Finch Robots Documentation

Release 1.0

Ryan Haasken

July 29, 2016

Contents

This documentation supports the CoderDojo Twin Cities' Robots code group. We primarily use Finch robots, but we also have a couple of mBots available. We focus on using the block programming language, Scratch, to program both types of robots, but it is possible to program them using other languages, such as Python for the Finch robots.

See the Finch robot webpage for general information about the Finch robot.

See the Makeblock webpage for general information about the mBot.

CHAPTER 1

Finch Robots

This section describes the Finch robots and how to program them.

1.1 Introduction to the Finch robot

This page describes the Finch robot hardware, at a high level. At the end of this lesson, you should be able to answer questions about the capabilities of the Finch robot.

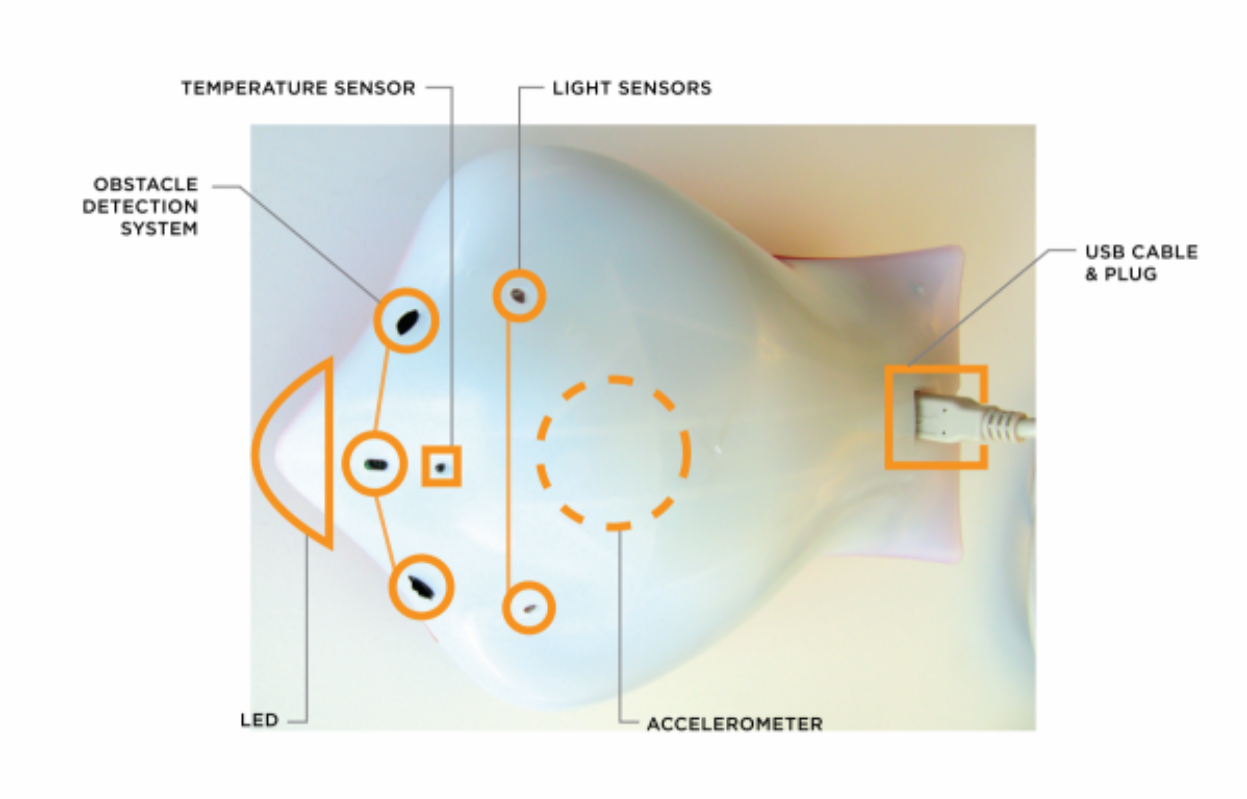


Fig. 1.1: Finch robot hardware diagram, with sensors labeled. Taken from the official hardware description

1.1.1 The Finch's sensors

The Finch has the following sensors:

Light The Finch has two sensors which can measure light levels. There is one on the right and one on the left.

Temperature The Finch has one temperature sensor. This can tell you the temperature of the room.

Obstacle The finch has two obstacle sensors in the front. There is one on the left, and one on the right. They do not tell you the distance to an obstacle, but they do tell you if there is an obstacle in front of them. They work best against light-colored or reflective objects.

These work by sending out infrared light (light we can't see with our eyes) and looking for it to bounce back and hit the sensors.

Accelerometers The Finch has an accelerometer that can tell you the orientation of the Finch. That means it can tell you whether the Finch is upside down, or tilted in any direction. This sensor can also tell you if the Finch is shaken or tapped.

1.1.2 The Finch's actuators

The Finch has the following actuators (things that do actions):

Motors The Finch has two motors, one for each wheel. The Finch's movement direction can be controlled by giving the motors different speeds.

Light The Finch has a Light-Emitting Diode (LED) in its nose. It can light up in any color by mixing the three colors, red, green, and blue (RGB).

Buzzer The finch has a buzzer which can play frequencies between 100 Hz and 10,000 Hz (10 KHz). 100Hz is a low sound, and 10,000 Hz is a high-pitched sound.

1.1.3 The Finch's connection

The Finch connects to your computer with a USB cable, and it must remain connected in order for you to control it. You might find that it helps your Finch move more freely if you hold the cord above it while it moves.

1.1.4 Check your understanding

Answer the following questions to check that you understand the Finch robot's hardware:

- Can you point to and name each of the sensors on your robot?
- What is an LED?
- Can the Finch's buzzer play a 50 Hz sound?
- How would you use the Finch's motors to make the Finch turn right? Left?
- Would the Finch's obstacle sensors be more likely to see a black object or a white object?
- How could you tell if the Finch ran into something behind it?

1.1.5 Additional resources

This page is geared towards a younger audience. For a more detailed description of the Finch robot's hardware, see the official hardware description

1.2 Programming Finch with Scratch

This section will describe how to use the Scratch programming language. Scratch is a block-based graphical programming language. See the Scratch home page for more information.

1.2.1 Setting up Finch with Scratch

This page will describe how to set up your computer to program the Finch robot with Scratch.

Install Scratch Offline Editor

You will need to install the Scratch 2 Offline editor, which can be downloaded from the official Scratch downloads page

Install BirdBrain Robot Server

You will also need to install the BirdBrain Robot server. Here are the instructions for Windows and Mac:

Windows Download and run the BirdBrain Robot Server installer for Windows.

Mac OSX Download the BirdBrain Robot Server application for Mac, and drag the app icon into your Applications folder. Then, if runnning Mac OSX 10.9 or later, open your Applications folder in Finder, command-click on the BirdBrainRobot Server, click "Get Info", and then check the box to disable App Nap.

Linux See the official Finch scratch installation instructions.

Start BirdBrain and Scratch

Plug in your Finch robot to your computer and launch the BirdBrain Robot Server. BirdBrain should show that your Finch robot is connected. Click to launch scratch from the BirdBrain Robot Server application.

When Scratch opens, the Finch blocks should be available under the "More Blocks" section of your Scratch Editor.

If the Finch blocks are not available under 'More Blocks', make sure that you launched the Scratch Offline Editor from the button in the BirdBrain Scratch Robot Server application, and make sure that the Finch robot was connected when you launched Scratch.

If the Finch blocks are still not available, download the BirdBrain scratch blocks and unzip it somewhere (e.g. in your 'Documents' or 'My Documents' folder). Then open the Scratch Offline Editor. Hold down 'Shift' and then click 'File' -> 'Import experimental HTTP extension'.

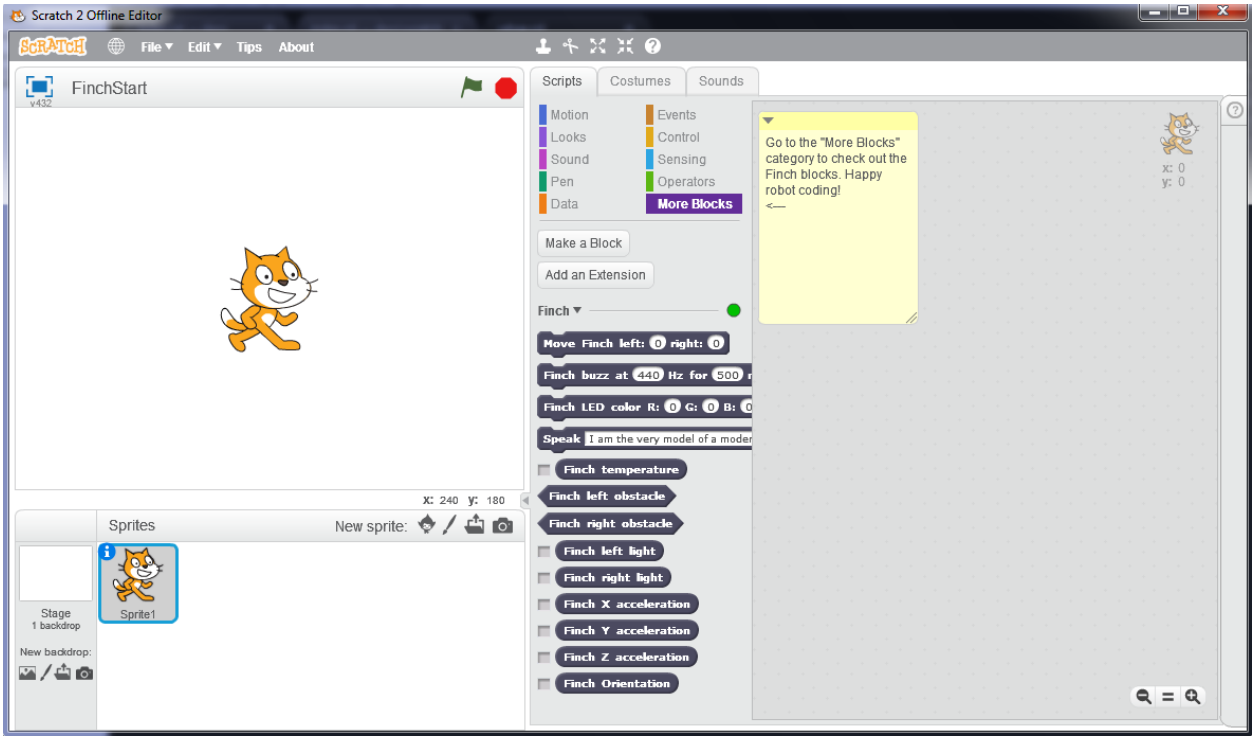
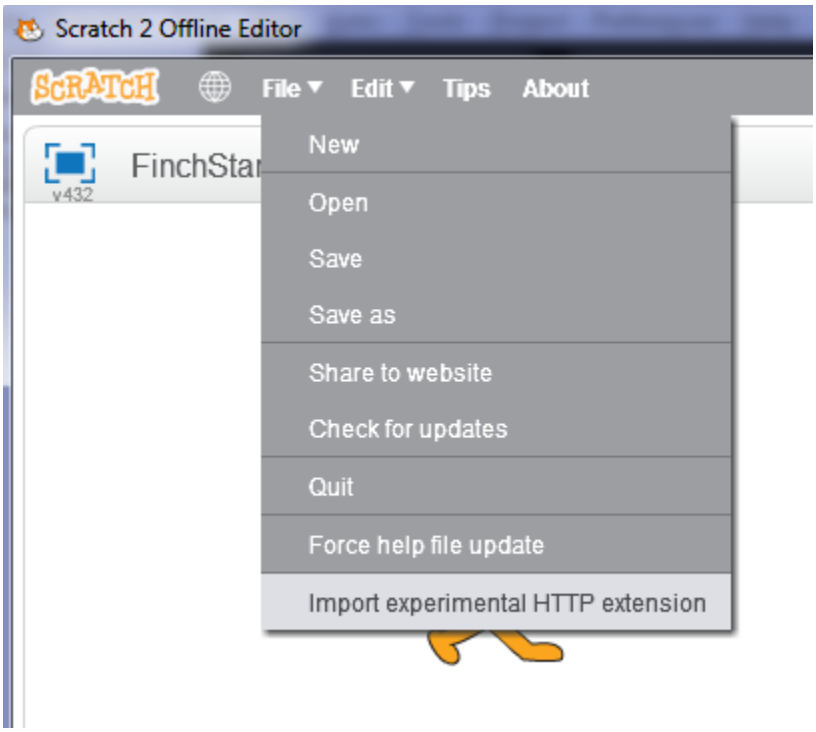


Fig. 1.2: Scratch Offline Editor showing the Finch blocks under 'More Blocks'.

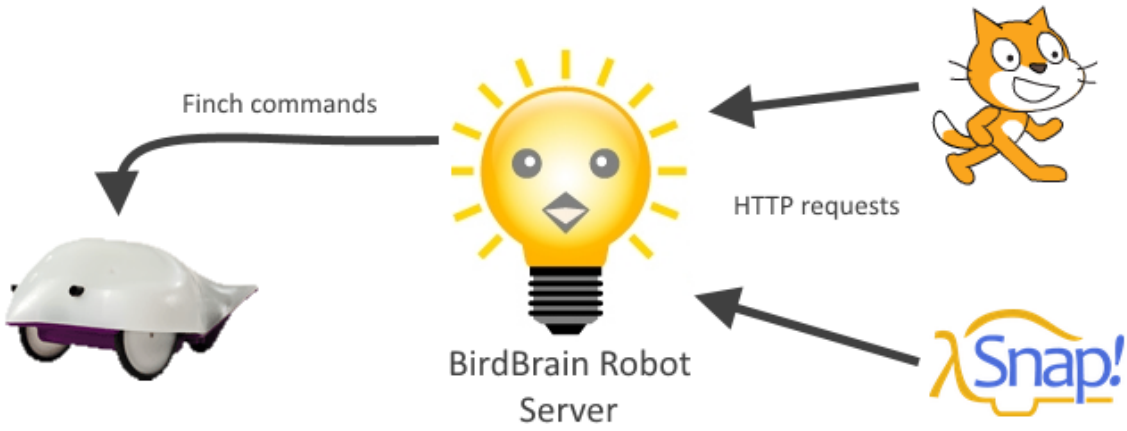


Browse to the location of the Finch.s2e file in the folder that you unzipped, and select it. The Finch blocks should now be available under 'More Blocks'.

You are now ready to start programming your Finch robot in Scratch!

How BirdBrain Robot Server works

The BirdBrain robot server is only required for programming the Finch robot in Scratch or Snap. Its job is to receive requests from Scratch or Snap and send the correct commands to the Finch Robot over USB.



Other Resources

Please see the official Finch scratch installation instructions for more details on getting your Finch working with Scratch if the above instructions are not sufficient. You can also find instructions for installing on Linux there.

1.2.2 Beginner Lessons

These are lessons on programming the Finch robot with scratch geared towards beginners.

Controlling the LED

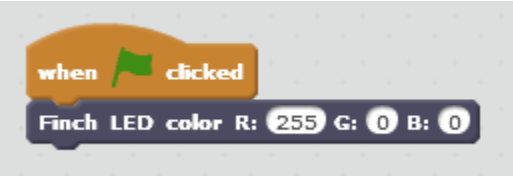
Light up a single color

The Finch has a Light Emitting Diode (LED) in its nose (see *The Finch's actuators* for more information). In this lesson, we will learn how to light up the Finch LED in any color we like!

The first step is to drag an event from the "Events" block group into the script area. This event will tell Scratch when to start our program. For now, we will use the event that is triggered when the green flag icon is clicked.

Now we can connect blocks to the bottom of the "When <Flag> clicked" block, and those blocks will be run whenever we click the green flag.

The block that controls the Finch LED says "Finch LED color R: 0 G: 0 B: 0". Drag this block into your script and connect it to the bottom of the event block.



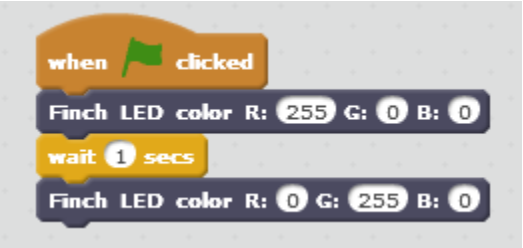
The three values in this block are the red (R), green (G), and blue (B) values of the color. A larger value for the color means more of that color will be lit up. The values can be anywhere from 0 to 255. Set the colors to "R: 255 G: 0 B: 0". The LED should light up bright red when you click the flag icon.

Challenge Try to set the Finch color to the following colors: green, blue, purple, yellow.

Hint: you will need to mix the colors to get the colors purple and yelllow. Ask a mentor if you need help.

Light up two different colors

Now let's use a "Wait" block to make the light change color after 1 second. The "Wait" block can be found under the "Control" blocks. Drag the "Wait" block into your script, and connect it to the bottom of the "Finch LED" block. Then drag another "Finch LED block" into your script and connect it to the bottom of the "Wait" block.



In the above example, when you click the green flag, the program Finch will light up Red for one second, and then the color will change to green, and it will stay that way.

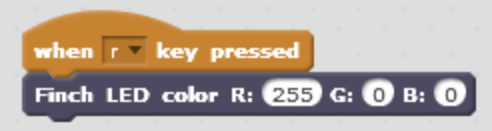
Challenge Can you make the Finch go through a pattern of colors and repeat forever?

Hint: You will need to use a "Forever" loop block from the "Control" blocks. Ask a mentor if you need help.

Control colors with the keyboard

There are other event blocks besides the "When <flag> clicked" block that we used in the previous scripts. Under "Events", find the block which says "When [space] key pressed", and drag that block into your script. If you click on the down arrow next to the word "space", you can select any of the space key, or any of the letter, number, or arrow keys on your keyboard. Set the key to "r". Then drag a "Finch LED" block and attach it to the bottom or the "When [r] key pressed" block.

To allow Scratch to listen to your key presses, click on any blank space in your script window. Now press "r". Your Finch should light up red.



Challenge Can you write a Scratch program that will make your Finch light up red when you press the "r" key, green when you press the "g" key, and blue when you press the "b" key? Can you also make the Finch turn off its LED when you press the "space" key?

Hint: You will need to add three more "When [key] pressed" blocks, select the correct keys for each block, and connect a "Finch LED" block to each one. Ask a mentor if you need help.

Other Resources

You can find RGB values for colors on various web pages. Try this page for example. That page has a color picker that allows you to pick a color and then view its red, green, and blue values.

Controlling the Buzzer

Making a sound with the buzzer

The Finch has a small speaker which we will call a buzzer (see *The Finch's actuators* for more information). In this lesson, we will learn how to make that buzzer play notes of different frequencies.

The first step is to drag an event from the "Events" block group into the script area. This event will tell Scratch when to start our program. For now, we will use the event that is triggered when the green flag icon is clicked.

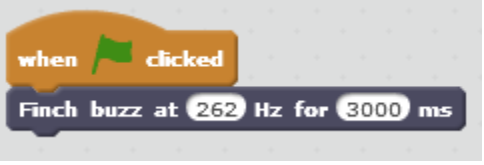
Now we can connect blocks to the bottom of the "When <Flag> clicked" block, and those blocks will be run whenever we click the green flag.

The block that controls the Finch buzzer says "Finch buzz at 440 Hz for 500 ms". Drag this block into your script and connect it to the bottom of the event block.

The first value in the "Finch buzz" block is the frequency (pitch) that the buzzer will play. It is measured in units of Hertz (Hz). A higher value for the frequency will make a higher pitched sound. The Finch can play frequencies from 100 Hz to 10,000 Hz (10 KHz).

The second value in the "Finch buzz" block tells the Finch how long to play the note. It is measured in units of milliseconds (ms). A millisecond is one thousandth of a second, or 0.001 seconds. 1000 ms is equal to 1 second.

In the "Finch buzz" block, set the frequency to 262 Hz, and set the duration to 3000 ms. Then click the green flag, and you should hear your Finch play a note for 3 seconds. This note is called a middle C, which is the C note in the middle of a piano.

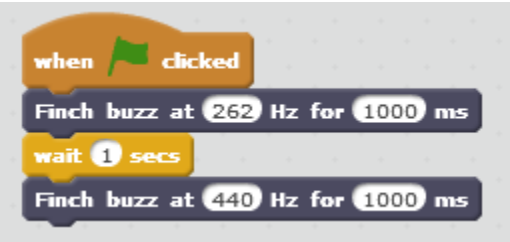


Play around with the values in the "Finch buzz" block and click the green flag to see how the Finch's behavior changes.

Playing a melody

Now lets play a simple two-note melody. You might think you can do this by just dragging another "Finch buzz" block and connecting it to the bottom of the other "Finch buzz" block. However, the "Finch buzz" block does not wait while it plays the note. Instead, it just starts the note and goes on to the next block. This means that if you connect two "Finch buzz" blocks without a "Wait" in between, you will only be able to hear the Finch play the second note. You can try this to see for yourself!

To allow the Finch to play one note after another, we must use a "Wait" block which can be found under the "Control" category. Drag a "Wait" block into your script and attach it to the bottom of the "Finch buzz" block. Set the duration of the "Finch Buzz" block to 1000 ms, and then set the "Wait" block to wait for 1 second (remember, 1000 ms = 1 second). Then drag another "Finch buzz" block and attach it to the bottom of the "Wait" block. Set the frequency of the two blocks to different values. Then click the green flag to run your program. You should hear two separate notes.



Challenge Can you write a Scratch program that will make your Finch play the C major scale? The notes of the C major scale are listed below.

- C (262 Hz)
- D (294 Hz)
- E (330 Hz)
- F (349 Hz)
- G (392 Hz)
- A (440 Hz)
- B (494 Hz)
- C (523 Hz)

Challenge Can you create two scratch variables called "duration_sec" and "duration_ms" and use those instead of numbers as the durations for your "Finch buzz" and "Wait" blocks. Then can you change the speed at which your finch plays the notes by changing a single value?

Hints:

- To create a variable, click "Make a Variable" under the "Data" block category. Then you can drag those variable values into the "Finch buzz" and "Wait" blocks.
- Remember that 1 second is equal to 1000 ms, so to convert from seconds to milliseconds, you can set "duration_ms" to "duration_secs" * 1000. You can find a multiplication block under the "Operators" category because multiplication is an operation.

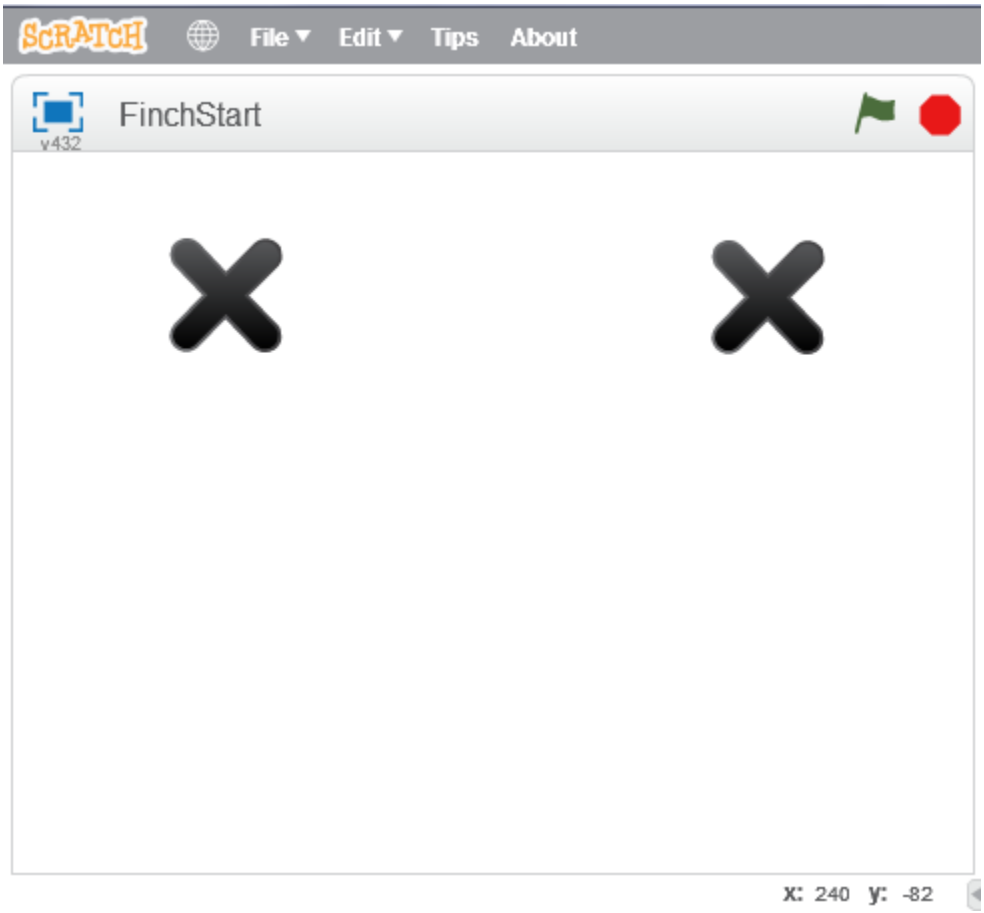
Sensing Obstacles

Reading obstacle sensor values

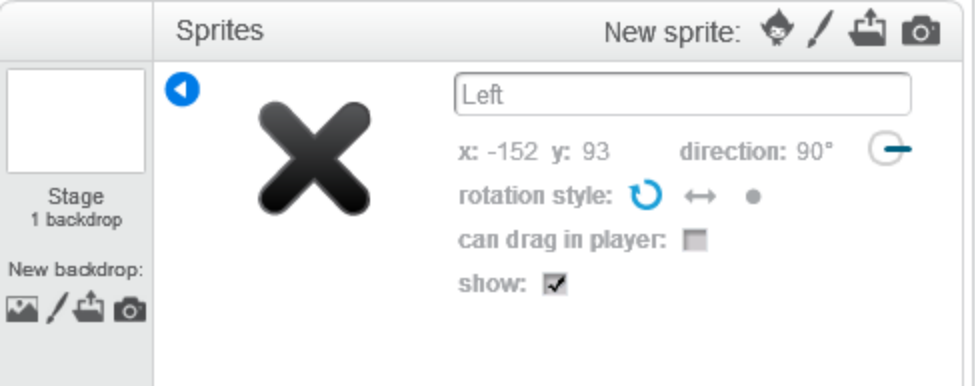
The Finch has a left and a right obstacle sensor (see *The Finch's sensors* for more information). In this lesson, we will learn how to read the values of each obstacle sensor and to use those value to affect the Finch's behavior.

There are two blocks that read the Finch obstacle sensor values, one for the left sensor, and one for the right sensor. They are called "Finch left obstacle" and "Finch right obstacle". These blocks have Boolean values, meaning they are either *True* or *False*. This means they can be used in conditional statement blocks, such as the "if < > then" block.

We will use sprites in Scratch to show the sensor values. Sprite is just a name for a two-dimensional image that can be drawn on the screen. Remove the main cat sprite from your window by right-clicking on it and selecting "delete". Then click the sprite icon next to "New sprite" and select the "X" sprite. Add a second "X" sprite. Your screen should look like the screen shown below:



You should rename your sprites by right-clicking on them and selecting "Info". Enter "Left" for the name of the left sprite, and enter "Right" for the name of the right sprite. This is just so we don't get confused.



Now we want to write a Scratch program to make the left sprite appear when there is an obstacle in front of the Finch's left sensor and to make the right sprite appear when there is an obstacle in front of the Finch's right sensor. We will cover how this is done for the left sensor, and then it will be up to you to figure out how to make this work for the right sensor.

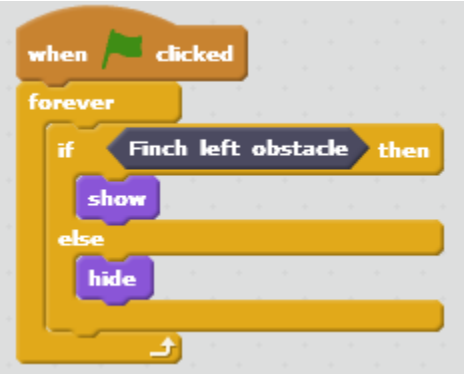
First, click on the right sprite. This will bring up the script for this sprite on the right side of your screen. Start as we usually do by dragging the "When <Flag> clicked" block into the script area.

Next, find the "forever" loop block from the "Control" category. Drag this into your script and connect it to the bottom of the "When <flag> clicked" block. This block will loop forever, running the blocks inside of it repeatedly until you stop the program.

Each time our loop runs, we want to check whether the left obstacle sensor sees an obstacle. If it does, we want to show the "Left" sprite. Otherwise, we want to hide the "Left" sprite. We will use an if-then-else block to do the check on the obstacle sensor value.

Find the "if <> then [] else" block from the "Control" category. Drag it inside the "forever" loop block in your script. Next, find the "Finch left obstacle" block under the "More Blocks" category. Drag this into the space after the "if". The "Finch left obstacle" block has the value "True" when it senses and obstacle and "False" when it doesn't.

Next find the "show" and "hide" blocks from the "Looks" category. Drag the "show" into the space under the "if". Drag the "hide" into the space under the "else". Your script for the "Left" sprite should now look like this.



Click the green flag, and try putting your hand in front of the left obstacle sensor. Note that if you get too close, it won't sense the obstacle. When your hand is in front of the obstacle, the left sprite should appear on the screen.

Challenge Write a script for the right obstacle so that the right sprite will show up only when there is an obstacle in front of the right sensor. Hints:

- Click on the "Right" sprite to open the script for that sprite.
- Use the "Finch right obstacle" block.

Ask a mentor if you need help.

Challenge Can you write a Scratch program so that the LED lights up different colors depending on whether the Finch sees an obstacle on the right or left?

Controlling the Wheels

Making the Finch move forward

The Finch has two wheels which can be controlled independently (see *The Finch's actuators* for more information). In this lesson we will learn how to control the Finch's wheels to make it move in the directions we want.

Instead of using the "When <flag> clicked" event block to start our program, we will use the "When [space] key pressed" block from the "Events" category. Drag that block into your script.

The block to control the Finch's wheels is called "Move Finch left: 0 right: 0". The *left* value controls the speed of the left wheel, and the *right* value controls the speed of the right wheel. Use values of at least 20 for the speeds. Drag the "Move Finch" block into your script and connect it to the bottom of the "When [space] key pressed" block. Click in some empty space in your script, and then press the space bar. Does anything happen?

Nothing should happen yet because the wheel speeds are set to 0. Let's keep the speeds set to zero when we press the space bar, so that we have an easy way to stop the Finch once it's going.

Drag another "When [space] key pressed" block into the script, and change the key from "space" to "up". This event will be activated when you press the up arrow key. Drag a "Move Finch" block and connect it to the bottom of the "When [up] key pressed" block. Set *left* and *right* to 80.

Now click anywhere in your script. You should now be able to press the up arrow key to make the Finch move forward, and you should be able to press the space bar to stop it.

Challenge Can you add three more events to your script so that the Finch moves backward when the down arrow key is pressed, left when the left arrow key is pressed, and right when the right arrow key is pressed?

Hints:

- You will need to add more "When [ ] key pressed" blocks to your script.
- To make the Finch turn, try setting *left* and *right* to different values to make one wheel move faster than the other.
- What happens if you use a negative number (for example, -100) for *left* and *right*?

Challenge Can you figure out the maximum value for the wheel speeds? How did you figure it out?

Hint: what makes the Finch turn?

Challenge Can you add the ability to change the Finch's speed with other keys on the keyboard? For example, make the "F" key increase the speed, and make the "D" key decrease the speed.

Hint:

- Use a variable to store the Finch's current speed. Select "Make a Variable" under the "Data" category to create a variable.
- Use "When [ ] key pressed" blocks to add to and subtract from the speed variable. You can find addition and subtraction blocks under the "Operators" category.
- Use another variable to store the Finch's direction.
- Use a "When <flag> clicked" block to start a forever loop that sets the Finch's wheel speeds according to the values of the speed and direction variables.

Avoiding obstacles when moving

Next we will write a program to make the Finch wander around and avoid obstacles. This will require using the obstacle sensors. If you are unfamiliar with this, see *Reading obstacle sensor values*.

First, let's add a "When <flag> clicked" block to our script. Then drag a "forever" loop block from the "Control" category and connect it to the bottom of the "When <flag> clicked" block.

Now let's think about what we want our program to do. Sometimes it can be helpful to write down what you want your program to do using plain English. This can be referred to as pseudocode (fake code). Let's try it. Here's what we want our Finch robot to do written as a set of steps:

1. Move forward
2. If there is an obstacle on my right, ...
    - then back up and turn left a little
    - continue moving forward
3. If there is an obstacle on my left, ...
    - then back up and turn right a little
    - continue moving forward
4. If there is no obstacle, ...
    - then just continue moving forward.

Given this pseudocode, it is fairly simple to write our Scratch program. First, insert a "Move finch" block into the "forever" loop, and set its *right* and *left* values to 80.

Then, insert an "if <> then" block from the "Control" category. For the condition after the "if", we want to check whether there is an obstacle in front of the right sensor. From the "More Blocks" category, drag the "Finch right obstacle" block into the space after the "if".

Now let's also add a second "if <> then" block under the first "if <> then" block. The condition for this one should be whether there is an obstacle in front of the left sensor. From the "More Blocks" category, drag the "Finch left obstacle" block into the space after this second "if".

This is what your program should look like now.

If there is an obstacle in front of the right sensor, we want to tell the Finch to back up and turn left for a short time. Drag a "Move Finch" block into the space under the first "if". Set *left* to -80 and *right* to -40. Then drag a "Wait" block from the "Control" category and connect it to the bottom of the "Move Finch" block. Set it to wait for 1 second.

This is what your program should look like now.

Now, drag the correct blocks into the space under the second "if" to make the Finch back up and turn right for a short time if it detects an obstacle on the left.

Challenge Can you make the obstacle avoidance better by changing some of the speed or wait values? Try setting up walls of boxes on the floor, and see how well the Finch avoids the walls.

1.3 Programming Finch with Python

This section will describe how to use the Python programming language to program the Finch robot. See the Python home page for more information about Python.

1.3.1 Setting up Finch with Python

This page will describe how to set up your computer to program the Finch robot with Python.

Install Python

If you have not already done so, you will need to install the Python interpreter on your computer. If you are not sure whether you have Python installed on your computer, search for "Python" via Spotlight search on the Mac (Cmd + space) or via the search on Windows (Windows key). If the Python command line program does not show up, you probably do not have Python installed.

Go to the official Python downloads page to download Python for your platform. You can download either a 2.7 or a 3.x version, since the Finch Python code indicates it has been tested on both. Whichever version you download, make sure you look at the correct version of the documentation when you are looking for help programming in Python.

Download Finch Python Files

You will need to download the Finch Python software package. The same package supports Windows, Mac OSX, and Linux because it contains libraries compatible with each of those platforms. The package is just a zip file containing python files and dynamic libraries which communicate with the Finch Robot over USB. The BirdBrain Robot Server is **not** required for programming the Finch with Python.

For Windows, Mac, or Linux, download the Finch Python package. If that link is broken or out of date, find the most current link from the Finch robot downloads page.

Unzip the Finch Python Package

Once you've downloaded the Finch Python package, open the containing directory in Finder/Explorer and extract the zip file. You may want to move the directory containing the Finch Python package into your *Documents* or *My Documents* directory.

Running Python Programs

Option 1 (Easy): Use IDLE (Python Integrated DeveLopment Environment) Open the directory containing the Finch Python files in Explorer/Finder, and then right-click the file to "Edit with IDLE" or "Open with IDLE".

Alternatively, open the IDLE program on your computer, and then select, "File" -> "Open", and navigate to the Python file you would like to open. You can edit files in the provided text editor and run them via "Run" -> "Run module".

Option 2 (Harder): Use Terminal Application Once you've unzipped the Finch Python package, open the "Terminal" program in Mac OSX or the "Command Prompt" in Windows. Navigate to the directory containing your Finch Python files. Use "cd" (Mac OSX) or "CD" (Windows) to change directories in these programs. For example, in Windows,

```
CD Documents\CoderDojo\FinchPython120\
```

Or for Mac OSX

```
cd Documents/CoderDojo/FinchPython120/
```

Then, once you are in that directory in your terminal application, type "python" followed by the name of the python file you would like to execute to execute a python program. For example:

```
python musicexample.py
```

Getting Help with Python

There are plenty of great resources out there on the web for learning to program in Python. See the official Python documentation page for a starting point.

In addition, since Python is an interpreted language, you can run the interpreter and type commands interactively to see what they do. Within the interpreter you can also use the *help* function on Python modules, classes, functions, and objects to learn more about what they do.

Finally, if you have a specific question about how to do something in Python, chances are somebody has already asked that question on StackOverflow or some similar site. Just do a google search for your question and see what you can find.

Other Resources

See the official Finch Python page for more information on using Python to program the Finch.

Also see this page on running your Python Finch programs, which covers Windows, Mac OSX, and Linux.

CHAPTER 2

mBot Robots

This section describes the mBot robots and how to program them.

2.1 Introduction to the mBot robot

This page gives an introduction to the mBot and describes how to set it up in the CoderDojoTC environment.

There are two Macbooks labeled for the mBot robot. These already have the program mBlock (a version scratch for mBot) installed. If you need to install mBlock on another computer, you can download it from the mBlock downloads page.

2.1.1 Getting started with the mBot

Follow these steps to get started with the mBot:

1. Turn on the mBot using the switch on the side. There will be a blue light that blinks on the wireless card inside the mBot.
2. Plug in the wireless USB dongle to the computer. The previously mentioned blue light will stop blinking and become solid blue. This indicates that the connection has been made between the mBot and the dongle.
3. Open the mBlock program on the computer.
4. Connect to the mBot from within mBlock to send code. In the toolbar at the top of the screen, click *connect* and then click on the *2.4G* option from the drop-down menu.