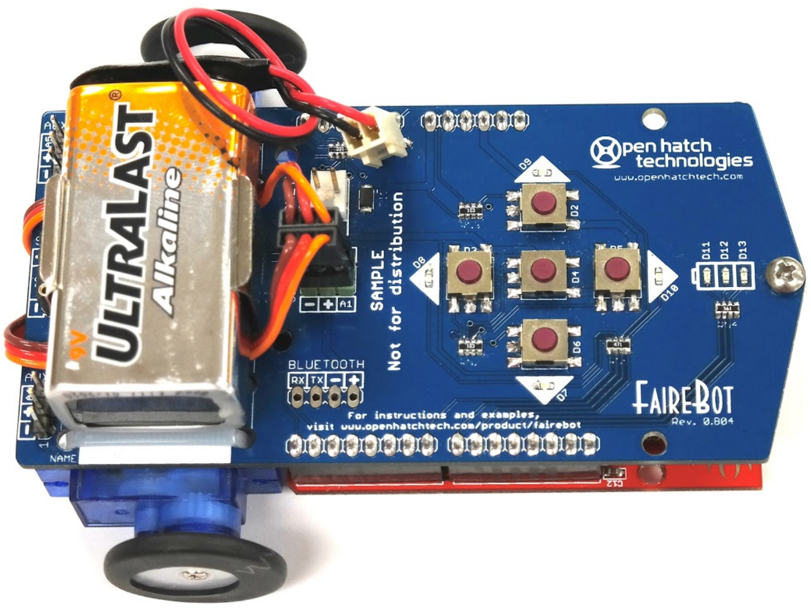


# FaireBot

Assembly and Use

Rev. 110214

http://www.openhatchtech.com/product/fairebot

**FaireBot** Assembly and Use
Copyright (c) 2014 Open Hatch Technologies, LLC

Revision history:
10/31/14: Initial release
11/2/14: Minor text and image updates reflecting post-manufacturing changes

## Table of Contents

## Introduction

FaireBot is a unique Arduino shield designed to replace costly educational robotics platforms by integrating all of the components necessary for robotic motion onto one circuit board. Now, for a fraction of the price of most Arduino based robots, you can learn the fundamentals of robotics both inside and outside the classroom. As your skill grow and you become more familiar with the FaireBot's hardware you can begin to expand your FaireBot's capability by adding sensors and wireless connectivity. The possibilities are endless!

### In the classroom

FaireBot can be used to teach concepts in robotics ranging from simple mechanics and timing to complex terrain navigation and remote data acquisition. All with one platform. Just think, in one class you could use FaireBot to help you learn the basics of programming Arduino and motion control. Then you take your FaireBot to the next level class and use it to learn more advanced programming concepts, wireless control and sensing! This vertically articulated model can be integrated into existing curriculum and with a small amount of collaboration, multiple teachers can use one standardized platform to teach a variety of skills and concepts. The result is higher student knowledge retention and a better understanding of the technology. Some project ideas include:

#### The Maze Challenge

Take a sheet of plywood, or equivalent flat surface, and create a maze using 2" x 2" wood strips around the board and its perimeter. By having a set maze configuration, students can write concrete programs for their FaireBot to successfully follow and navigate. As the student ability increases, they can attach touch or acoustic sensors to their FaireBot and have it navigate the maze autonomously.

#### The FaireBot Sketch-Bot

FaireBot makes an excellent drawing robot. If you attach a pen or other writing tool to the ¾" standoff found at the back of the robot you can have it draw as it moves! This has immense application for art and design classrooms. Students can explore the effect of code on the path of their path of their robots and can have them write out something as simple as a square to even their name!

#### FaireBot Communication

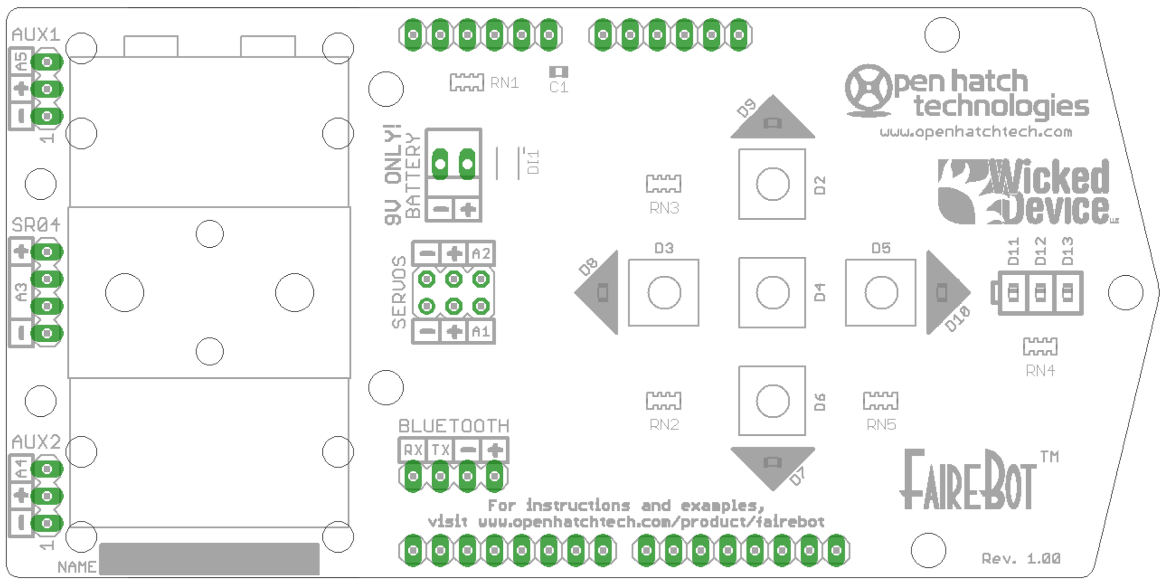
Your FaireBot shouldn't be lonely, it should have friends. A great use for FaireBot in the classroom is to explore electronic communication. This can be obtained by attaching an infrared LED transmitter to the AUX 1 pins and an infrared receiver to the AUX 2 pins. The students can then write programs for their FaireBot's to drive around and communicate with other robots, or even exchange information.

## Technical Overview

The FaireBot shield utilizes every Arduino I/O pin in an attempt to simplify code while maximizing function. The table below highlights the pin-mapping for the shield:

| Pin | Function | Pin | Function |
|---|---|---|---|
| 0 | Bluetooth Receive | 10 | Backward LED |
| 1 | Bluetooth Transmit | 11 | Full Battery LED |
| 2 | Right Button | 12 | Mid Battery LED |
| 3 | Forward Button | 13 | Low Battery LED |
| 4 | Execute Button | A0 | Battery Level |
| 5 | Backward Button | A1 | Left Servo |
| 6 | Left Button | A2 | Right Servo |
| 7 | Left LED | A3 | SR04 Sensor |
| 8 | Forward LED | A4 | AUX 2 Port |
| 9 | Right LED | A5 | AUX 1 Port |

### Silkscreen Graphic



## Assembly

Open your FaireBot package and check that you have all of the following components. Keep in mind that some of the fasteners are small, so make sure you are working on a clean and flat surface. You should see two bags, in addition to your FaireBot, that contain all of the components required to get your robot moving.

Large bag:

| Component | Quantity |
|---|---|
| 9V battery holder | 1 |
| Battery clip | 1 |
| Continuous rotation servos | 2 |
| Cable ties | 5 |
| Party balloons (colors may vary) | 2 |

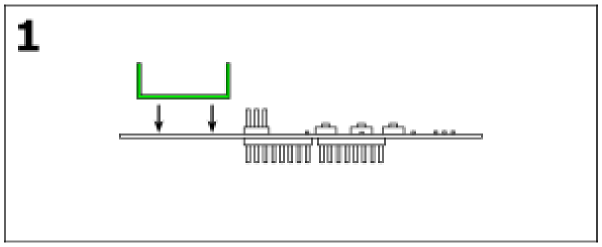
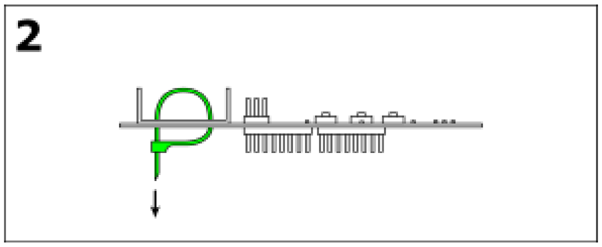
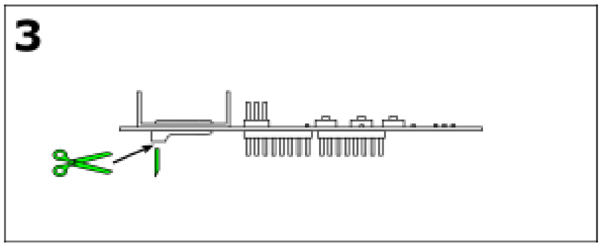
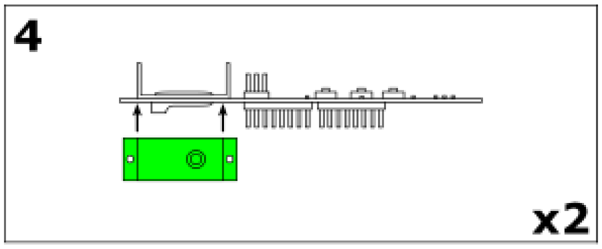
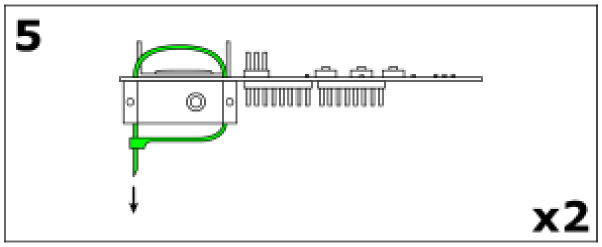
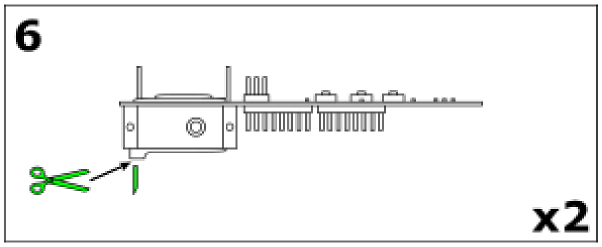
Small bag:

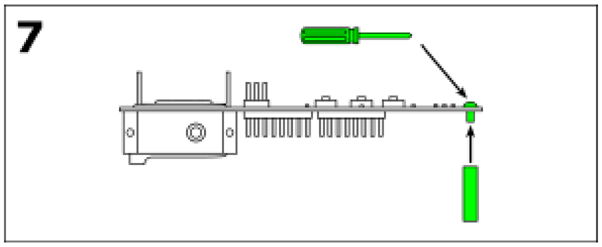
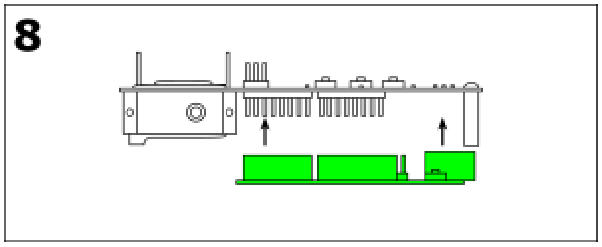
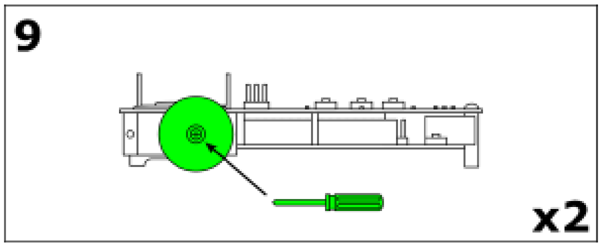
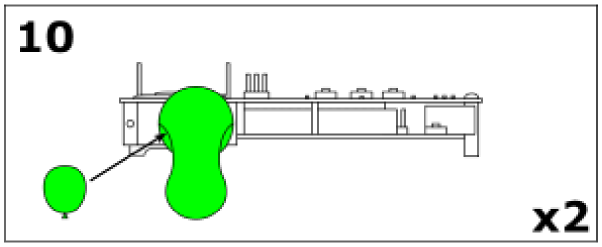
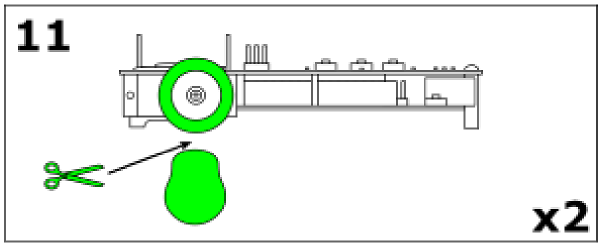
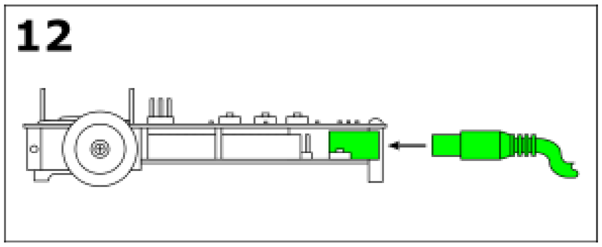
| Component | Quantity |
|---|---|
| Circular servo horns | 2 |
| Servo horn screws | 2 |
| ¾" standoff | 1 |
| 4-40 screw | 1 |

Required components and tools:

| Component/Tool |
|---|
| Arduino UNO, Diecimila or Leonardo |
| High capacity 9V battery |
| Small Phillips screwdriver |
| Snips or scissors |

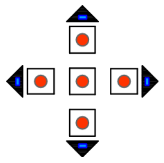
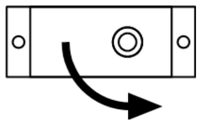
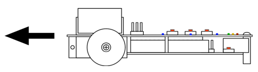
Now that our inventory is complete and you have the necessary tools, let's begin!

| | |
|---|---|
|  | **Step 1** Locate the battery holder and align it in the marked area at the front of your FaireBot shield. (The top of the board is the side with the buttons) |
|  | **Step 2** Using one of your cable ties, attach the battery holder to the top of the FaireBot shield by passing one cable tie through the two big holes. |
|  | **Step 3** Use your snips to trim the excess cable tie "tail" as close to the clasp as possible. Be CAREFUL after you cut off the tails as the remaining tail may be sharp. If necessary, you can smooth the sharp edged with a little sandpaper or simply cover them with tape. |
|  | **Step 4** Align one servo at a time with the marked area on the bottom of the FaireBot shield. The output shaft of the servo should be closer to the back of the shield as shown. |
|  | **Step 5** Make sure that head of the cable tie positioned as shown. Pull the cable tie tight so that is sits flat across all sides of the servo and circuit board. Repeat this process so that there are two cable ties holding each servo. |
|  | **Step 6** Trim off the tail using your snips and repeat steps 4-6 for the remaining servo. |

| | |
|---|---|
|  | **Step 7** Attach the standoff to the back of the FaireBot shield by securing it in place with the 4-40 screw and a Phillips screwdriver. Make sure that the standoff extends downward and elevates your robot off of your work surface. |
|  | **Step 8** Attach the FaireBot shield to the top of your Arduino by carefully aligning the pins and gently squeezing until fully seated. DO NOT attach your shield by pressing down while the FaireBot is on a hard surface, as this may damage your servos. |
|  | **Step 9** Attach the one round servo horns to the each of the servos using the small servo horn screw and a small Phillips screwdriver. |
|  | **Step 10** Stretch the balloons over each servo horn so that it just extends over the back. Make sure that none of the balloon touches the front of the servo. |
|  | **Step 11** Trim off the excess balloon using your scissors and save the leftover. It can be used to cover your servo horns again after the first set wears out. |
|  | **Step 12** Program your Arduino using the latest FaireBot code located on the FaireBot Product Page. Unplug the FaireBot from your computer, attach your battery in the holder, plug in the JST power connector and your FaireBot is ready to go! |

## Getting Started

The following tutorials will help you get started using your FaireBot and will show you the wide range of capability it has to offer. Each lab focuses on a specific element of the FaireBot's use, ranging from programming your Arduino with the latest code to remotely controlling your FaireBot with your phone (or other Bluetooth enabled device). Follow the links below to begin!

| | |
|---|---|
|  | **Lab 1: Hello, FaireBot**<br>It's always best to start with the basics. In this lab we will be investigating how two fundamental components of the FaireBot shield work, the buttons and the LEDs. |
|  | **Lab 2: Servo Tuning**<br>In this lab you will investigate how servos function, how they are tuned and how to connect them to your FaireBot. |
|  | **Lab 3: Basic Navigation**<br>In this lab you will learn how to tune your FaireBot for optimal turning and the basics of navigation. |
| Coming soon! | **Lab 4: Advanced Navigation**<br>In this lab you will learn how to program and configure your FaireBot to navigate autonomously using simple touch and acoustic sensors. |
| Coming soon! | **Lab 5: Wireless Control**<br>In this lab you will learn how to wirelessly control your FaireBot using a Bluetooth module, Android and a Bluetooth terminal. |

## Lab 1: Hello, FaireBot

Before we get started you should have followed the directions on how to assemble your FaireBot and are ready to begin programming. Rather than load the official FaireBot code, lets start off learning how the shield works.

### Requirements

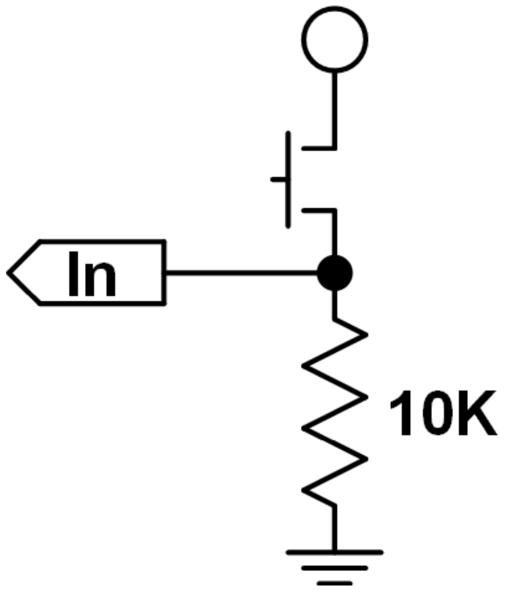
- FaireBot Shield with Arduino
- USB cable for Arduino
- Computer with the latest Arduino IDE

### Overview

FaireBot was designed to be an easy to use, educational robotics shield that allows the user to experience the joy of robotics without requiring complex tools, materials and components. Each FaireBot shield utilizes EVERY available Arduino input and output pin, which makes interfacing with the shield easy. The following steps will walk you through the process of connecting your FaireBot to your computer, running a couple of simple test programs and verifying that everything is running properly.

## Part 1: About the Buttons

Each button is connected to one Arduino input pin. The button works by changing the state of the input pin from a 0 (or low) to a 1 (or high) state when pressed and back to 0 when released. This state change can be sensed by your program using a simple If..Then statement. The following graphic illustrates the button circuitry.



As you see, there are only two components that make up the circuit: the button and a resistor. The button is responsible for changing the state of the pin, while the resistor is used to set the initial state and prevents too much current from flowing through the circuit. If the button is not pressed, the resistor "pulls" the state of the pin low and a 0 is read by your program. Then if the button is pressed, electricity is passed through the button and into both the input pin and the resistor. If the resistor was not present, you would have a short circuit situation and the Arduino would brown out or be permanently damaged. Now that that's sorted out, let's test those buttons!

## Part 2: Button Testing

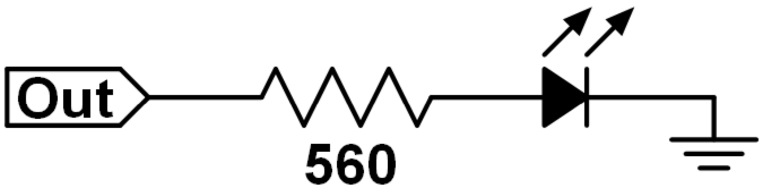
In order to test the buttons, we are going to need to program the FaireBot with a simple program. First, connect your FaireBot to your computer and open the Arduino IDE. Create a new sketch and copy the following code into the editor:

```
//defines variables as pin numbers
int fwd = 3;
int bck = 5;
int lft = 6;
int rht = 2;
int exe = 4;

void setup() {
  //opens and configures the serial port to 9600 baud
  Serial.begin(9600);

  //configure the button pins as inputs
  pinMode(fwd, INPUT);
  pinMode(bck, INPUT);
  pinMode(lft, INPUT);
  pinMode(rht, INPUT);
  pinMode(exe, INPUT);
}

void loop() {
  //checks the state of each input pin and if pressed, shouts out the button name
  if (digitalRead(fwd) == 1) {
    Serial.println("FORWARD!");
  }
  else if (digitalRead(bck) == 1) {
    Serial.println("BACKWARD!");
  }
  else if (digitalRead(lft) == 1) {
    Serial.println("LEFT!");
  }
  else if (digitalRead(rht) == 1) {
    Serial.println("RIGHT!");
  }
  else if (digitalRead(exe) == 1) {
    Serial.println("EXECUTE!");
  }
delay(100);
}
```

Once your FaireBot is programmed, open the Serial Monitor and make sure the baud is set to 9600. Press any of the buttons on the shield and you should see the button's name print in the terminal. Try pressing each button to make sure they all respond. Now that you have tested your buttons, let's go flash some lights!

## Part 3: About the LEDs

LEDs are really cool. Unlike a standard light bulb that uses a filament in a vacuum to produce light, LEDs are solid state and will produce light for a really, really, really long time. The FaireBot shield features 7 individually controllable LEDs that provide visual feedback as to the status of your robot. Three LEDs (the red, yellow and green) are used to indicate the battery level and the remaining four (the blue ones) are used to indicate the robots direction. The following circuitry highlights the LED circuitry:



Each LED is attached to one of the Arduino's output pins through a current limiting resistor. The LED is illuminated when the output changes state from low to high and turns off when changed back to low. The value of the resistor resistor depends on the LED's operating voltage and forward current which can be found in the parts data sheet (usually around 5-10mA). To determine the value of the resistor, you can use the following calculation:

$$\mathbf{R = (V_{source}-V_{forward}) / I_{forward}}$$

Using the formula, we can calculate the proper resistor value for each LED. For example: if the red "Low Battery" LED has a forward voltage of 2.4V and a forward current of 5mA, we would calculate that we would need a 520 ohm resistor to properly illuminate the light. Since resistors only come in standardized values, we would choose the closest match at 560 ohms. Now that that's sorted out, lets make some lights flash!

## Part 4: LED Testing

In order to test out the FaireBot's LEDs, you will need to upload some test code. The following code toggles each LED sequentially. Try changing the time delay and see what happens!

```
int low = 13;
int mid = 12;
int full = 11;
int fwd = 8;
int bck = 10;
int lft = 7;
int rht = 9;

//flash delay time delay(ms)
int flashDelay = 100;

void setup() {
  //Configures pins as outputs
  pinMode(low, OUTPUT);
  pinMode(mid, OUTPUT);
  pinMode(full, OUTPUT);
  pinMode(fwd, OUTPUT);
  pinMode(bck, OUTPUT);
  pinMode(lft, OUTPUT);
  pinMode(rht, OUTPUT);
}

void loop() {
  //flash away!
  digitalWrite(low, HIGH);
  delay(flashDelay);
  digitalWrite(low, LOW);
  delay(flashDelay);

  digitalWrite(mid, HIGH);
  delay(flashDelay);
  digitalWrite(mid, LOW);
  delay(flashDelay);

  digitalWrite(full, HIGH);
  delay(flashDelay);
  digitalWrite(full, LOW);
  delay(flashDelay);

  digitalWrite(bck, HIGH);
  delay(flashDelay);
  digitalWrite(bck, LOW);
  delay(flashDelay);

  digitalWrite(lft, HIGH);
  delay(flashDelay);
  digitalWrite(lft, LOW);
  delay(flashDelay);

  digitalWrite(rht, HIGH);
  delay(flashDelay);
  digitalWrite(rht, LOW);
  delay(flashDelay);

  digitalWrite(fwd, HIGH);
  delay(flashDelay);
  digitalWrite(fwd, LOW);
  delay(flashDelay);
}
```

## FaireBot Lab 2

In Lab 1 we discussed how both the buttons and the LEDs work on the FaireBot Shield. Now its time to get moving! In this lab we will be investigating how the continuous rotation servos work, how we tune them and how they move the FaireBot robot.

### Requirements

- FaireBot Shield with Arduino
- USB cable for Arduino
- Small Phillips or slotted screwdriver
- Computer with the latest Arduino IDE

### Overview

Servos are really cool devices. They combine a DC motor, positional encoder and control circuitry into a small rectangular package. Traditional servos can accurately sweep back and forth within a range of around 180º. The controller sends a varying pulse to the servo and the servo interprets the pulse width as a position. Using the Servo library included with your Arduino, you can accurately position your servo between 0º and 180º with 1º increments. But we have a problem here. What if you want to move your servo past the 180º point? Enter the continuous rotation servo.

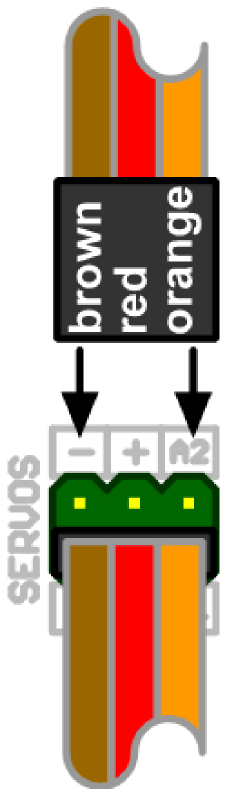
Continuous rotation servos do away with the positional encoder and are capable of continuously spinning either clockwise, counter clockwise or anywhere in between at varying speeds. This fact makes them an excellent solution for moving your robot. So, lets find out how to program them and get them tuned.

## Part 1: Connecting and controlling your servos

Servos feature a three color cable that carries power and a control signal. The colors of the wires depend on the manufacturer, but there is always one thing that is constant: the middle wire is almost ALWAYS positive (with exception of old Airtronics servos). This fact helps prevent you from damaging the servo, or your controller, if you plug it in incorrectly. The common color combinations are:

| Signal | Positive (+) | Ground (-) |
|---|---|---|
| White | Red | Black |
| Yellow | Red | Black |
| Blue | Red | Black |
| **Orange** | **Red** | **Brown** |

To connect your servos to your FaireBot shield, simply look at the servo header (labeled “SERVOS”) and connect the wires as shown. The FeeTech servos used with the FaireBot kit follow the Orange/Red/Brown color coding. The left servo connects to the header closest to the left of the robot and the right servo connects to the header closest to the right.



## Part 2: Tuning

Now that your servos are connected, it's time to tune them. Locate the small hole on the back of each servo. This hole gives you access to a small potentiometer that is used to set the servo's center-point, or the point where the servo is powered and receiving a signal, but not moving. Go ahead and connect your FaireBot to your computer, open the Arduino IDE and copy the following program into a new sketch:

```
#include <Servo.h>

//setup servo names
Servo leftServo;
Servo rightServo;

//setup servo position value (90 = stop)
int pos = 90;

void setup() {
  //attach the servo names to physical pins
  leftServo.attach(A1);
  rightServo.attach(A2);
}

void loop() {
  //sets the position of the servos
  leftServo.write(pos);
  rightServo.write(pos);

  //manditory delay
  delay(15);
}
```

This program writes a value to both servos and if set to 90, will tell them to stop. This value can range from 0 to 180 and will make the servo spin clockwise, counter clockwise, or anywhere in between. Try changing the value of the "pos" variable and see what happens!

To tune the servo's center point, simply upload the previous code, with "pos" set to 90, and use your screwdriver to gently turn the tuning potentiometer either clockwise or counter clockwise until the servo stops spinning. Do this for both servos until they are both stopped. At this point your FaireBot should be completely setup and ready for the official FaireBot code!

## FaireBot Lab 3

Up until this point we have taken a look at the technology that drives the FaireBot and allows us to interface with it. Now it's time to set it free! In this lab we will be discussing how to program your FaireBot with the official code and the basics of turn-by-turn navigation.

### Requirements

- FaireBot Shield with Arduino
- High capacity 9V battery w/ the FaireBot battery clip
- USB cable for Arduino
- Computer with the latest Arduino IDE
- A copy of the FaireBot Program Sheet

### Overview

Although its not very nice to say, the FaireBot is not a very smart little robot. It relies on your brilliance to know where its going and how to interact with the world around it. This is done by inputting your individual directional movements into the FaireBot's memory by pressing one of its 4 directional buttons. Without modification, the FaireBot can store up to 256 directional movements in its memory and will respond with a series of blinking lights if you try to exceed this limit. So press away and see where you FaireBot goes!

## Part 1: Programming the Official FaireBot firmware

The Official FaireBot firmware is quite simple really. It allows the FaireBot to take your input, through the 4 button directional pad, and stores the button presses in memory and then recalls those movements once the Execute button is pressed. The program then makes a copy of those movements in secondary memory, erases its primary memory and waits for a new set of movement commands. If you want your FaireBot to follow the initial movements you entered again, simply press the Execute button and those movements are run from secondary memory.

So, lets get a move on and get that FaireBot Programmed!

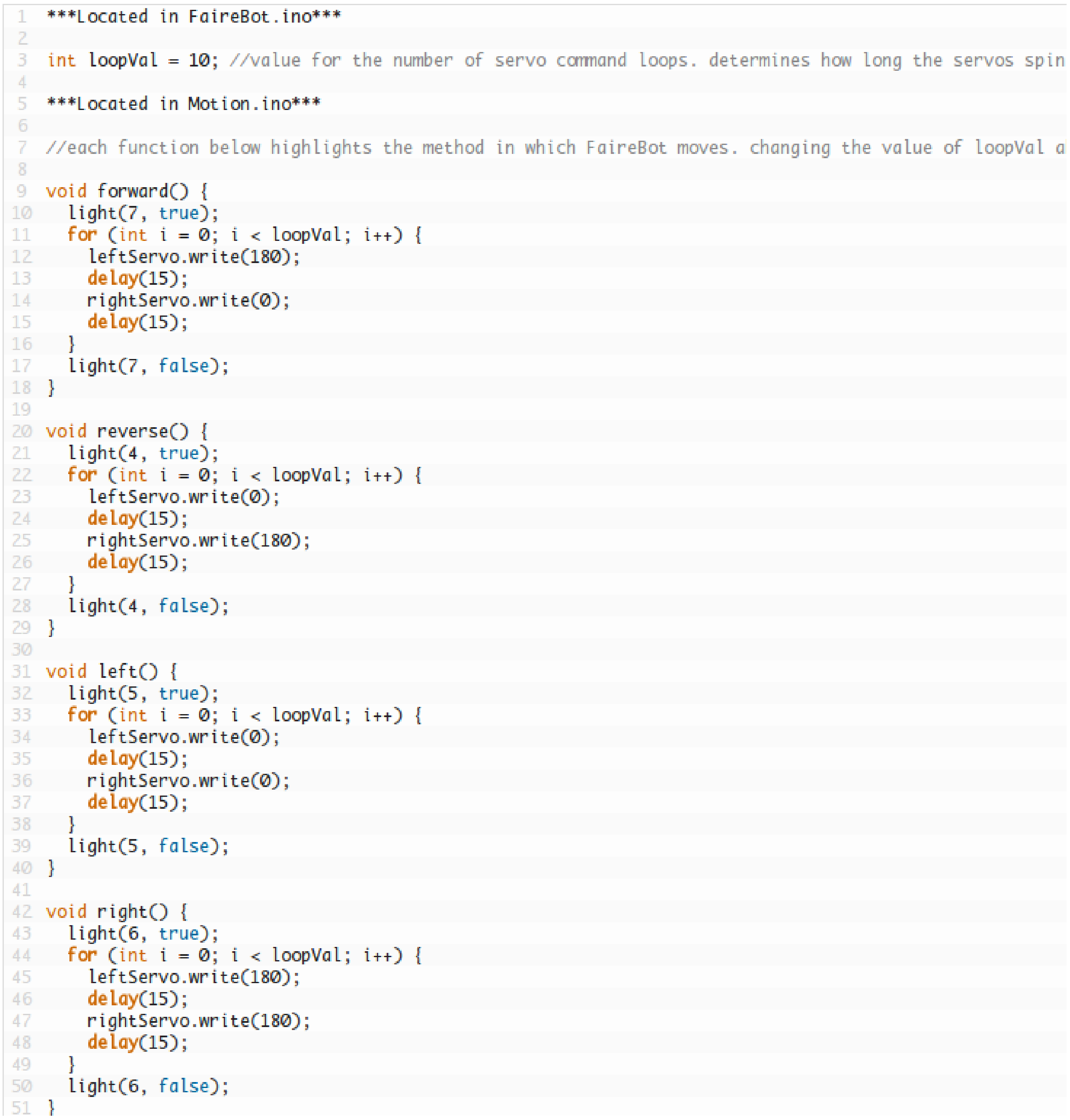
1. Go ahead and download the Official FaireBot Firmware and program your FaireBot using the Arduino IDE.
2. Unplug your FaireBot from your computer, attach your 9V battery and plug in the the JST power connector.

**Do not connect the 9V battery if your FaireBot if it is still plugged into your computer! If you do you may cause PERMANENT damage to your Arduino.**

3. Set your FaireBot on a smooth and flat surface. Program the FaireBot by pressing any of the 4 directional buttons and press the Execute button when you are ready to go!

## Part 2: Turn Calibration

As you may have noticed by now, you FaireBot doesn't turn at exactly 45°. (If it does, lucky you!) In order to calibrate your servos you will need to change the value of the "loopVal" variable in the Official FaireBot firmware as shown in the following code:



```
***Located in FaireBot.ino***

int loopVal = 10; //value for the number of servo command loops. determines how long the servos spin

***Located in Motion.ino***

//each function below highlights the method in which FaireBot moves. changing the value of loopVal a

void forward() {
  light(7, true);
  for (int i = 0; i < loopVal; i++) {
    leftServo.write(180);
    delay(15);
    rightServo.write(0);
    delay(15);
  }
  light(7, false);
}

void reverse() {
  light(4, true);
  for (int i = 0; i < loopVal; i++) {
    leftServo.write(0);
    delay(15);
    rightServo.write(180);
    delay(15);
  }
  light(4, false);
}

void left() {
  light(5, true);
  for (int i = 0; i < loopVal; i++) {
    leftServo.write(0);
    delay(15);
    rightServo.write(0);
    delay(15);
  }
  light(5, false);
}

void right() {
  light(6, true);
  for (int i = 0; i < loopVal; i++) {
    leftServo.write(180);
    delay(15);
    rightServo.write(180);
    delay(15);
  }
  light(6, false);
}
```

Change the value of "loopVal" by +5 or -5 units at a time, reprogram your FaireBot and try turning again. Repeat this process until your FaireBot turns exactly 45°. Now that you have fully calibrated your FaireBot, it's time to try some navigation!

**Part 3: Basic Navigation**

The time has arrived. At this point your FaireBot should be properly constructed, programmed, tuned and ready to amaze. Find a nice flat surface, some obstacles and your FaireBot Programming Sheet and lets get started!

1. Go ahead and print out a copy of the FaireBot Program Sheet and lets get started.
2. Place some obstacles for your FaireBot on your work surface, just make sure you don't put up too many obstacles, you don't want to trip!
3. Set up your FaireBot and start writing a program using your Program Sheet. Your program should be made up of individual directional movements (i.e. fwd, fwd, fwd, rht, rht, fwd, lft, lft, fwd, fwd……) Run your bot from the same starting point until it can successfully navigate through your maze.
4. Rejoice! You just wrote a rudimentary and repeatable program for your Fairebot!

## ***Congratulations!***

You have completed the first 3 labs covering the assembly and use of the FaireBot robotics shield. Stay tuned for more labs, including sensor based navigation and wireless control!

# FaireBot Programming Sheet

Name: ____________________

Instructions:
The following table can be used to record motion control programs for your FaireBot. Simply input each directional button press into the while you program your FaireBot and that program can be reused at a later time. Use the following code for each direction:

Forward = **F** Backward = **B** Left = **L** Right = **R**

| **Line #** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| **2** | | | | | | | | |
| **3** | | | | | | | | |
| **4** | | | | | | | | |
| **5** | | | | | | | | |
| **6** | | | | | | | | |
| **7** | | | | | | | | |
| **8** | | | | | | | | |
| **9** | | | | | | | | |
| **10** | | | | | | | | |
| **11** | | | | | | | | |
| **12** | | | | | | | | |
| **13** | | | | | | | | |
| **14** | | | | | | | | |
| **15** | | | | | | | | |
| **16** | | | | | | | | |
| **17** | | | | | | | | |
| **18** | | | | | | | | |
| **19** | | | | | | | | |
| **20** | | | | | | | | |