:

Cut out a small piece of protoboard, and mount the optoisolator on it:

You'll be connecting the LED pins to the USB wires, with a current-limiting resistor in series. You'll be connecting the collector and emitter to the positive voltage and ground connections of the wired remote cable.







Insert a 100R resistor next to the positive LED pin:

Insert the red USB wire next to the resistor lead, and the black USB wire next to the ground LED pin on the 4N35:

Solder the connections together; I attach the USB cable to the protoboard with a bit of twisted wire for strain relief:For some remote cables you can find generic plugs, like the 2.5mm stereo jacks used for older Canon camera wired remotes. For most recent cameras, though, the connectors are non-standard/proprietary.

Cheapest source for one of these is to order an inexpensive wired remote for your model from eBay or Amazon, then cut off the remote switch. Strip the insulation off, and for Canon/Nikon/a few others, you will likely see three wires (this is an older Nikon cable):

For some remote cables you can find generic plugs, like the 2.5mm stereo jacks used for older Canon camera wired remotes. For most recent cameras, though, the connectors are non-standard/proprietary. Cheapest source for one of these is to order an inexpensive wired remote for your model from eBay or Amazon, then cut off the remote switch. Strip the insulation off, and for Canon/Nikon/a few others, you will likely see three wires (this is an older Nikon cable):

One of these is the ground connector, one controls focus, and the third fires the shutter. The colors aren't standard, unfortunately, so you'll have to use a multimeter and the pinout diagrams from the DIY.net site to figure out which is which. In this case, blue was ground, green was focus, and red was shutter.

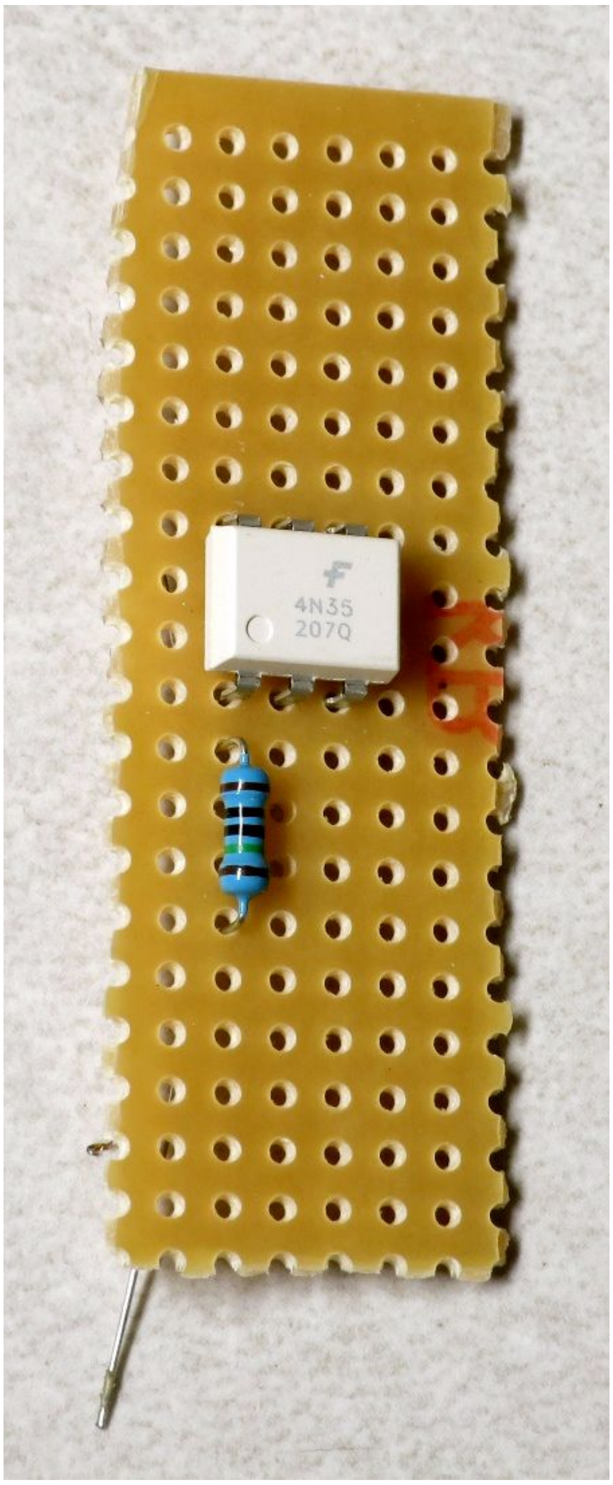
From experience, I know that with Canon cameras you only have to connect the ground and shutter wires to the remote to have it work correctly with the RTI-Mage. However, with Nikon cameras, both the focus and shutter wires need to be connected together in order for the shutter to fire correctly:

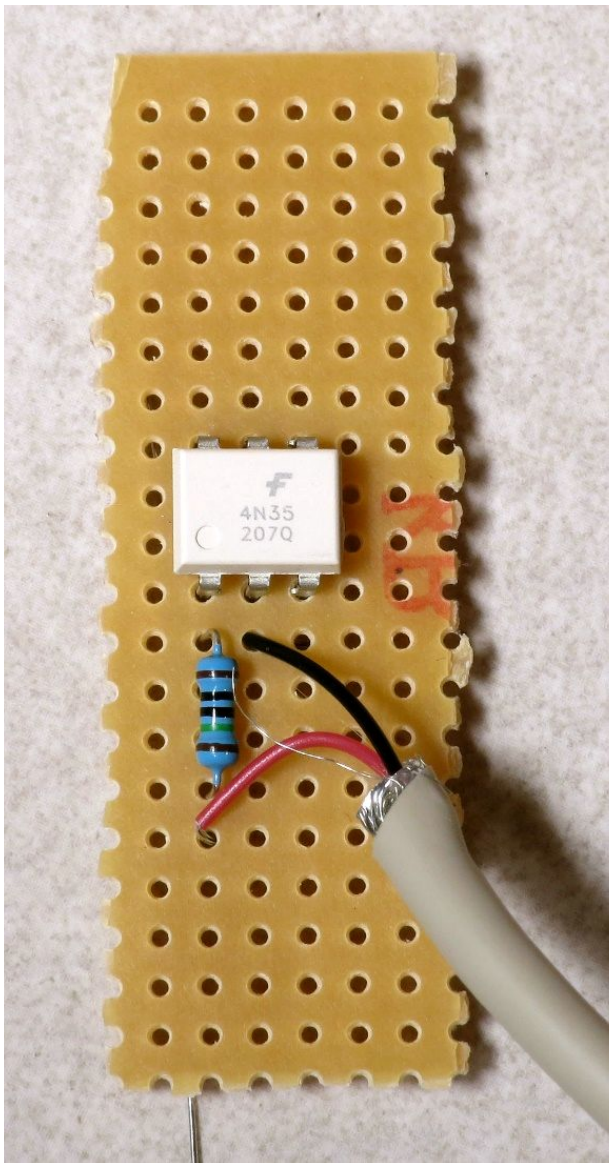
Insert the ground wire next to the emitter pin on the 4N35, and the focus/shutter wires next to the collector pin. Solder in place; add strain relief to the cable with twisted wires:

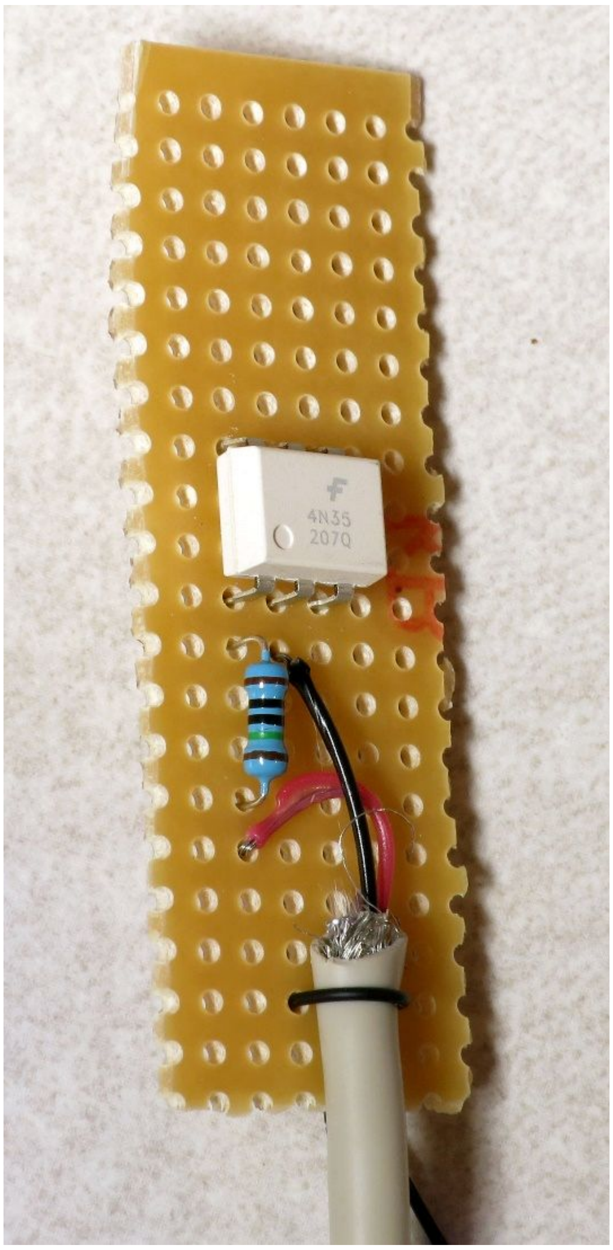
Just to secure everything in place, I globbed hot glue on everything:

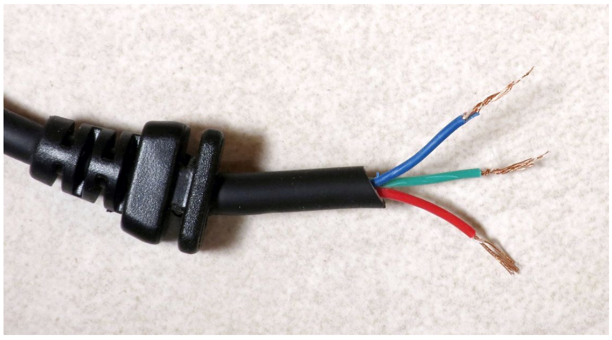
And then wrapped it all up in Gorilla tape:

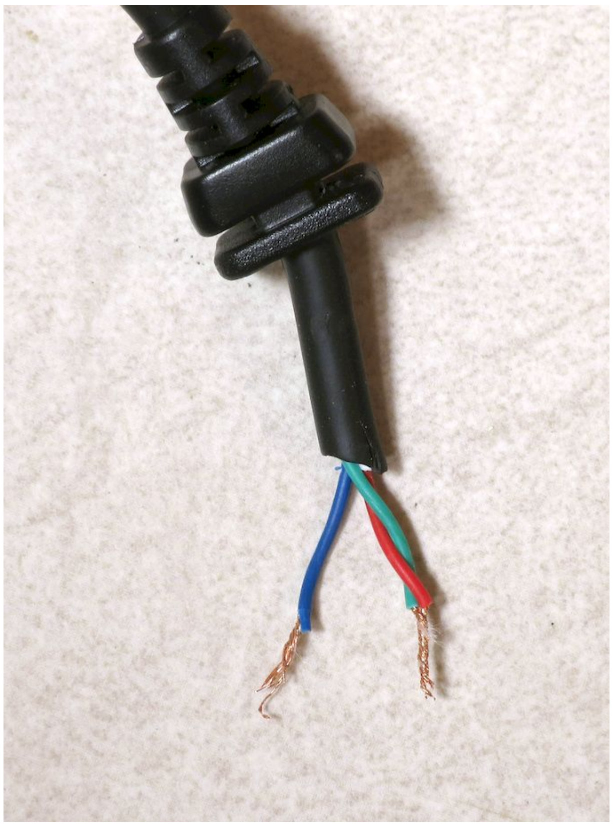
To test the cable, connect your multimeter in resistance mode to the shutter and ground pins on the connector, then plug it into any powered USB jack. You should see the resistance drop from infinity down to about 50R or so, which should be enough to trigger the camera shutter. Canons usually work reliably, Nikons can be a bit flakey; that's why I prefer the IR remote, since it works more reliably with all cameras.

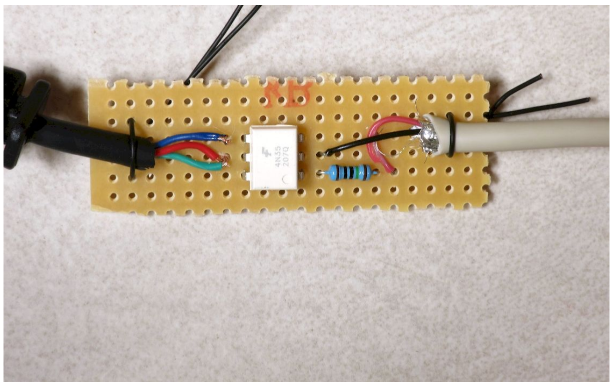

4N35
207Q

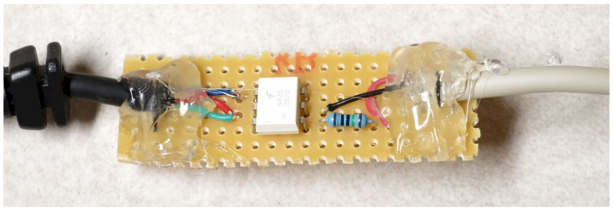

4N35
207Q
KB


4N35
207Q









To use this remote cable with the RTI-Mage:

1. Plug the wired remote cable into the USB Shutter jack.
2. Set the Mode switch to Auto, and the Shutter mode to USB.
3. Check your owner's manual for how to set the camera to respond to an IR signal. Typically, this involves one of the timer-related control settings on your camera.

That should be it. Once that's done, starting the photography cycle in Auto mode will turn on one LED, fire the camera shutter, then turn off the LED and advance to the next LED for the same cycle.

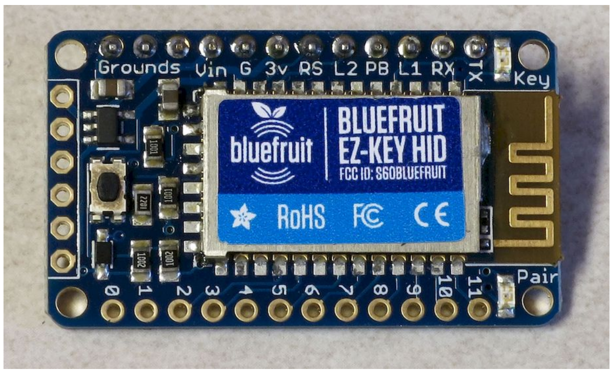
### 12.4.6 Bluetooth

For Bluetooth, the camera shutter controlled via a computer program is fired using the Adafruit Bluetooth HID module. This is useful for USB microscopes, and Canon/Nikon cameras operated remotely via proprietary software.

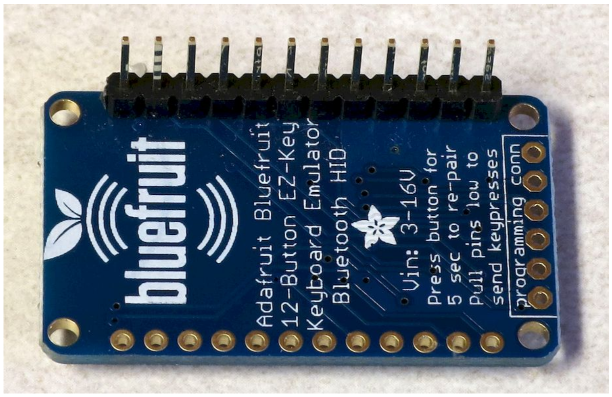
There are certain circumstances where you might want to trigger the camera shutter using a computer program, rather than the main camera shutter control. Canon and Nikon have programs that allow you to display the camera view on a computer monitor, and set focus/exposure parameters from that program as well as firing the camera. In most cases, USB microscopes need to be controlled via a computer program, since most of them don't have onboard storage.

I've been using a USB microscope with my RTI-Mage system, and for that I have added the Adafruit Bluetooth EZ-Key HID adapter to the control box. This little board lets you send either a series of keyboard commands, or a mouse click, to the computer you've paired it with. All programs controlling a camera/microscope will have either a set of keyboard commands, or a button enabled with a mouse click, that can be triggered with this adapter.

Here's my Adafruit board:

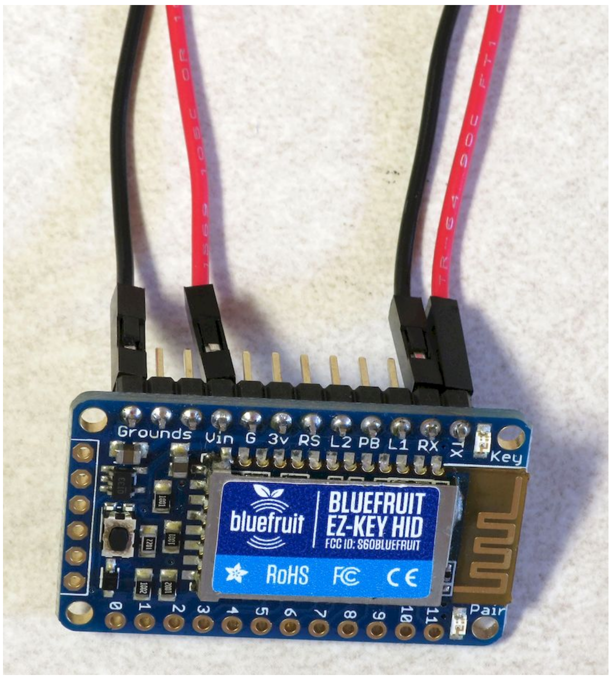


It comes without the pin header soldered on, but does come with the pin header so that you can do the soldering:



You can find the documentation for all the pinouts at the Adafruit site. For the RTI-Mage system, only four of the pins are used: Vin (to supply 5V power), Grounds (the other half of the power), and Tx/Rx (transmit to and receive information from the Arduino). Even the Tx is superfluous, since the Arduino is currently only sending info to the EZ-Key and not receiving it, but you might have a use for it at some point (e.g. having the computer trigger some action in the control box).

You need to connect wires to the four pins specified above, and have a bare wire lead or pin at the other end. I cut four pieces of wire, soldered female Dupont pins to one end, put plastic covers on the Dupont pins, then slid them onto the male pins:



I'd recommend using red/black wires for Vin/Grounds respectively, following conventions. I'd also recommend using different colors for Tx and Rx, to help differentiate them.

Install the EZ-Key in the RTI-Mage control box:

- Vin connects to the +5V female strip on the power board.
- Grounds connects to the ground strip on the same board.
- Tx (red above) connects to pin A14 on the Arduino Mega (digital pin 68)
- Rx (black above) connects to pin A15 on the Arduino Mega (digital pin 69)

That's it for connections. When you hook up power, a red LED should flash on the EZ-Key. Pairing the EZ-Key with your computer requires pressing the button on the EZ-Key until the red LED starts flashing quickly, then using the Bluetooth control panel on your computer to pair with the EZ-Key. To use Bluetooth commands, set the Shutter mode switch in back to IR/Bluetooth mode.

In the RTI-Mage control software, there's a subroutine called BT_Shutter that sends a keyboard or mouse command to your computer, depending on how you want to control the computer program. Here's the relevant code:

```
void BT_Shutter() { //Sets commands for Adafruit Bluetooth HID module to fire␣
↪shutter using computer
       //Fires shutter with Alt-f, s command
       //BT.write(0xE2); //Alt key
       //BT.write(0x66); //f key
       //delay(25);
       //BT.write(0x73); //s key

       //Fires shutter with left mouse button
       delay(LED_Pause_Time);
       mouseCommand(0x1,0,0); //Left button down
       delay(25);
```

(continues on next page)

(continued from previous page)

```
        mouseCommand(0,0,0); //Release button
}
```

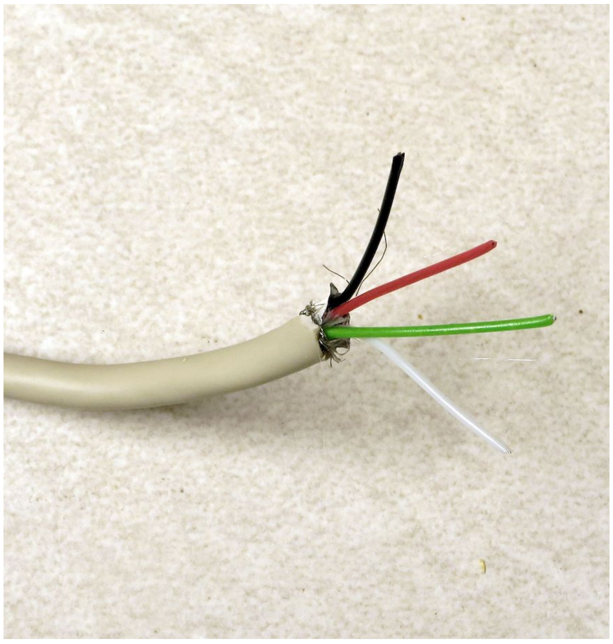
I have two USB microscopes. One of them lets you capture an image with keyboard commands, Alt-F, S. The first part of the code implements sending those keyboard commands to the program controlling the USB camera, and saving images. The program window must be active in order for the command to work. For example, if you start recording images, then open up a different program, no more images will be recorded/saved, since the window is no longer active. This code is commented out here with two slashes in front, since I don't normally use that USB microscope. If your program uses different keyboard commands to fire the shutter, you'll need to modify this – check the Adafruit documentation for the required hex codes.

My other USB microscope doesn't have a keyboard command to capture an image; instead, it relies on a mouse click on a capture button. The second part of the code implements that mouse click for the EZ-Key HID board. The capture program needs to be active, and the cursor needs to be sitting on top of the capture button, in order for this to work. If you move the cursor off the capture button, or the capture program is no longer active, then image capture will stop while the RTI lights keep moving on.

The delay(LED_Pause_Time) command introduces a short delay between turning on the LED and enabling the EZ-Key. I found that without this, the program might only capture part of the image. This constant is set in the top of the program, and you can tune it to whatever value works for you. You'll also have to experiment with LED and Delay times to find the optimal setting for your application.

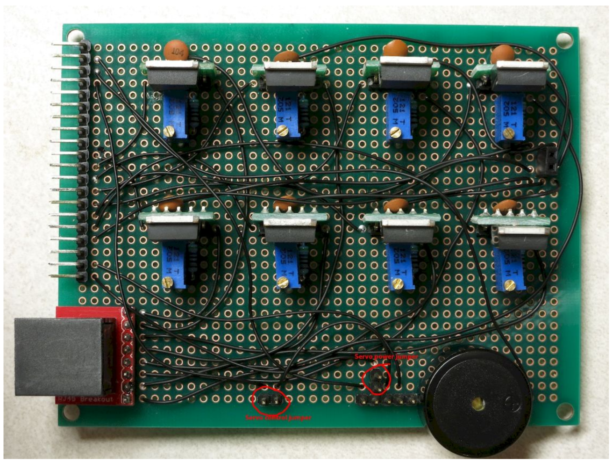
### 12.4.7 Servo

A control system to fire the shutter mechanically using a rotating arm. This isn't ready yet, hope to work on it shortly. If you'd like to try making this work on your own, start by cutting a standard USB cable to leave the USB A connector intact. Strip off the cable insulation to reveal the four wires:



Cut off the red wire, that won't be used here. Black is ground, white is power, green is control. You'll need to connect two jumpers on the CAT4101 board to enable both the white and green wires:

Arduino pin 27 is the servo control pin. The rest, programming and mechanical connections, is up to you for now. Hope to get to it myself shortly.


Servo power jumper
Servo control jumper

# CHAPTER 13

# Controller software, calibration, and acquisition

## 13.1 Control system software

If you've gotten to this point, then all the hardware you need to do RTI is complete, with the exception of some minor optional extras. So it's time to load the software into the Arduino that makes the control box fully operational for acquiring RTI datasets. The software's name is RTI-Mage_Main_Control, and the latest version will always be in the Files section of the Hackaday project page. But before you upload it to the Arduino, there are some constants you may want to modify, either right away or sometime in the future. Load the program into the Arduino IDE, and you'll see right at the top a whole bunch of constants are defined. Many of them are hardware related, so if you modify those, the system may not work. But there's a number of them that can/should be modified depending on your needs. Here are the relevant program strings, what they mean, and how you can change them.

```
String Current_setting="700mA";
```

This is a text string that shows up on the boot-up splash screen, telling people what the current output value you set earlier is. The system has no way to detect the current output setting, so you should definitely specify it here. Make sure it stays in quotes.

```
const unsigned int Min_Data_Transfer_Time=500;   // minimum time after LED goes␣
↪off to save photo data
const unsigned int Max_Data_Transfer_Time=6000; // maximum time after LED goes off␣
↪to save photo data
```

These are the minimum and maximum values for the "Delay" time, controlled by the right "Delay" knob on the front panel. This delay is to allow the camera time to save the photo it's just taken to memory. For high-quality DSLRs, this time can be a fraction of second; for lower-end cameras, it can take a few seconds. The values are in milliseconds, thousandths of a second, so the number above correspond to 0.5 seconds and 6 seconds. Feel free to modify either of these if you feel the values are too high or too small for your needs. Make sure you only enter an integer value, no decimal point.

```
const unsigned int Min_LED_Time=100; //minimum time LED is on for picture to be␣
↪taken
const unsigned int Max_LED_Time=4000; // maximum time LED is on for picture to be␣
↪taken; set by USB microscope time
```

These specify the minimum and maximum times the LED light is on during picture taking, controlled by the left "LED" knob on the control panel. This will depend on the exposure time required. Don't increase the maximum time unless you absolutely have to, since the longer the LED is on, the warmer it gets. For example, I sometimes have these set to a wider range (100 to 8000) since one of the USB microscopes I'm using requires a long exposure time, roughly 6 seconds. However, for that system I also have the current set to a very low value of 150 mA, so it won't get hot despite the longer on time.

```
const unsigned long White_Balance_Time=3000; //time for one white balance LED␣
↪before switching to the next one
const unsigned long White_Balance_Max_Time=24000; //maximum total time for white␣
↪balance before it turns off (to prevent LED stress)
```

Pressing the black "WB" button starts a long light cycle, where the LEDs in a single row are lit up in succession, each for the time specified by the White Balance time. This is to allow you to set the camera white balance, focus, and exposure parameters. The time is limited to prevent the LEDs from getting too warm, and the Max_Time is set to turn off the function after a limited time, to keep it from going forever. I wouldn't modify these unless you absolutely have to.

```
const unsigned long Min_LED_On_Time=500; //Minimum on time for LED in manual mode
const unsigned long Max_LED_On_Time=4000; //Maximum on time in manual mode for LED␣
↪(to prevent overheating); turn back on with white balance button. Higher values␣
↪not recommended.
```

These are the minimum/maximum LED on times in manual mode; you set the actual value using the LED potentiometer knob. Probably shouldn't modify these unless absolutely necessary. If the LED goes off in manual mode before you have a chance to press the camera shutter, pressing the black WB button will turn that LED back on again.

```
const unsigned int Frequency=1000; //Sets default frequency of beeper (in Hz,␣
↪cycles per second)
```

Sets the base frequency for the beeper in Hz; 1000 Hz was chosen as the default because it's a balance between high human ear sensitivity and annoyance. Lower frequencies are more pleasant, but the ear's response drops off significantly below 1000 Hz. Frequencies up to 5000 Hz have a better ear response, but sound more annoying. Change this if you like, won't affect actual operation. And remember that there is a switch on the control box that turns off all sounds.

```
const unsigned int Rows=8; //Number of Rows being used (cathode connection to␣
↪CAT4101s)
const unsigned int Columns=8; //Number of Columns being used (anode connection to␣
↪MOSFET driver outputs)
```

These two are the most important variables to modify. The defaults of 8 are the maximum number of rows and columns the RTI-Mage control box can support. If you are using a dome with fewer rows or columns, you should modify these values accordingly. These values are also displayed on the boot-up splash screen.

```
const int Shutter_Voltage_Time=100; //Time for 5V at USB to either fire CHDK or␣
↪remote wired shutter.
```

This is the length of the positive voltage pulse used by the CHDK or wired remote shutter systems. This value has worked fine for every Canon camera I've used with CHDK. If you're having problems getting your wired remote cable to work, you might try increasing this value.

## 13.2 Camera constants

Below the constant variables is a section of commands related to using the IR remote shutter function with various makes of cameras. Only one camera make can be enabled at any time. The default is Canon. To change to a different make, take the two Canon lines:

```
Canon CanonCamera(USB_Camera_Pin);
String Camera_type="Canon";
```

And put two forward slashes in the beginning of each line; this makes the line a comment that the program interpreter will ignore. Remove the slashes from the similar lines for the desired camera make, and that will enable those lines. You're not quite done yet – you need to change one more line as well. Scroll down in the program to the IR Shutter subroutine, where you will see another set of commands with camera makes associated with them, e.g this one for Canon:

```
CanonCamera.shutterNow();
```

As above, put two forward slashes at the beginning of the line you want to deactivate (Canon in this example), then remove the two forward slashes from the make you want to activate.

Once you've modified whatever constants you need to, upload the program to the control box. Once uploaded, you don't need to have the system connected any longer to your computer via the USB cable; in fact, it won't serve any purpose at all.

A quick visual review of the controls/jacks on various panels of the control box:

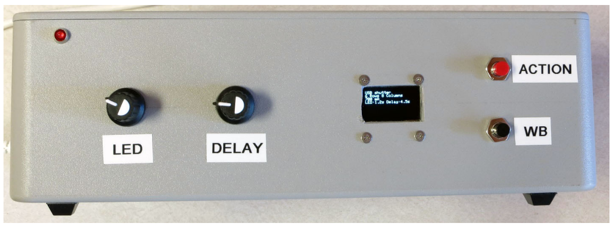


Fig. 1: Front

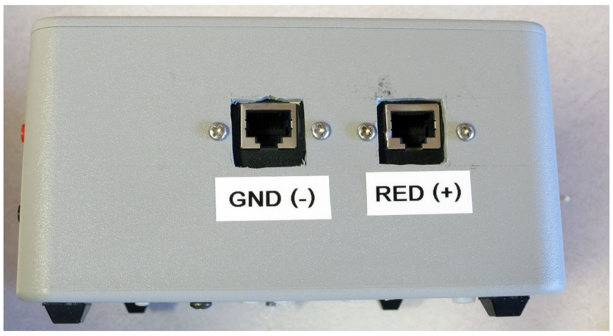


Fig. 2: Right

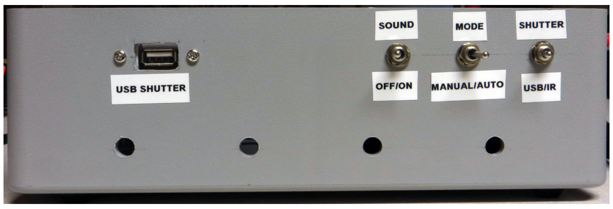


Fig. 3: Rear

Disconnect the USB cable in the jack on the left panel connecting the Arduino to your computer, as it's not necessary. If you haven't already, connect the positive and ground cables from the LED dome to the control box ("red to the rear", "close to the ground"). Now plug in your 9-12V 2A+ DC power supply into the jack on the left. If the Sound switch is on, you should hear 3 rising tones.

If you haven't installed the OLED screen, ignore the following OLED-related screens and descriptions, but be aware that the descriptions of modes and functionality is the same. If you've installed the OLED display, you will see the following splash screen for 5 seconds:

This will show you the software version and date, the number of rows/columns/LEDs (handy in case you use the

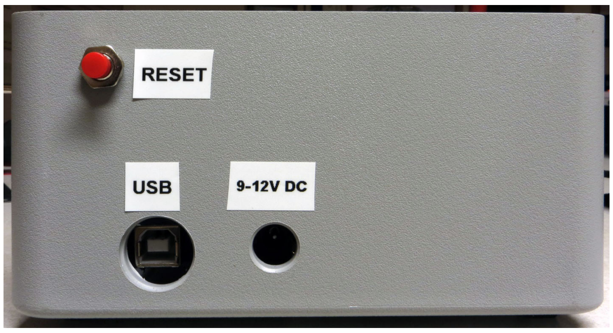
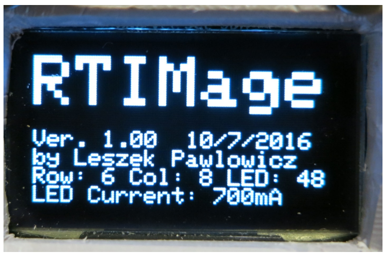
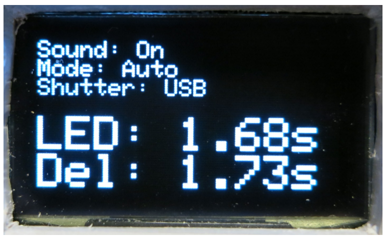


Fig. 4: Left



```
RTIMage
Ver. 1.00  10/7/2016
by Leszek Pawlowicz
Row: 6 Col: 8 LED: 48
LED Current: 700mA
```

box to control multiple domes with different numbers of LEDs), and the current value you set before uploading the program. After 5 seconds, the OLED screen goes to the main settings screen:



```
Sound: On
Mode: Auto
Shutter: USB
LED: 1.68s
Del: 1.73s
```

This is where you see the setting of the three mode switches (first three lines), then the status of the LED on time and the Delay time (in Auto mode). In Manual mode, there is no Delay time, so the screen will look a bit different:

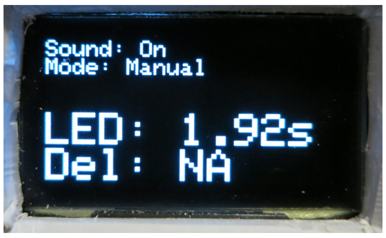
Switch back to Auto mode. The first picture was with the Shutter mode in "USB setting"; switch to the "IR/Bluetooth" shutter setting, and you'll see this:

Not only does the shutter mode change to IR/Bluetooth, but there's a new line that indicates what camera make the IR remote can control. The default is Canon, but as mentioned above, you can change this in the control software to any supported camera model (CanonWLDC100, Nikon, Sony, Pentax, Olympus and Minolta).

Now push the black WB button (lower right on the front panel). The LEDs on the top row will each light up in sequence for the time set by the constant mentioned above. This will give you the time needed to set the exposure values, white balance, and focus. If you press the red Action button, the next row of LEDs down will start lighting up, and additional presses will push you down one row further. This is mainly intended to check that if you are photographing multiple objects simultaneously, neither object will cast a shadow on the other one for low LED lighting angles. Pressing the black button briefly will turn off this mode, or you can wait for the cycle to end automatically.

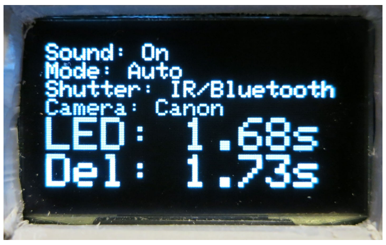
With the control box in Automatic mode, press the red Action button – the LEDs in the dome will now start turning on in sequence. If the sound is on, you'll hear a beep every time an LED lights up. The time the LED is on, and the time between one LED going off and the next going on, is set with the two knobs on the left front panel. Try playing with these during this Action sequence – the times will update automatically during the sequence. If you check the OLED screen, you'll see it's changed to this:

It will show you on top the number of the LED currently on, and how many total LEDs there are, so you can see how far along in the process you are. On the bottom are the LED on and Delay times, which you set with the front



Sound: On
Mode: Manual

LED: 1.92s
Del: NA



Sound: On
Mode: Auto
Shutter: IR/Bluetooth
Camera: Canon
LED: 1.68s
Del: 1.73s

knobs. Once the series is completed, and all LEDs have been turned on and off, the OLED display returns to the main settings screen. If the sound is on, you'll hear three short beeps to indicate the photo cycle is done.

Now change the mode to Manual. Press the Action button, and the first LED will light up for the time set by the LED knob; you'll see that time in the Manual status screen in the lower half.

The Delay knob has no function in this mode. During the time the LED is on, you should be engaging the shutter button on your camera manually. If the LED goes off before you have a chance to press the shutter, the display will change to show you that:

Pressing the black WB button will light the same numbered LED again for the same period of time. Once you've taken the photo, you can advance to the next LED by pressing the red Action button. The OLED display will keep track of which number LED you're currently on. When you've lit the last LED and taken the last picture, press the red button one last time. The OLED display will go back to the settings screen, and you'll hear three short beeps.

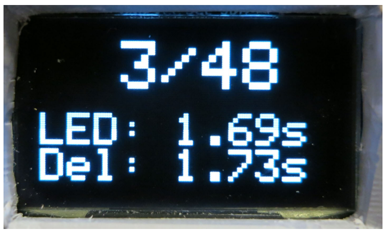
If you need to stop an Auto run, or if anything seems to go wrong, just press the Reset button on the top rear of the left panel.

Play with these functions and controls for a few minutes to make sure you understand how the system functions; there really aren't that many, so shouldn't take you too long to figure it out. Now you're ready to start doing some actual photography, beginning with some required system calibration information.

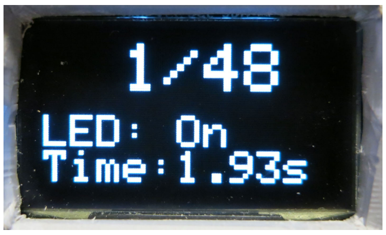
## 13.3 System calibration

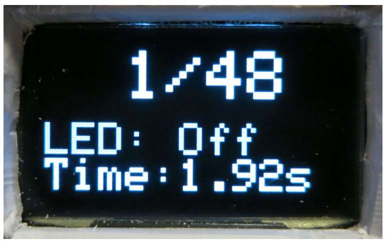
In creating an RTI dataset from the photos you generate with the RTI-Mage, software fits a separate curve of pixel luminance as a function of lighting angle for every pixel in the image, using the pixel luminance data from the photos you've taken. It knows which lighting angle was used for each photo with a special text file called a "light position" file, with a .lp extension.

The first line of the file contains the total number of photos, while the remaining lines contain the path of each individual photo, along with information about the lighting angle. More specifically, it contains the x,y,z projections of the lighting angle vector from the center of the photograph out to the light source. Here's a few lines from one of my lp files:



3/48
LED: 1.69s
Del: 1.73s





```
48
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_01.jpg 0.
↪13742885 0.44443336 0.8852075
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_02.jpg -0.
↪21295299 0.39356261 0.8942927
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_03.jpg -0.
↪4445167 0.11216316 0.88872063
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_04.jpg -0.
↪39619443 -0.2289121 0.8891733
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_05.jpg -0.
↪11848045 -0.46047336 0.879731
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_06.jpg 0.
↪24940234 -0.41810092 0.873493
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_07.jpg 0.
↪45666453 -0.12569694 0.88071436
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_08.jpg 0.
↪4082768 0.20535836 0.8894594
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_09.jpg -0.
↪06528884 0.61326444 0.78717476
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_10.jpg -0.
↪47801396 0.36893168 0.79711485
H:\RTI_Data\Hackaday_field_point_PTM\jpeg-exports\Hackaday_field_point_11.jpg -0.
↪6062705 -0.072234996 0.7919711
```

And so on for the full set of 48 photos. You can open a .lp file in any text editor.

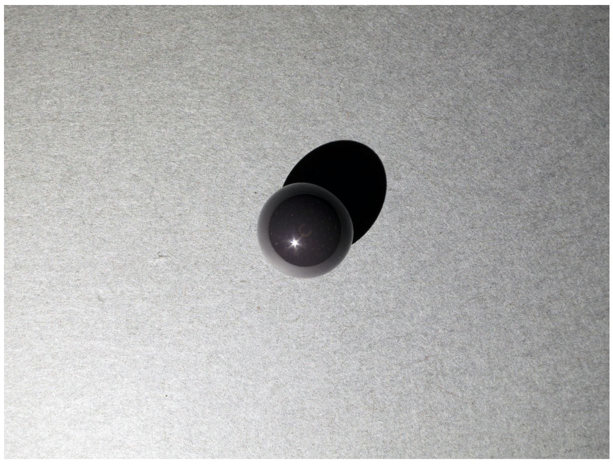
For simplicity, the length of the vector from the center to the light source is normalized to a dimension of one. Take all the numbers in one line, square them, add them up, and they should add up to one.

You could measure the angles of every single LED in the dome, and calculate these values, but that would be a pain. Fortunately, there's an easier way, using highlights in a shiny black or red sphere.

Here's a photo of such a shiny sphere taken inside the RTI dome with one of the LED lights on:

See the bright spot on the surface of the sphere? That's a specular highlight from the LED. If it's a perfect sphere, you can use the position of that highlight relative to the rest of the ball to calculate the lighting angle when the photo was taken. This would still be a pain to do if you had to measure and calculate the angle manually; fortunately, there's software that can do this for you automatically.

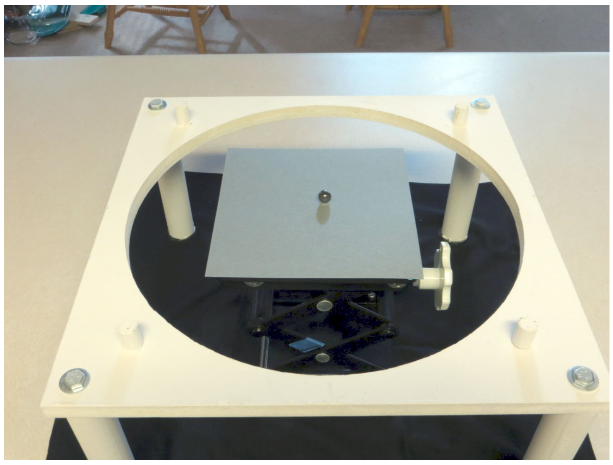
First, we need to get a full set of photographs of such a shiny ball using the RTI dome. The ball above is a 12mm silicon nitride ball, with a very smooth and dark surface. However, you could use a shiny black marble, a ball bearing dipped in ink, or any other dark shiny object as long as it's spherical in shape. When you photograph it in your dome, it will need to be at least 250 pixels across in the photo, or else the software may have trouble picking out the highlight. But you also don't want it to be too big, as you may not get accurate angular positions if you put a large sphere in a small dome.



This step will walk you through the process of getting the light position calibration data you need. This will be almost identical to the process of photographing an object/artifact in the dome; the only real difference will be in the post-processing.

Let's start by setting up the dome stand. I put the stand on top of a dark cloth to minimize back-scattered light, but truthfully it makes little difference – I've done runs with and without the cloth, and can't see any difference in the results.

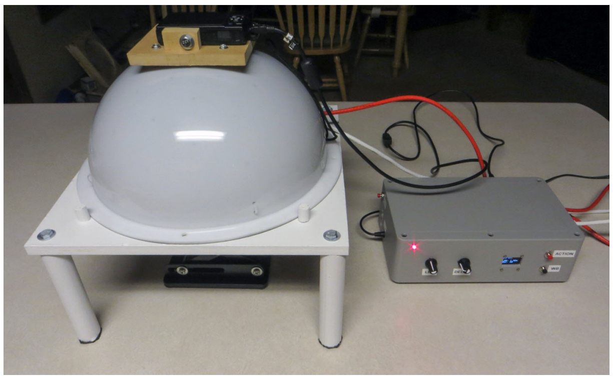
I've put my lab jack in the middle of it, which is what the shiny ball (or any object) will sit on, and raised it to be at the same height as the top of the dome rim (usually a few mm above the stand base that the dome will sit on). Don't worry about getting it perfectly centered – there's room to move it around inside of the stand. I put a gray background on top of the lab jack, and place the shiny black ball as close to the center as I can. It's gray so that you can distinguish the black sphere from the background. Once again, don't worry about precise positioning:



If the ball is a perfect sphere, it will definitely have a tendency to want to roll away. If you look closely at the base of the sphere, you'll see there's a small plastic washer that the ball is sitting on, which keeps it in place. Whatever washer you use, make sure it's smaller than the diameter of the ball. Time to put the rest of my RTI-Mage system parts in place:

1. Dome on top of the stand.
2. Ethernet cables from the dome plugged into the control box.
3. Camera in position on top of the dome (my Canon S110 in this case)
4. A micro-USB cable running from the USB Shutter jack on the rear of the control box to the USB input of the camera (this will fire the shutter automatically using CHDK). If you were using a different remote mode, like the IR remote cable, you would have that plugged into the same jack, with the LED pointed at the IR sensor of your camera. You can also do this procedure in Manual mode.
5. 9-12V DC power plugged into the control box.

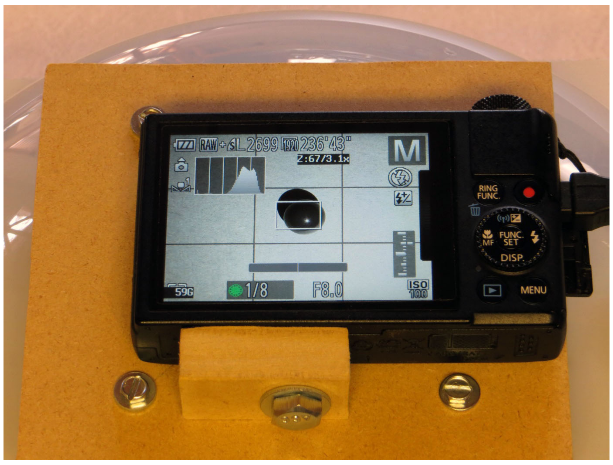
6. Correct mode and shutter setting. In this case, I'll be running in Auto mode, with the shutter set in USB mode for firing the Canon.



Once the system is setup and ready to go:

1. Set the camera to its Manual setting, where you set fixed exposure time and aperture values.
2. Press the black WB button, which turns on the top row of LED lights for a prolonged period. Use this lit time to:
   (a) Center the object in the camera view screen by moving the lab jack or dome around.
   (b) If possible, zoom in the camera as much as you can to get the largest possible image of the sphere.
   (c) Focus on the sphere by partially depressing the shutter button.
3. Determine the proper exposure settings. Set the aperture as small as possible to get good depth of field, all parts of the sphere in focus. Then set the exposure time so that the LED reflected highlight is clearly visible on the sphere, but the edge of the sphere is clearly defined against the background. The first picture above is a pretty decent example of that, but you could make the exposure a bit lighter or darker than that and still get acceptable results.
4. Set the JPG picture quality to the highest level. If you can shoot in RAW format, do that, and convert to JPG later on.
5. You will want to use your camera's manual focus mode. Auto focus can work sometimes, but it will take longer to acquire the full dataset since the camera will have to re-focus for every picture. And for shallow lighting angles, autofocus can sometimes have problems because of the reduced lighting intensity. What I normally do with my Canon cameras is an initial autofocus with a half-press of the button, then switch over to manual focus mode. Normally, the camera will keep the same focus set with the autofocus button.

So here's a shot of the camera view screen with the dome lights on, after I've done all of the above steps:

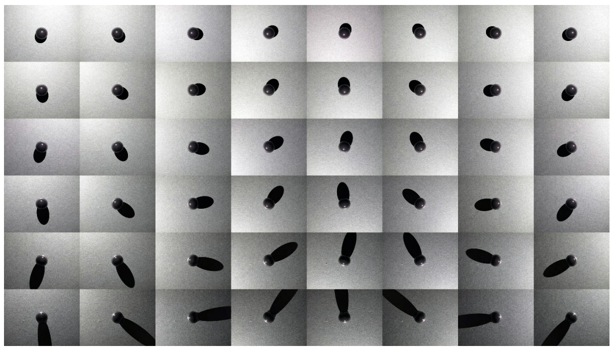


Now it's time to set appropriate values for the LED and Delay times. The LED time, set by the left knob control, determines how long the LED is on when taking the picture. Too short, and the LED may turn off before the picture is taken; too long, and it will extend the time for the photo acquisition needlessly. The Delay time is to allow the camera to store the image on the SD card, and then display it on the screen so that you can check it. Too short and the camera won't be able to keep up; too long, and once again the total acquisition time will be longer than necessary.

Start by setting both knobs in their middle position. Press the red Action button to start the Automatic process. If the LED time is too short, and you're only getting black pictures, make the LED time longer. If the LED is on for too long, reduce the LED time bit by bit until you wind up getting a black picture, then turn it back up to a longer length. After you set the correct LED time, work on the Delay time in a similar fashion. You may have to go back and forth between the two knobs until you find the right balance of times that takes the picture, but doesn't waste too much time. With time, you'll figure out roughly where to set these knobs as you change the exposure parameters.

When you've got the times set the way you want, stop the running Auto mode by pressing the Reset button. When the controller has rebooted, start up the Auto mode again, and hopefully you'll see something like this (sped up to minimize the viewing time; it's not that exciting) video.

Once the run is done, you should have 48 photos of the black sphere, each taken at a different lighting angle:



Transfer all these image files to a folder on your computer.

The position of each highlight on the sphere is related to the lighting angle when the photo was taken. Using that, we can create a calibration file that will allow you to process any set of photos from this system easily, as long as the camera is oriented in the same position, or close to it. That's true for any camera you might use. So make some kind of reference mark on the dome to indicate how the camera should be oriented. If for some reason you need to have absolute precision for this calibration, you can just do a calibration run for the first photo set, then use that calibration data for all subsequent photos sets you shoot of the objects of interest (as long as you don't move the camera).

To extract the calibration info, we'll need to use a free program called RTIBuilder, available from Cultural Heritage Imaging. You'll also need to have Java installed on your computer.

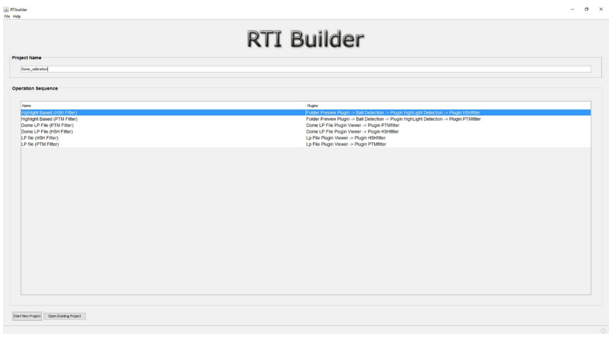
Install the program on your computer. While you're at it, you might as well download and install the RTIViewer program, which you'll use to actually view your RTI datasets. It's also available at the CHI website.

**Note:** Both of these programs are free, freely distributable and open source. CHI has done indispensable work in developing the RTI technique, including making these programs available for free (unlike one European university I could mention). If you use their software on a regular basis, make a donation, preferably on a yearly basis; I do.

After installation, run RTIBuilder. Be advised, the program is a little flakey; a new version is in the works, but the release date is unknown. The biggest quirk is that any filenames or filepaths you use must not have any spaces in them.

Sometimes you'll get a warning, other times you'll just get cryptic error message. When you first run the program, you'll get the startup screen. Enter a name for the project at the top (no spaces), and select the Highlight Based (HSH Fitter) option.

The Start New Project button at the bottom should now be active, click it. You'll get this window:

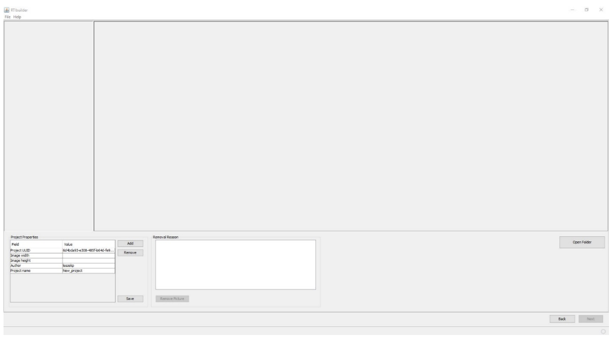


Project Name

Dome_calibration

Operation Sequence

| Name | Plugins |
|---|---|
| Highlight Based (HSH Fitter) | Folder Preview Plugin -> Ball Detection -> Plugin HighLight Detection -> Plugin HSHfitter |
| Highlight Based (PTM Fitter) | Folder Preview Plugin -> Ball Detection -> Plugin HighLight Detection -> Plugin PTMfitter |
| Dome LP File (PTM Fitter) | Dome LP File Plugin Viewer -> Plugin PTMfitter |
| Dome LP File (HSH Fitter) | Dome LP File Plugin Viewer -> Plugin HSHfitter |
| LP file (HSH Fitter) | Lp File Plugin Viewer -> Plugin HSHfitter |
| LP file (PTM Fitter) | Lp File Plugin Viewer -> Plugin PTMfitter |



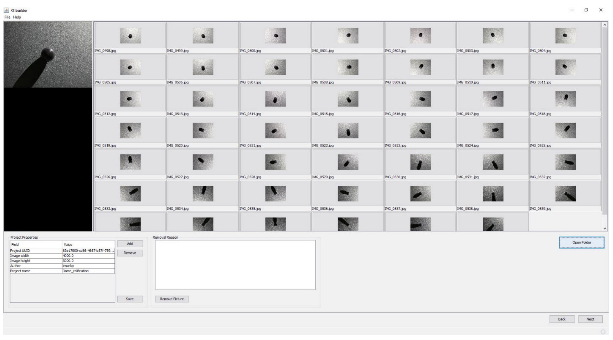
Project Properties

| Field | Value |
|---|---|
| Project UUID | 6d4bda93-e308-485f-b64d-fe9... |
| Image width | |
| Image height | |
| Author | leszekp |
| Project name | New_project |

What it wants is the location of the sphere pictures you took. But time for a few quirks of RTIBuilder. First, you need to create a project folder to hold the pictures; give it any name you want, except make sure that there are no spaces in the folder name or drive path. Secondly, you'll need to create a subfolder inside the project folder named "jpeg-exports", no quotes, dash mandatory.

Copy all the sphere highlight photos into this folder. Finally, if the photos have a capitalized "JPG" extension instead of "jpg", the program will give a fatal error message at the very last step. So you may need to rename all of the pictures with a "jpg" file extension. In Window, the simplest way to do this is to shift-right-click on the folder, choose "Open command window here", then type "ren *.JPG *.jpg" in the window that pops up. Don't know how to do this with a Macintosh, but I'm guessing there's a way.

Once you've taken care of that, in RTIBuilder click the "Open Folder" button, then choose the project folder from the "Open" dialog; don't choose the "jpeg-exports" folder. The program should then load in all the sphere highlight pictures in the folder:



Project Properties

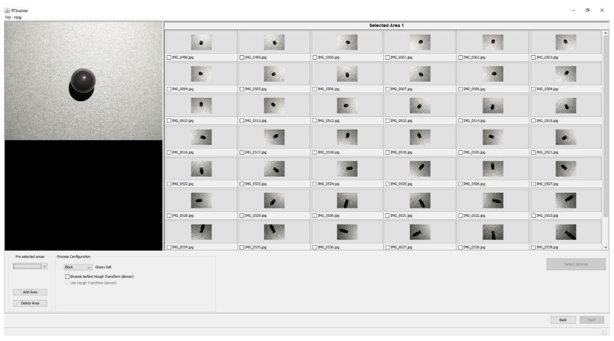
| Field | Value |
|---|---|
| Project UUID | 63e17000-cd66-4667-b57f-759... |
| Image width | 4000.0 |
| Image height | 3000.0 |
| Author | leszekp |
| Project name | Dome_calibration |

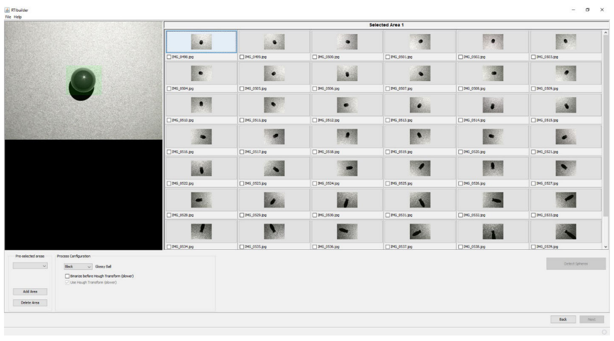
Make sure that all the pictures came out clear and in focus; if not, you'll just have to shoot another set. Click on Next when ready, and you'll get this screen:

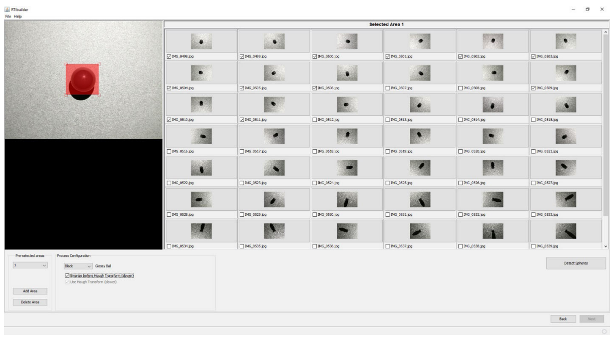
Click and drag in the picture at upper left to draw a green square that covers the sphere plus a small margin on the outside:

Click on the "Add Area" button at lower left:

The area you defined will now be in red. If you want to refine that selection, you can click and drag the controls on the corners of the red area. But as long as you have the entire sphere selected, plus a little bit of margin, you should be fine. There's a check box for "Binarize before Hough transform (slower)" at lower left. This improves



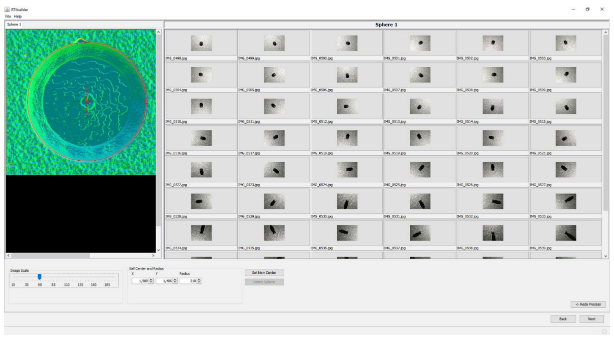




threshold detection, but not sure exactly how.

Nevertheless, I usually check this box – slower is better, right?

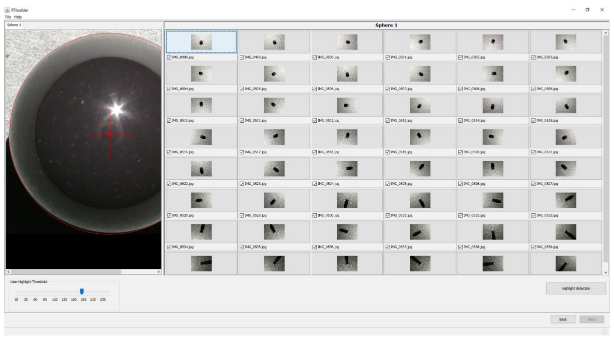
When done, click the “Detect Spheres” button at lower right. Next screen:



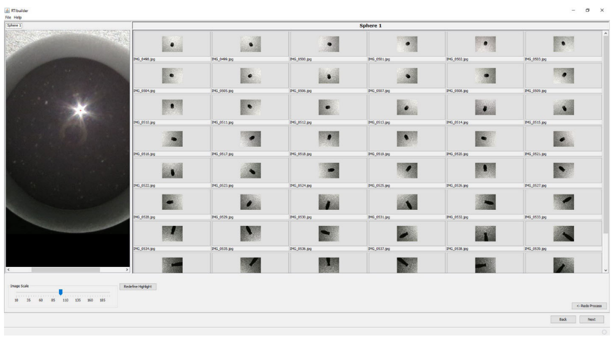
The program has analyzed all the pictures, and shown you its best guess as to the position of the edge of the sphere against the background, and its center. Use the slider at lower left to change the size of the image. Clicking any of the thumbnails will put that image inside the viewer; try clicking on a number of pictures to see whether the program’s guess for the radius/center of the sphere was correct.

If it’s a bit off, you can correct it using the X/Y/Radius controls to modify those parameters, then clicking on “Set New Center” when done. Caution: if the Radius value is less than 250, the program may have problems getting an accurate calibration; you may need to reshoot the photo set with either a higher zoom factor or a larger black sphere.

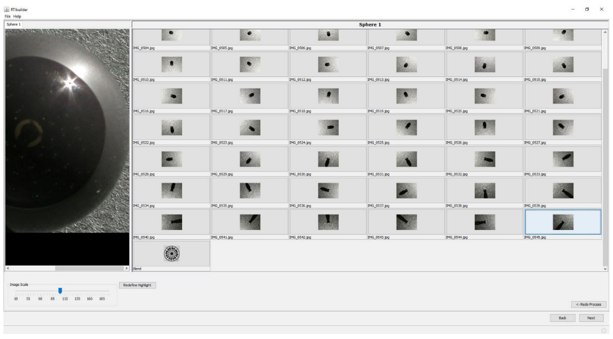
When you’re happy at this stage, click the Next button:



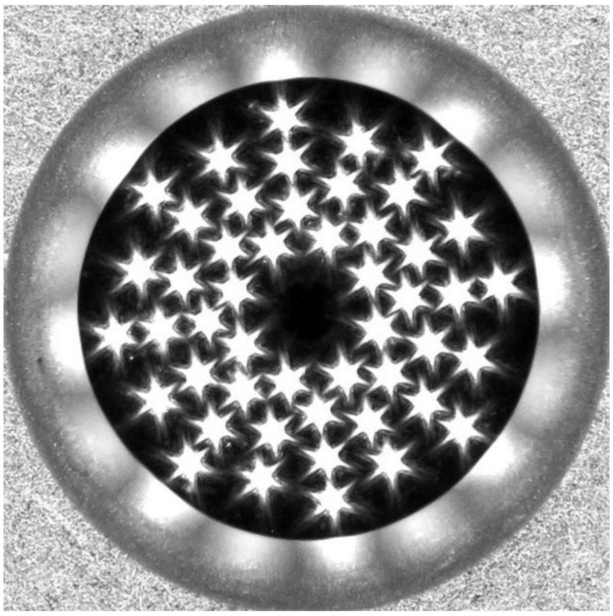
The radius and center will still be displayed here; if you decide you’re unhappy here, just click on the Back button to go to the previous screen. Otherwise, click on the “Highlight detection” button; the program will now examine the spheres and find the exact center position for every highlight on every picture. When done, you’ll see this:



Notice that the radius/center marks are now gone, replaced with a small cross inside of the highlight; this is where the program thinks the center is. In this case, it looks pretty good. Clicking on other thumbnails will bring up a different picture, and it’s worth checking a few of these to make sure that the program has done a good job:



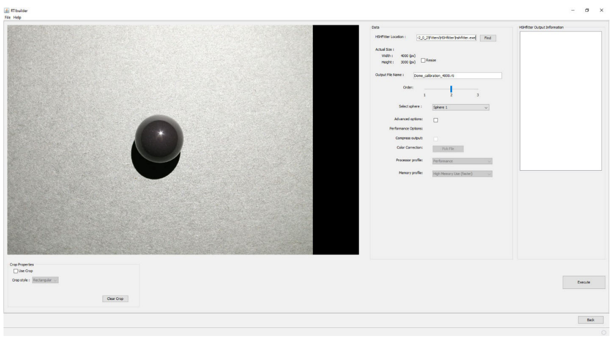
If the results aren't satisfactory, click on the "Redo Process" button, and try setting a different value for the "User Highlight Threshold". I've never had to do this, so I don't know how well that works. After doing the highlight detection, the program adds one more image to the set of the thumbnails, called "Blend":



This is a blend of all the images, showing the highlights reflecting the distribution of LEDs inside the dome.

Why the star pattern? Diffraction from the shutter, accentuated because you're using a small aperture for the best focus. I've never had a problem with this, but if you're having problems getting good highlight detection, try shooting a calibration photo set with a slightly larger aperture.

When you're happy with highlight detection, click Next:



HSHFitter Location : …2_0_2\Fitters\HSHfitter\hshfitter.exe

Actual Size :
Width : 4000 (px)
Height : 3000 (px) ☐ Resize

Output File Name : Dome_calibration_4000.rti

Order: 2

Select sphere : Sphere 1

Advanced options: ☐

Performance Options:

Compress output:

Color Correction: Pick File

Processor profile: Performance

Memory profile: High Memory Use (faster)

☐ Use Crop

Crop style : Rectangular

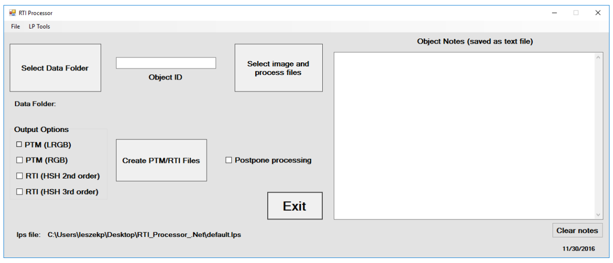
Leave all settings as is, and click Execute. The program will generate an lp calibration file for these photos, and then process them. If you look in the project folder, you'll now see a folder called "assembly-files". Look in there, and you'll see two .lp files (both identical), which combine the calibration data with the names and paths of the photos, used for processing. Here's a few lines from one of them:

```
48
H:\RTI_Data\Hackaday_dome_calibrations\write_up_3\jpeg-exports\IMG_0498.jpg 0.
↪13417545 0.45608342 0.8797641
H:\RTI_Data\Hackaday_dome_calibrations\write_up_3\jpeg-exports\IMG_0499.jpg -0.
↪22573107 0.40408084 0.88643336
H:\RTI_Data\Hackaday_dome_calibrations\write_up_3\jpeg-exports\IMG_0500.jpg -0.
↪46390244 0.1131883 0.8786256
H:\RTI_Data\Hackaday_dome_calibrations\write_up_3\jpeg-exports\IMG_0501.jpg -0.
↪4129351 -0.24111442 0.8782644
H:\RTI_Data\Hackaday_dome_calibrations\write_up_3\jpeg-exports\IMG_0502.jpg -0.
↪12138778 -0.4695002 0.8745482
[...]
```

First line is the number of photos, remaining lines start with the photo name and path, and angle information appended to the end.

What we need to do now is extract out the number and angle information from this file, so that we can then append it to any new photo dataset created with the RTI-Mage; once again, that will be valid for every photo set where the camera is oriented the same way it was when you shot the calibration set. To do this, I've created a simple utility called RTI_Processor, which you can find in the file section; unzip the folder to any location, then run the RTI_Processor.exe program file (no installation required). Sorry Mac people, but only Windows for now.

Where's the source code? This is a preliminary VB .Net version, replacing an older VB6 version, but it's still not clean, and is missing some new features I want to add. When that is done, I'll release the source code. It may be possible to compile .Net programs to work with Macs, but I don't know enough about this to say for sure.



lps file: C:\Users\leszekp\Desktop\RTI_Processor_.Net\default.lps

11/30/2016

Go to the LP Tools menu, and select "Create lps from lp". Navigate to the "assembly-files" folder, then select and open one of the two .lp files. You'll then get the option to save a new file with the .lps extension. Give it a descriptive name, and save it in the same folder as the RTI_Processor program.

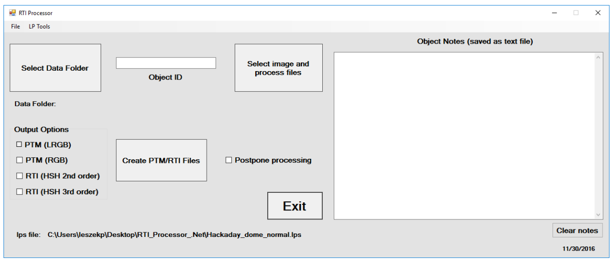
What's this .lps file? It's very similar to the .lp file, except that all the filenames and paths have been stripped out:

```
48
0.13417545 0.45608342 0.8797641
-0.22573107 0.40408084 0.88643336
-0.46390244 0.1131883 0.8786256
-0.4129351 -0.24111442 0.8782644
-0.12138778 -0.4695002 0.8745482
0.24190147 -0.4294567 0.87008655
0.45700282 -0.13436723 0.87925756
0.40734085 0.2106088 0.88866043
```

As with the .lp file, .lps files can be opened in any text editor.

One final step. In RTI_Processor, open this new lps file with File -> Open (or Ctrl-O). Select the lps file you've just saved; this loads it into the program as the set of angle positions it will use for subsequent processing. If this is the main dome you will be using, choose "Save current lps as default", and it will load this dataset in automatically whenever the program starts.

If you use multiple domes, you'll need to make sure you're using the right data for the right dome in this program, which may require you to specifically load in the correct lps file for that specific dome every time. Whenever you load in a custom lps file, it will show the name at the lower left:



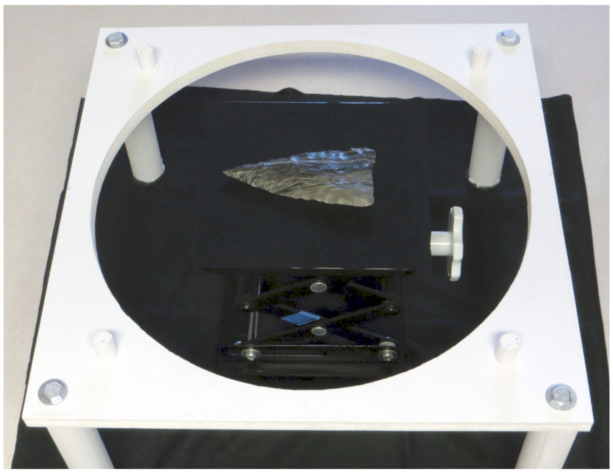
lps file: C:\Users\leszekp\Desktop\RTI_Processor_.Net\Hackaday_dome_normal.lps

11/30/2016

The default.lps file will always show up there when you start the program. If you load in a different lps file, then want to go back to the default, there's a menu item under File that will let you do that.

Calibration is out of the way, and for the most part you only have to do this once (I'll discuss one exception in a future step). Now you're ready to do RTI on your first real object. You'll need this RTI Processor program to handle data, so don't get rid of it.

Been a long road to this point, but you're now ready to photograph your first real object, convert the photos into an RTI dataset, then view it. Start by putting your object on the sample stage, with the top roughly at the same position as the bottom of the dome. Here, I'm using a modern replica projectile point:



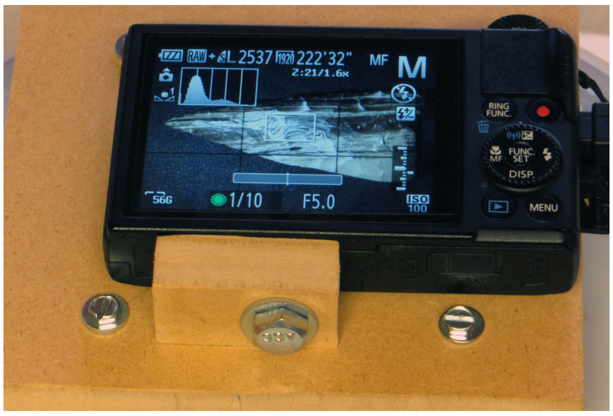
Many of the following steps are cut-and-paste from the previous calibration step.

Time to put the rest of the RTI-Mage system parts in place:

1. Dome on top of the stand.
2. Ethernet cables from the dome plugged into the control box.
3. Camera in position on top of the dome (my Canon S110 in this case)
4. A micro-USB cable running from the USB Shutter jack on the rear of the control box to the USB input of the camera (this will fire the shutter automatically using CHDK). If you were using a different remote mode, like the IR remote cable, you would have that plugged into the same jack, with the LED pointed at the IR sensor of your camera. You can also do this procedure in Manual mode.
5. 9-12V DC power plugged into the control box.
6. Correct mode and shutter setting. In this case, I'll be running in Auto mode, with the shutter set in USB mode for firing the Canon.

Once the system is setup and ready to go:

1. Set the camera to its Manual setting, where you set fixed exposure time and aperture values.
2. Press the black WB button, which turns on the top row of LED lights for a prolonged period. Use this lit time to:
   (a) Center the object in the camera view screen by moving the lab jack or dome around.



   (b) If possible, zoom in the camera as much as you can to get the largest possible image of the sphere.

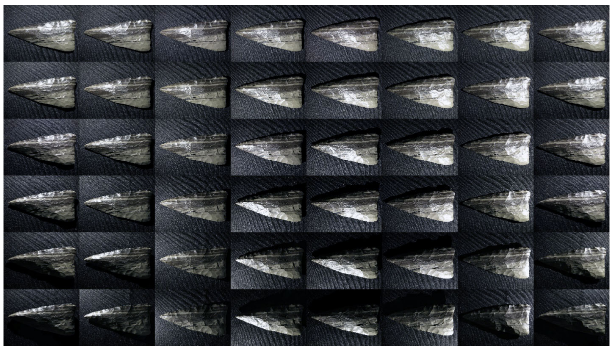
   (c) Focus on the sphere by partially depressing the shutter button.

   (d) If you have a gray card, set the camera white balance (WB).

3. Determine the proper exposure settings. Set the aperture large for short exposure times, small for greater depth of field (at the cost of longer exposure times). Then set the exposure time so that the object is properly exposed. If your camera has a histogram display, use it to help figure out the correct exposure settings.

4. Set the JPG picture quality to the highest level. If you can shoot in RAW format, do that.

5. You will want to use your camera's manual focus mode. Auto focus can work sometimes, but it will take longer to acquire the full dataset since the camera will have to re-focus for every picture. And for shallow lighting angles, autofocus can sometimes have problems because of the reduced lighting intensity. What I normally do with my Canon cameras is an initial autofocus with a half-press of the button, then switch over to manual focus mode. Normally, the camera will keep the same focus set with the autofocus button.

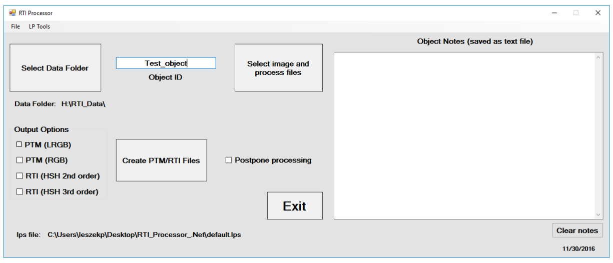


Now it's time to set appropriate values for the LED and Delay times. The LED time, set by the left knob control, determines how long the LED is on when taking the picture. Too short, and the LED may turn off before the picture is taken; too long, and it will extend the time for the photo acquisition needlessly. The Delay time is to allow the camera to store the image on the SD card, and then display it on the screen so that you can check it. Too short and the camera won't be able to keep up; too long, and once again the total acquisition time will be longer than necessary.

Start by setting both knobs in their middle position. Press the red Action button to start the Automatic process. If the LED time is too short, and you're only getting black pictures, make the LED time longer. If the LED is on for too long, reduce the LED time bit by bit until you wind up getting a black picture, then turn it back up to a longer length. After you set the correct LED time, work on the Delay time in a similar fashion. You may have to go back and forth between the two knobs until you find the right balance of times that takes the picture, but doesn't waste too much time. With time, you'll figure out roughly where to set these knobs as you change the exposure parameters.

When you've got the times set the way you want, stop any running Auto mode by pressing the Reset button. When the system has rebooted, start up the Auto mode again, and hopefully you'll see something like this (sped up to minimize the viewing time; it's not that exciting) video.

This process will generate a full set of pictures at different lighting angles:



RTI Processor

Object ID: Test_object

Data Folder: H:\RTI_Data\

Output Options
- ☐ PTM (LRGB)
- ☐ PTM (RGB)
- ☐ RTI (HSH 2nd order)
- ☐ RTI (HSH 3rd order)

☐ Postpone processing

lps file: C:\Users\leszekp\Desktop\RTI_Processor_.Net\default.lps

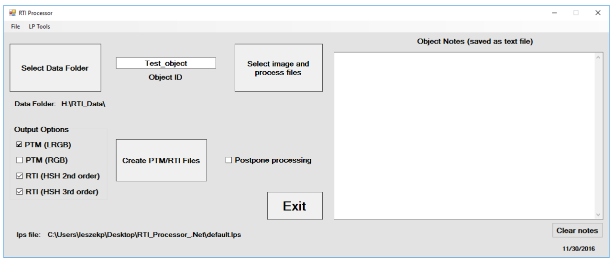
11/30/2016

Copy these pictures over to your computer.

You'll be using the RTI_Processor software, and for the most part it's ready to go. There's one exception. The program is a front-end for command line programs that take the photos, and convert them into RTI datafiles. There are two types of these files. PTM files have a binomial quadratic fit to the light curve at every pixel; RTI files fit 2nd and 3rd order Legendre polynomials to that curve. RTI files are a better fit, but PTM files have more visualization options in the RTIViewer software. You can create both of them with RTI_Processor as long as the command line programs are in the directory.

The RTI fitter, called hshfitter.exe, is open source and is included in the RTI_Processor folder you downloaded from the Hacakday Files section. The PTM fitter, ptmfitter.exe is not open source; HP makes it freely available, but you have to go to the HP website to download it. Go to HP Labs, accept the licensing agreement, then download the Windows PTM Fitter zip file. Unzip the contents into the RTI_Processor folder, and you're good to go.

Start up the RTI_Processor software; make sure the proper .lps calibration file is loaded in. In this case, it's the default file loaded in upon program start, but you can choose another one using the File menu.



RTI Processor

Object ID: Test_object

Data Folder: H:\RTI_Data\

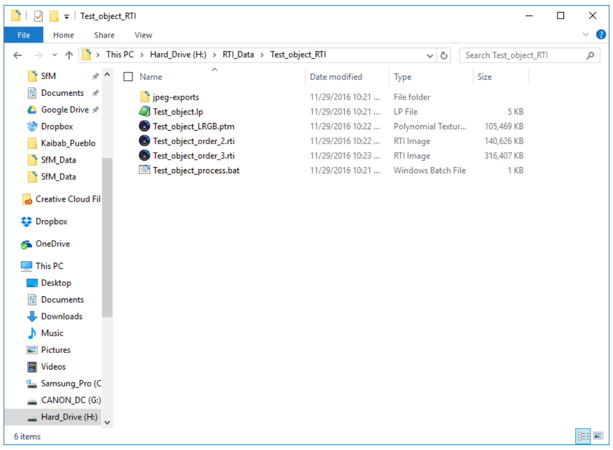
Output Options
- ☑ PTM (LRGB)
- ☐ PTM (RGB)
- ☑ RTI (HSH 2nd order)
- ☑ RTI (HSH 3rd order)

☐ Postpone processing

lps file: C:\Users\leszekp\Desktop\RTI_Processor_.Net\default.lps

11/30/2016

First, click on the "Select Data Folder", and choose the location where you want to store the processed files. This location must have no spaces in its name. Unless you need to change this location, you only have to select it once. The program will check to make sure you've entered a Data Folder before letting you go on anywhere else. Next, enter some kind of ID name for the object; no spaces allowed. Here's how my program looks. I've loaded a special lps file (see in the lower left hand corner), I have H:RTI_Data as the data folder, and the name of the object is Test_object.



Test_object_RTI

This PC › Hard_Drive (H:) › RTI_Data › Test_object_RTI

| Name | Date modified | Type | Size |
|---|---|---|---|
| jpeg-exports | 11/29/2016 10:21 ... | File folder | |
| Test_object.lp | 11/29/2016 10:21 ... | LP File | 5 KB |
| Test_object_LRGB.ptm | 11/29/2016 10:22 ... | Polynomial Textur... | 105,469 KB |
| Test_object_order_2.rti | 11/29/2016 10:22 ... | RTI Image | 140,626 KB |
| Test_object_order_3.rti | 11/29/2016 10:23 ... | RTI Image | 316,407 KB |
| Test_object_process.bat | 11/29/2016 10:21 ... | Windows Batch File | 1 KB |

6 items

Click on the "Select Image And Copy Files" button, and navigate to the folder that has the photographs taken with the RTI-Mage system. Note: photos must be in JPG format. Double-click on any picture, and the following will happen:

1. The program will create a subfolder in the data folder with the Object ID name, plus "_RTI" appended to it.
2. It will create a jpeg-exports subfolder in that folder, copying all the JPG files there, renaming them by Object ID and photo number, and making sure the file extension is lower case (jpg, not JPG).

You'll get a pop-up when all the files have been copied over.

Next, you need to choose what output files you want to get. There are four choices:

**PTM (LRGB)** Creates a file that fits a single binomial quadratic curve to just the luminosity, and assumes the RGB color values don't change with light. This would be the normal choice for PTM file output. On Windows systems, the largest image size that can be processed with this option is 24 megapixels; if your photos are larger than this, you'll need to crop or resize all of them to use this option successfully.

**PTM (RGB)** Fits separate light curve files to the R,G,B colors of every pixel. This is only useful for objects whose color changes with lighting angle. On Windows systems, this only works with smaller images (< 8 megapixels), so you'll need to crop or resize all your images.

**RTI (HSH 2nd order)** Fits a 2nd order Legendre polynomial (9 coefficients) to the light curve for every pixel.

**RTI (HSH 3rd order)** Fits a 3rd order Legendre polynomial (16 coefficients) to the light curve for every pixel.

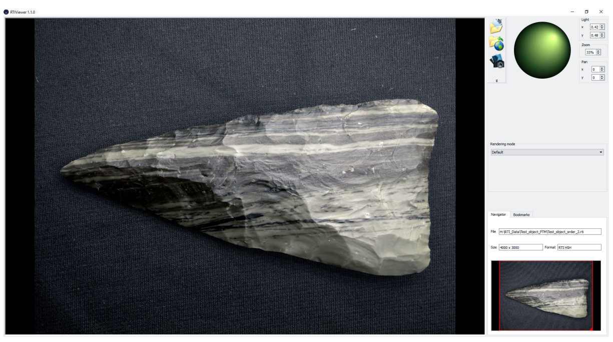
You can choose any/all of these by checking the boxes next to them. However, for most cases, I only select PTM (LRGB) and RTI (HSH 2nd or 3rd order). I haven't photographed a single object whose colors change with lighting angle, so PTM (RGB) is pointless. RTI (HSH 3rd order) can sometimes give better results than just 2nd order, but not always.

There's also a box at left where you can enter information about the object; it's saved as a text file in the working directory. I put this in, but never use it – the .Net version will eventually have a more sophisticated metadata file capability.

Final option: the Postpone processing box. If this is unchecked, the RTI datafile creation process starts immediately. If checked, the commands are added to a batch file called "master_batch.bat", located in the root Data Folder directory.

Depending on the size of the images, it can take a few minutes to process these datafiles. By postponing process, you can line up a queue of files to be processed, and when you're ready, run the master.bat program to process all of them at the same time, say overnight.

For this example, I'll process a PTM and RTI output file right away:



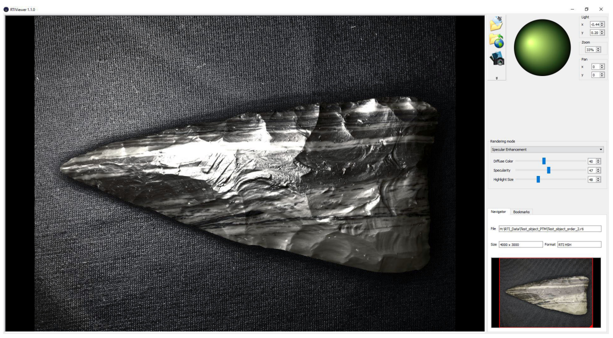
Light x 0.42 y 0.48
Zoom 33%
Pan x 0 y 0
Rendering mode: Default
File: H:\RTI_Data\Test_object_PTM\Test_object_order_2.rti
Size: 4000 x 3000 Format: RTI HSH

Push the Create PTM/RTI Files button, and it will run a batch file that will generate the RTI data files; you'll see a command window open up, and watch the progress of the processing. When the window closes, the processing will be done, and you'll see the following files/folders in the working directory:

**jpeg-exports** As mentioned above, contains the copied renamed photographs from your RTI dome run.

**Test_object.lp** The light position file used by the fitting programs. This is plain text, so you can view it in any text editor.

**Test_object_LRGB.ptm** The PTM datafile. If you've installed the RTIViewer program, double-clicking on it will load it into the viewer.

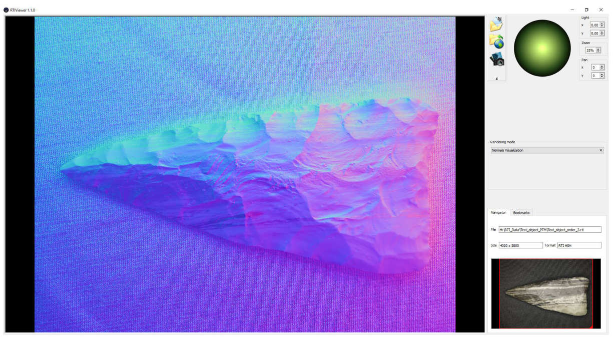


**Test_object_order_2.rti** The 2nd order RTI datafile.

**Test_object_order_3.rti** The 3rd order RTI datafile.

**Test_object_process.bat** The batch file used to run the PTMFitter and HSHfitter programs. You can delete this once processing is complete.

Double-clicking will load an RTI or PTM datafile into the RTIViewer program, where you can check out all the viewing options (see the RTIViewer manual for more information on using these):



That's it – repeat as often as desired. Have fun!