TinyCoreDoc

Release 0.0.3

Feb 14, 2020

Contents

Contents:

CHAPTER 1

Getting Started

1.1 Board Intro

TinyCore Series include TinyCore 16 (Attiny1616 breakout board), TinyCore 32 (Attiny3217 breakout board) and TinyCore Programmer. TinyCore is miniature prototpying board with common peripherals like I2C, SPI, UART. It also has PWM, Timers, Touch PINS, ADC, DAC, 16K / 32K Flash, 2K SRAM, 256 bytes EEPROM with 8-bit CPU running up to 20MHz all in its tiny body! It has Arduino Support and open source libraries. They are maker & hacker friendly.

Specifications	.
Flash (program memory)	32/16 KB
RAM	2 KB
EEPROM	256 bytes
Bootloader	No
GPIO Pins	18
ADC Channels	10
PWM Channels	3
Peripheral	USART, SPI, I2C, Touch
Clock	Up to 20 MHz
Power Consumption	min 2.9μA, max 10mA

Below is the overview of the TinyCore 16 Breakout Board:

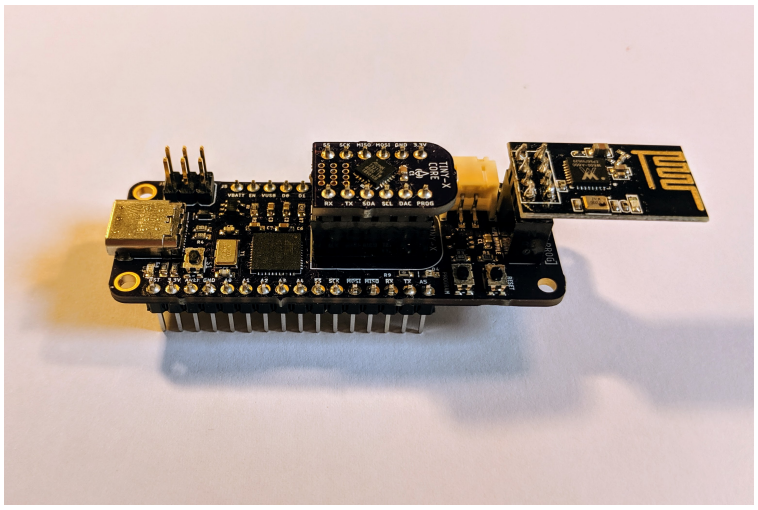
1.1.1 TinyCore 16 Pinout

1.1.2 TinyCore 32 Pinout

1.2 Installation

Hardware:

- Solder Header pins on both TinyCore 16/32 and TinyCore Programer.
- Plug in TinyCore 16 or 32
- Plug in Optional ESP8266 or W600 module
- Connect a USB-C cable and plug into your computer



Driver:

- Windows 10/Linux/Mac doesn't need a driver for TinyCore Programmer
- For Windows 7 or lower, please try to download this file to install driver properly. `Windows Driver`

Software Through Arduino IDE:

- Install the current upstream Arduino IDE at the 1.8.7 level or later. The current version is at the Arduino Website.
- Start Arduino and open Preferences window.
- Enter `https://raw.githubusercontent.com/xukangmin/TinyCore/master/avr/package/package_tinycore_index.json` into *Additional Board Manager URLs* field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install *TinyCore* platform (and don't forget to select your TinyCore board from Tools > Board menu after installation).

1.3 Blink!

A LED on the TinyCore Programmer board is connected to the DAC pin marked on the board silkscreen, copy below code to Arduino IDE and upload sketch:

Blink Code:

```
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);                       // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);                       // wait for a second
}
```

1.4 Compile & Upload

Select Board TinyCore 16 or TinyCore 32 in the board manager and click and compile the code.

There are several ways to upload sketches.

- **Upload through TinyCore Programmer**
  - Select related com port number and click upload directly, same as programming Arduino Uno
- **Upload through Arduino Uno or similar boards.**
  - Connect Arduno Uno and Tiny Core boards as described here AND follow the instructions for "Building with Arduino IDE."
  - Arduin IDE->Tools->Programmer->Arduino PIN as UPDI
  - Arduin IDE->Sketch->Upload using programer
- **Upload through Atmel-ICE Programmer**
  - Connect Atmel-ICE UPDI, VTG and GND PIN to TinyCore, supply TinyCore with external voltage, either 3.3V or 5V.
  - Make sure the green light on Atmel-ICE is on, that means target is properly powered
  - Arduin IDE->Tools->Programer->Atmel-ICE-UPDI
  - Arduin IDE->Sketch->Upload using programer

You should see an orange LED blinking on the TinyProgrammer Board.

Now You officially start your TinyCore Journal, Congratulations!

CHAPTER 2

Using TinyCore

2.1 GPIO

Follow the Arduino syntax, GPIO can be controlled as output or input.

Configured as output:

```
pinMode(pin, OUTPUT);
digitalWrite(pin, HIGH);
```

You can check the pinout for reference:

The Arduino Pin is marked as purple. To use a specific pin, for example:

```
void setup()
{
    pinMode(LED_BUILTIN , OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN , HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN , LOW);
    delay(1000);
}
```

Reference: https://www.arduino.cc/en/Tutorial/DigitalPins

2.2 PWM

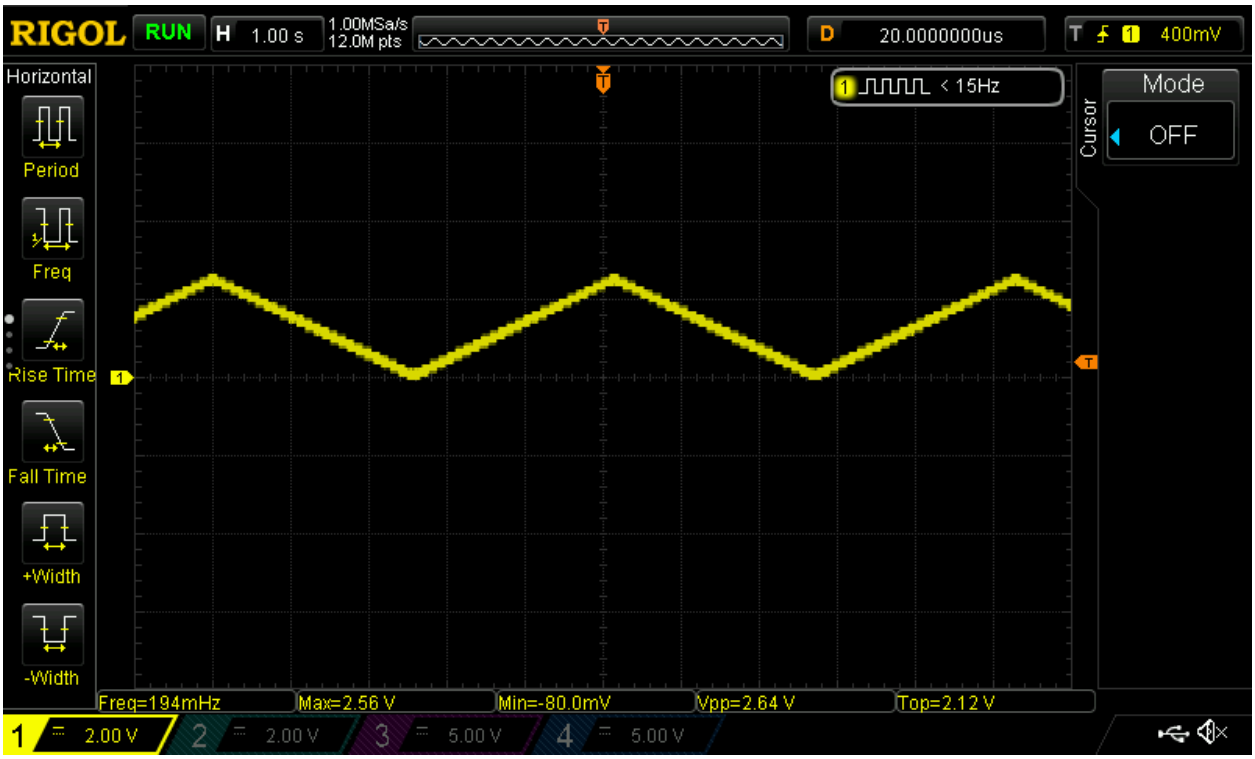
Same as Arduino, use analogWrite to generate PWM signals.

DAC PIN (or PIN 15) on TinyCore16/32 boards are true voltage output instead of PWM signals.

Simply use analogWrite on the DAC pin.:

```
int val = 100; // (0 to 255)
analogWrite(LED_BUILTIN, val);
```

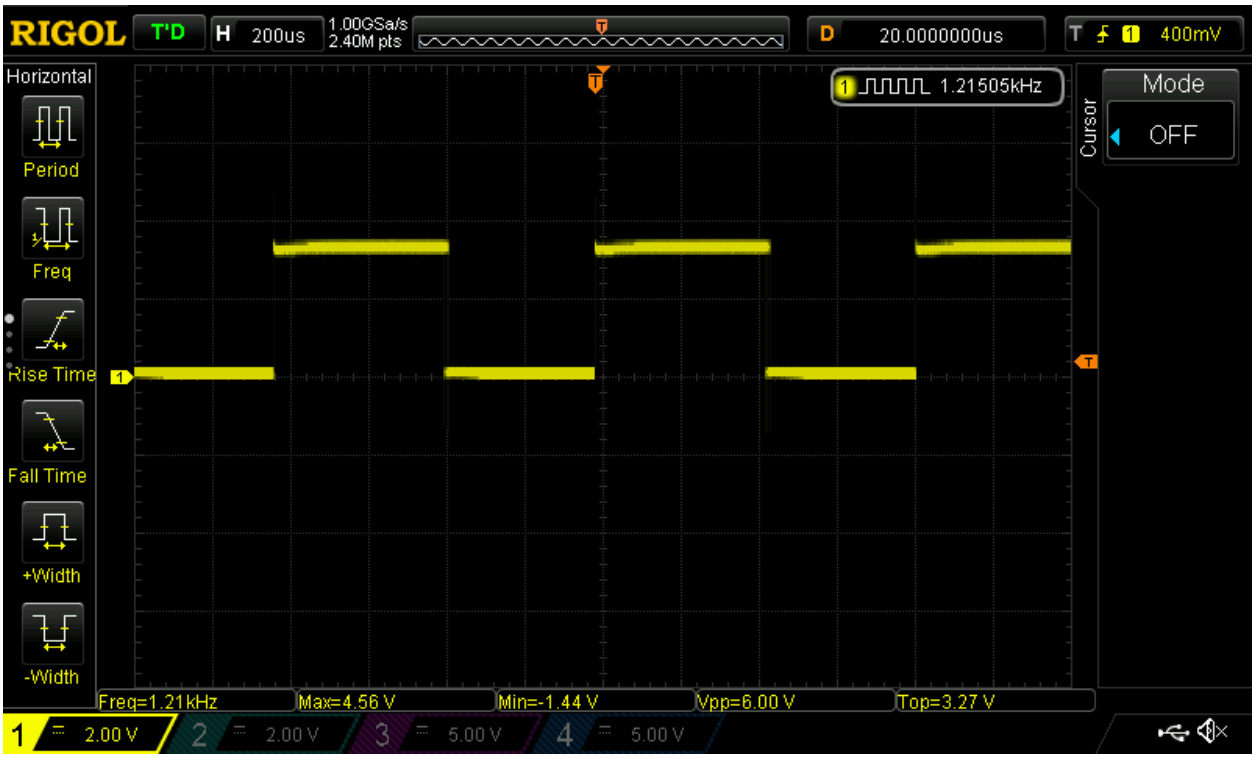
See DAC result below.



PWM pins includes PIN 1, 2, 3, 12, 13, 14, use analogWrite on these pins to generate PWM signals:

```
int val = 100;
analogWrite(1, val);
```

See PWM result below.



Reference: https://www.arduino.cc/en/Tutorial/PWM

2.3 ADC

There are 11 pins on TinyCore that support AnalogRead.

Available pins are marked in green color from A0 to A11. Some of the pins are in the extended pin area.

Use analogRead(A0) to read values from pins.

Reference: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

2.4 DAC

TinyCore has 1 pin support real DAC output, which is pin 15.

The default DAC reference voltage is set as 2.5V.

You can change reference voltage in the board.txt file.

- Find -DDACVREF
- **Use the lookup table below**
  - 0.55V -DDACVREF=0
  - 1.1V -DDACVREF=1
  - 1.5V -DDACVREF=4
  - 2.5V -DDACVREF=2

- 4.3V -DDACVREF=3

Note: when power supply voltage is 3.3V, 4.3V reference is not working properly, you need to supply 5V to use 4.3V.

2.5 Servo

Use Servo library like in Arduino. Servo should be available on all PWM pins (1,2,3,12,13,14).

2.6 I2C

Fully Implement Wire library. SCL/SDA pins are marked on the board which is pin 3 and pin 2. It is also possible to use alternative pins. (10/PA1 = SDA, 11/PA2 = SCL)

To enable alternative pins, simply put the following code at the beginning of setup before Wire.begin():

```
#include <Wire.h>

void setup() {
    Wire.useAlternatePins();
    Wire.begin();
}
```

Another option is changing the pins_arduino.h file located in the variants folder, change:

```
#define TWI_MUX             (PORTMUX_TWI0_DEFAULT_gc)
```

To:

```
#define TWI_MUX             (PORTMUX_TWI0_ALTERNATE_gc)
```

This option is only for advanced user or if you want to make your customized board based on Attiny1616/3217.

Simple Reader:

```
#include <Wire.h>

void setup() {
    Wire.begin();        // join i2c bus (address optional for master)
    Serial.begin(9600);  // start serial for output
}

void loop() {
    Wire.requestFrom(8, 6);    // request 6 bytes from slave device #8

    while (Wire.available()) { // slave may send less than requested
        char c = Wire.read(); // receive a byte as character
        Serial.print(c);         // print the character
    }

    delay(500);
}
```

Simple Writer:

```
#include <Wire.h>

void setup() {
    Wire.begin(); // join i2c bus (address optional for master)
}

byte x = 0;

void loop() {
    Wire.beginTransmission(8); // transmit to device #8
    Wire.write("x is ");        // sends five bytes
    Wire.write(x);              // sends one byte
    Wire.endTransmission();    // stop transmitting

    x++;
    delay(500);
}
```

receiveEvent and requestEvent are also supported.

2.7 UART

uart support up to 115200 baud rate.

To use uart, simply initialize it as:

```
Serial.begin(9600);
```

Then you can send or receive data:

```
Serial.println("test");
Serial.write(0x0d)
```

2.8 SPI

Same as Arduino:

```
#include <SPI.h>

byte address = 0;
byte value = 0x55;

void setup() {
    SPI.begin();
}

void loop() {
    SPI.transfer(address);
    SPI.transfer(value);
}
```

Reference: https://www.arduino.cc/en/reference/SPI

2.9 EEPROM

There are total 256 bytes eeprom on TinyCore 16/32 boards.

Usages are the same as Arduino:

```
#include <EEPROM.h>

int val = EEPROM.read(addr);

EEPROM.write(addr, val);
```

Check example code for more details.

Reference: https://www.arduino.cc/en/Reference/EEPROM

Now let's use these to build awesome tiny projects!

2.10 Touch

TinyCore 16 has 12 pins available for touch configuration, TinyCore 32 has 15 pins available for touch configuration.

To use touch pins, includes TinyTouch.h and follow the example code below:

```
#include "TinyTouch.h"

TinyTouch touch;

uint8_t touchPins[2] = {13, 2}; //initialize touch pins

void setup() {
    touch.begin(touchPins, sizeof(touchPins));

    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    test.touchHandle();

    if (test.getValue(0) > 800 or test.getValue(1) > 800) {
        digitalWrite(LED_BUILTIN, HIGH);
    } else {
        digitalWrite(LED_BUILTIN, LOW);
    }
}
```

CHAPTER 3

Projects Demo

3.1 Breathing LED

LED on the TinyProgrammer board will breath in a constant pace.

Here we use DAC pin to output various voltages.

breathing_led.c

```
int i = 0;
int dir = 1;
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    if (dir)
    {
        i++;
    } else {
        i--;
    }
    if (i >= 255) {
        dir = 0;
    } else if (i <= 150){
        dir = 1;
    }
    analogWrite(LED_BUILTIN, i);
    delay(10);
}
```

3.2 Controlling NeoPixel String

You need to first install Adafruit NeoPixel Library.

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN 12

#define NUM_LEDS 30

#define BRIGHTNESS 50

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUM_LEDS, PIN, NEO_GRBW + NEO_KHZ800);

byte neopix_gamma[] = {
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,
    1,  1,  1,  1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,
    2,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4,  4,  5,  5,  5,
    5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  9,  9,  9, 10,
10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16,
17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 24, 25,
25, 26, 27, 27, 28, 29, 29, 30, 31, 32, 32, 33, 34, 35, 35, 36,
37, 38, 39, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50,
51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68,
69, 70, 72, 73, 74, 75, 77, 78, 79, 81, 82, 83, 85, 86, 87, 89,
90, 92, 93, 95, 96, 98, 99,101,102,104,105,107,109,110,112,114,
115,117,119,120,122,124,126,127,129,131,133,135,137,138,140,142,
144,146,148,150,152,154,156,158,160,162,164,167,169,171,173,175,
177,180,182,184,186,189,191,193,196,198,200,203,205,208,210,213,
215,218,220,223,225,228,231,233,236,239,241,244,247,249,252,255 };


void setup() {
// This is for Trinket 5V 16MHz, you can remove these three lines if you are not␣
↪using a Trinket
#if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
// End of trinket special code
strip.setBrightness(BRIGHTNESS);
strip.begin();
strip.show(); // Initialize all pixels to 'off'
}

void loop() {
// Some example procedures showing how to display to the pixels:
colorWipe(strip.Color(255, 0, 0), 50); // Red
colorWipe(strip.Color(0, 255, 0), 50); // Green
colorWipe(strip.Color(0, 0, 255), 50); // Blue
colorWipe(strip.Color(0, 0, 0, 255), 50); // White

whiteOverRainbow(20,75,5);
```

(continues on next page)

(continued from previous page)

```
pulseWhite(5);

// fullWhite();
// delay(2000);

rainbowFade2White(3,3,1);


}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
}
}

void pulseWhite(uint8_t wait) {
for(int j = 0; j < 256 ; j++){
    for(uint16_t i=0; i<strip.numPixels(); i++) {
        strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        delay(wait);
        strip.show();
    }

for(int j = 255; j >= 0 ; j--){
    for(uint16_t i=0; i<strip.numPixels(); i++) {
        strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        delay(wait);
        strip.show();
    }
}


void rainbowFade2White(uint8_t wait, int rainbowLoops, int whiteLoops) {
float fadeMax = 100.0;
int fadeVal = 0;
uint32_t wheelVal;
int redVal, greenVal, blueVal;

for(int k = 0 ; k < rainbowLoops ; k ++){

    for(int j=0; j<256; j++) { // 5 cycles of all colors on wheel

    for(int i=0; i< strip.numPixels(); i++) {

        wheelVal = Wheel(((i * 256 / strip.numPixels()) + j) & 255);

        redVal = red(wheelVal) * float(fadeVal/fadeMax);
        greenVal = green(wheelVal) * float(fadeVal/fadeMax);
        blueVal = blue(wheelVal) * float(fadeVal/fadeMax);

        strip.setPixelColor( i, strip.Color( redVal, greenVal, blueVal ) );
```

(continues on next page)

(continued from previous page)

```
    }

    //First loop, fade in!
    if(k == 0 && fadeVal < fadeMax-1) {
        fadeVal++;
    }

    //Last loop, fade out!
    else if(k == rainbowLoops - 1 && j > 255 - fadeMax ){
        fadeVal--;
    }

        strip.show();
        delay(wait);
    }

}



delay(500);


for(int k = 0 ; k < whiteLoops ; k ++){

    for(int j = 0; j < 256 ; j++){

        for(uint16_t i=0; i < strip.numPixels(); i++) {
            strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        strip.show();
        }

        delay(2000);
    for(int j = 255; j >= 0 ; j--){

        for(uint16_t i=0; i < strip.numPixels(); i++) {
            strip.setPixelColor(i, strip.Color(0,0,0, neopix_gamma[j] ) );
        }
        strip.show();
        }
}

delay(500);


}

void whiteOverRainbow(uint8_t wait, uint8_t whiteSpeed, uint8_t whiteLength ) {

if(whiteLength >= strip.numPixels()) whiteLength = strip.numPixels() - 1;

int head = whiteLength - 1;
int tail = 0;

int loops = 3;
```

(continues on next page)

(continued from previous page)

```
int loopNum = 0;

static unsigned long lastTime = 0;


while(true){
    for(int j=0; j<256; j++) {
    for(uint16_t i=0; i<strip.numPixels(); i++) {
        if((i >= tail && i <= head) || (tail > head && i >= tail) || (tail > head &&
↪i <= head) ){
        strip.setPixelColor(i, strip.Color(0,0,0, 255 ) );
        }
        else{
        strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
        }

    }

    if(millis() - lastTime > whiteSpeed) {
        head++;
        tail++;
        if(head == strip.numPixels()){
        loopNum++;
        }
        lastTime = millis();
    }

    if(loopNum == loops) return;

    head%=strip.numPixels();
    tail%=strip.numPixels();
        strip.show();
        delay(wait);
    }
}

}
void fullWhite() {

    for(uint16_t i=0; i<strip.numPixels(); i++) {
        strip.setPixelColor(i, strip.Color(0,0,0, 255 ) );
    }
    strip.show();
}


// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
uint16_t i, j;

for(j=0; j<256 * 5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
    strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
}
```

(continues on next page)

(continued from previous page)

```
}

void rainbow(uint8_t wait) {
uint16_t i, j;

for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
}
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
WheelPos = 255 - WheelPos;
if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3,0);
}
if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3,0);
}
WheelPos -= 170;
return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0,0);
}

uint8_t red(uint32_t c) {
return (c >> 16);
}
uint8_t green(uint32_t c) {
return (c >> 8);
}
uint8_t blue(uint32_t c) {
return (c);
}
```

3.3 Touch With NeoPixel

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library
#include "TinyTouch.h"
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif

#define TOUCH_TRIGGER_VALUE 800

TinyTouch touch;

uint8_t touchPins[2] = {13, 2}; //initialize touch pins
```

(continues on next page)

(continued from previous page)

```
// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN            12

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      30

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to␣
↪use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--
↪see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int delayval = 20; // delay for 20 ms

int rainBowLoop = 0;

void setup() {

    touch.begin(touchPins, sizeof(touchPins));
    // This is for Trinket 5V 16MHz, you can remove these three lines if you are not␣
↪using a Trinket
    #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
    #endif
    // End of trinket special code

    pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {

    // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to␣
↪the count of pixels minus one.
    touch.touchHandle();

    if (touch.getValue(0) > TOUCH_TRIGGER_VALUE && touch.getValue(1) > TOUCH_TRIGGER_
↪VALUE)
    {
        for(int i=0;i<NUMPIXELS;i++){
            pixels.setPixelColor(i, pixels.Color(0,150,0));
            pixels.show();
        }
    }
    else if (touch.getValue(0) > TOUCH_TRIGGER_VALUE && touch.getValue(1) < TOUCH_
↪TRIGGER_VALUE) {
        for(int i=0;i<NUMPIXELS;i++){
            pixels.setPixelColor(i, pixels.Color(255,0,0));
            pixels.show();
        }
    }
    else if (touch.getValue(0) < TOUCH_TRIGGER_VALUE && touch.getValue(1) > TOUCH_
↪TRIGGER_VALUE) {
        for(int i=0;i<NUMPIXELS;i++){
            pixels.setPixelColor(i, pixels.Color(255,255,255));
```

(continues on next page)

(continued from previous page)

```
            pixels.show();
        }
    }
    else {
        rainBowLoop++;

        if (rainBowLoop >= 255) {
        rainBowLoop = 0;
        }

        for(int i=0; i<pixels.numPixels(); i++) {
        pixels.setPixelColor(i, Wheel((i+rainBowLoop) & 255));
        }
        pixels.show();
        delay(delayval);
    }
}

    // Input a value 0 to 255 to get a color value.
    // The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
    WheelPos = 255 - WheelPos;
    if(WheelPos < 85) {
        return pixels.Color(255 - WheelPos * 3, 0, WheelPos * 3);
    }
    if(WheelPos < 170) {
        WheelPos -= 85;
        return pixels.Color(0, WheelPos * 3, 255 - WheelPos * 3);
    }
    WheelPos -= 170;
    return pixels.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

3.4 Servo Control

```
#include <Servo_megaTinyCore.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
    myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
        // in steps of 1 degree
        myservo.write(pos);              // tell servo to go to position in variable
↪'pos'
        delay(15);                       // waits 15ms for the servo to reach the␣
↪position
    }
```

(continues on next page)

(continued from previous page)

```
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
        myservo.write(pos);              // tell servo to go to position in variable
↪'pos'
        delay(15);                       // waits 15ms for the servo to reach the␣
↪position
    }
}
```

3.5 EEPROM

```
#include <EEPROM.h>

void setup() {

float f = 0.00f;   //Variable to store data read from EEPROM.
int eeAddress = 0; //EEPROM address to start reading from

Serial.begin(9600);
while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
}
Serial.print("Read float from EEPROM: ");

//Get the float data from the EEPROM at position 'eeAddress'
EEPROM.get(eeAddress, f);
Serial.println(f, 3);    //This may print 'ovf, nan' if the data inside the EEPROM is␣
↪not a valid float.

/***
    As get also returns a reference to 'f', you can use it inline.
    E.g: Serial.print( EEPROM.get( eeAddress, f ) );
***/

/***
    Get can be used with custom structures too.
    I have separated this into an extra function.
***/

secondTest(); //Run the next test.
}

struct MyObject {
float field1;
byte field2;
char name[10];
};

void secondTest() {
int eeAddress = sizeof(float); //Move address to the next byte after float 'f'.

MyObject customVar; //Variable to store custom object read from EEPROM.
EEPROM.get(eeAddress, customVar);

Serial.println("Read custom object from EEPROM: ");
```

(continues on next page)

(continued from previous page)

```
Serial.println(customVar.field1);
Serial.println(customVar.field2);
Serial.println(customVar.name);
}

void loop() {
/* Empty loop */
}
```

3.6 UART Communication with ESP8266

```
#include "Adafruit_EPD.h"

char rec;
char rec_buf[10];
volatile int index = 0;
volatile int flag = 0;

float temp_data = 0;

#define EPD_CS     13
#define EPD_DC      3
#define SRAM_CS     2
#define EPD_RESET   1 // can set to -1 and share with microcontroller Reset!
#define EPD_BUSY    0 // can set to -1 to not use a pin (will wait a fixed delay)

/* Uncomment the following line if you are using 1.54" tricolor EPD */
Adafruit_IL0373 display(152, 152 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);

/* Uncomment the following line if you are using 2.15" tricolor EPD */
//Adafruit_IL0373 display(212, 104 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);

/* Uncomment the following line if you are using 2.7" tricolor EPD */
//Adafruit_IL91874 display(264, 176 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS);
char *buf = "123.45";

void get_weather_data() {
Serial.println("AT+CIPSTART=\"TCP\",\"api.openweathermap.org\",80");
delay(1000);
Serial.println("AT+CIPSEND=113");
delay(500);

while(Serial.available()){
    Serial.read();
}

index = 0;
flag = 0;

memset(rec_buf, 0, sizeof(rec_buf));

Serial.print("GET /data/2.5/weather?id=4885955&appid=a9077ed0f95a1800cd2e7752adfdc137␣
↪HTTP/1.1");
Serial.write(0x0d);
```

(continues on next page)

(continued from previous page)

```
Serial.write(0x0a);
Serial.println("Host: api.openweathermap.org");
Serial.write(0x0a);

for(int i = 0; i < 1000; i++){
    delay(1);
    while(Serial.available()){
    rec = Serial.read();

    if (rec == 't' && flag == 0) {
        flag++;
    } else if (rec == 'e' && flag == 1) {
        flag++;
    } else if (rec == 'm' && flag == 2) {
        flag++;
    } else if (rec == 'p' && flag == 3) {
        flag++;
    } else if (rec == 0x22 && flag == 4) {
        flag++;
    } else if (rec == 0x3a && flag == 5) {
        flag++;
    } else if (flag == 6) {
        if (index < 6) {
        rec_buf[index++] = rec;
        } else {
        flag = 0;
        }
    } else {
        flag = 0;
    }
    }

    if (index == 6) {
    break;
    }
}



delay(1000);

Serial.println("AT+CIPCLOSE");

}

void display_temp(float temp) {
    display.clearBuffer();
    display.setCursor(2, 0);
    display.fillScreen(EPD_WHITE);
    display.setTextColor(EPD_BLACK);
    display.setTextSize(2);
    display.println(" ");
    display.println("Current");
    display.println("Temperature:");
    display.println(" ");
    display.setTextSize(4);
    display.setTextColor(EPD_RED);
```

(continues on next page)

(continued from previous page)

```
    display.print(temp, 1);
    display.println(" F");
    //refresh the display
    display.display();
}

void setup() {
    Serial.begin(115200);

    display.begin();

    delay(10000);
}

void loop() {

    get_weather_data();

    temp_data = atof(rec_buf);

    temp_data = (temp_data - 273.15) * 1.8 + 32;

    Serial.println(temp_data,2);

    if (temp_data > -100)
    {
        display_temp(temp_data);
    }


    delay(600000);
    //don't do anything!
}
```

3.7 Interfacing with LCD Display

```
#include <Wire.h>
#include "rgb_lcd.h"
#include "Adafruit_HTU21DF.h"

rgb_lcd lcd;

const int colorR = 255;
const int colorG = 255;
const int colorB = 255;

Adafruit_HTU21DF htu = Adafruit_HTU21DF();

void setup()
{

    Serial.begin(9600);
    Serial.println("HTU21D-F test");
```

(continues on next page)

(continued from previous page)

```
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);

    lcd.setRGB(colorR, colorG, colorB);

    // Print a message to the LCD.
    //lcd.print("hello, world!");


    if (!htu.begin()) {
    Serial.println("Couldn't find sensor!");
    while (1);
    }

    delay(1000);
}

void loop()
{
    // set the cursor to column 0, line 1
    // (note: line 1 is the second row, since counting begins with 0):
    float temp = htu.readTemperature();
    float rel_hum = htu.readHumidity();
    lcd.setCursor(0, 0);
    lcd.print("Temp=");
    lcd.print(temp, 2);
    lcd.print(" C");

    lcd.setCursor(0, 1);
    // print the number of seconds since reset:
    lcd.print("Hum=");
    lcd.print(rel_hum, 2);
    lcd.print(" \%");

    Serial.print("Temp: "); Serial.print(temp); Serial.print(" C");
    Serial.print("   ");
    Serial.print("Humidity: "); Serial.print(rel_hum); Serial.println(" \%");
    delay(1000);
}
```

3.8 Interfacing with E-ink Display

```
#include "Adafruit_EPD.h"


#define EPD_CS     13
#define EPD_DC      3
#define SRAM_CS     2
#define EPD_RESET   -1 // can set to -1 and share with microcontroller Reset!
#define EPD_BUSY    -1 // can set to -1 to not use a pin (will wait a fixed delay)

/* Uncomment the following line if you are using 1.54" tricolor EPD */
Adafruit_IL0373 display(152, 152 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);
```

(continues on next page)

(continued from previous page)

```
/* Uncomment the following line if you are using 2.15" tricolor EPD */
//Adafruit_IL0373 display(212, 104 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);

/* Uncomment the following line if you are using 2.7" tricolor EPD */
//Adafruit_IL91874 display(264, 176 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS);

void setup() {

    Serial.begin(9600);

    display.begin();
    display.clearBuffer();

    //draw some pretty lines
    for (int16_t i=0; i<display.width(); i+=4) {
        display.drawLine(0, 0, i, display.height()-1, EPD_BLACK);
    }

    for (int16_t i=0; i<display.height(); i+=4) {
        display.drawLine(display.width()-1, 0, 0, i, EPD_RED);
    }

    //refresh the display
    display.display();
}

void loop() {
//don't do anything!
}
```

CHAPTER 4

Advanced Usage

4.1 Using Ateml Studio

CHAPTER 5

Support

5.1 Source Code

You can find the source code here.

For issues, please submit issues.

You are most welcomed to submit pull requests for contributions.

5.2 Feedback

If you have aditional feedback, please send email to xukangmin@gmail.com

CHAPTER 6

Indices and tables

- genindex
- modindex
- search