AdaBox 002

Created by Tyler Cooper

Last updated on 2017-01-20 08:54:00 PM UTC

Guide Contents

Introduction

Hi there!

If you're looking to buy AdaBox001 click here! (http://adafru.it/3193)

If you're looking to buy AdaBox002 click here! (http://adafru.it/3235)

If you're looking to subscribe to AdaBox, click here! (http://adafru.it/tNC)

If you're here, it's because you were given the gift of electronics with an AdaBox! You are a beginner who is getting started with your AdaBox or you just want to relive what it's like being a beginner at electronics again. But most of all, you want to learn how to build and make stuff with electronics! (If, rather than learn electronics, you'd like to look at pictures of cats instead, please check https://www.adafruit.com/galleries/cats-of-engineering (http://adafru.it/oAd))

And, you're in luck: there's *never* been a better time.

Gone are the days where you need thousands of dollars of equipment and lots physics/math background. Nowadays, if you want to learn to work with electronics, you can jump right in for $100 or less, and any sort of computer. And we're talking about learning *a lot* of electronics - from the basics of analog to the complexities of firmware. With a good pack of parts, you can build a base of knowledge that will take you from your first blinking LED to someone who can start prototyping and inventing custom products.

Who is this for?

Anyone with a computer they can install software on, an Arduino or compatible and the ability to type and click a mouse. That's pretty much the minimum. Remember, this guide is specifically for people who have purchased or received an AdaBox subscription!

You don't need to know a lot of physics or math, and just like an Art Degree isn't required for making art and being creative, **you *don't* need to have a computer science degree**. It helps if you're comfortable using computers but that's a skill most people pick up through life.

If you know how to program already - great! If not, don't worry, we'll teach you enough to be dangerous.

Who isn't this for?

While you can follow along without an AdaBox, it will not make as much sense unless you have *all* of the components and more which either came as a gift or purchased yourself - remember, the goal is helping beginners!

If you're an expert, please visit our hundreds other tutorials and jump right in at learn.adafruit.com (http://adafru.it/rdw)

Who are you?

Great question. This is me:

I'm Ladyada, and I love to teach people how to build stuff and how they can be creative with technology.

So, are you ready?

Let's do this thing!

Unboxing AdaBox 002

ADABOX 002 is designed to introduce a new person to the joys of making with electronics. We decided to come up with a fun pack of parts that:

- Could introduce a beginner to making
- Does not require any additional tools or paid software
- Teach electronics and programming skills
- Does not assume any prior experience
- Comes with enough fun parts that could be combined and adapted for months or*years!*

Kit Contents

After a lot of thinking, here's what we came up with:

Feather, USB Cable, & Batteries

1x Adafruit Feather 32u4 Bluefruit LE (http://adafru.it/keO) (fully assembled with stacking headers) - Our take on an 'all-in-one' Arduino-compatible + Bluetooth Low Energy with built in USB and battery charging. Its an Adafruit Feather 32u4 with a BTLE module, ready to rock!

1x USB Cable - A/MicroB (http://adafru.it/iia) - use this to install new code onto your Feather (from any computer) - and to recharge the Feather's battery for portable projects!

1x Lithium Ion Polymer Battery - 3.7v 500mAh (http://adafru.it/drL) (domestic subscriptions only!) - this rechargeable battery can be used to make your Feather project portable. Plug it into the Feather to have it automatically charge over USB. When removed from USB power, the Feather will automatically flip over to battery power.

4x AA Batteries - Use this to power your cute little robot car

Robot Chassis & Assembly Tools

1x Three layer Robot Chassis Kit in Black (http://adafru.it/3244) - This kit gives you everything you need to build the shell of a 2-wheel-drive Mobile Platform Robot to help you channel your inner Mad Max.

1x Adafruit Pocket Screwdriver (http://adafru.it/3284) - To help assemble your cute little car

1x DC Motor and Stepper FeatherWing Add-on (http://adafru.it/sci)(fully assembled) - Lets you use 2 x bi-polar stepper motors or 4 x brushed DC motors (or 1 stepper and 2 DC motors) - or enough motors to power the chassis kit!

Protoyping Parts and Components

1x Half-size Breadboard (http://adafru.it/keP) - Perfect fit for your chassis kit!

1x 4xAA Battery Holder w/ On/Off Switch (http://adafru.it/sfq) - A nice portable battery holder for your batteries.

1x Piezo Buzzer (http://adafru.it/dCD) - Add beeps and boops to your mini robot!

1x 6-pin Male Header w/ Extra Long Pins - Some header pins to make connecting things easier

1x Rubber Bumper Feet (http://adafru.it/dLG) - Helps keep the battery pack safe and secure

Bonus Parts (subscription only!)

1x Coupon for a 4-pack of batteries from Radioshack - Our good friends at Radioshack added this coupon - good for a free 4-pack of AA batteries at any Radioshack store. Great for when you've used your car so much it's out of juice!

1x Enamel Pin - We are oddly obsessed with enamel pins. (http://adafru.it/tbf)

Assembling and Wiring Your Robot

The wiring and assembly is pretty easy, and there is no soldering required. All you need is that handy screwdriver that was included in the AdaBox, and it wouldn't hurt to grab some pliers. You will also need 4 x AA batteries and a micro usb cable to charge the included LiPo battery and upload code to the BLE Feather board.

First off, click the button below for the robot chassis assembly guide. All the parts needed for this are inside the brown box with the 'Custom Black 2WD Robot + extra layer' sticker on it. You can stop at the point of the guide where it has you add in the top layer of the robot. We will be making some slight modifications here. Once you have gotten this far, come back to this guide to continue.

Robot Chassis Instructions
http://adafru.it/taY

Prepare the Battery Box

For this step, you will need the battery box, 4 x AA batteries (these are not in your AdaBox, but in the outer box that the AdaBox was shipped in), the screwdriver, and the sheet of 4 rubber bumpers.

The 4 x AA batteries were added to the outer box that your AdaBox was shipped in.

First, open the battery box, grab out the screw, insert your batteries, and then screw the box shut.**Oh, and make sure you have the box switched to the off position.** Now, take the 4 rubber bumpers and place them as shown in the picture below. Notice how the one bumper on the left side is not in the upper left corner. Important: don't throw away the leftover piece of bumper material, we are going to use that on the next step.

Do not discard the leftover piece of bumper material.

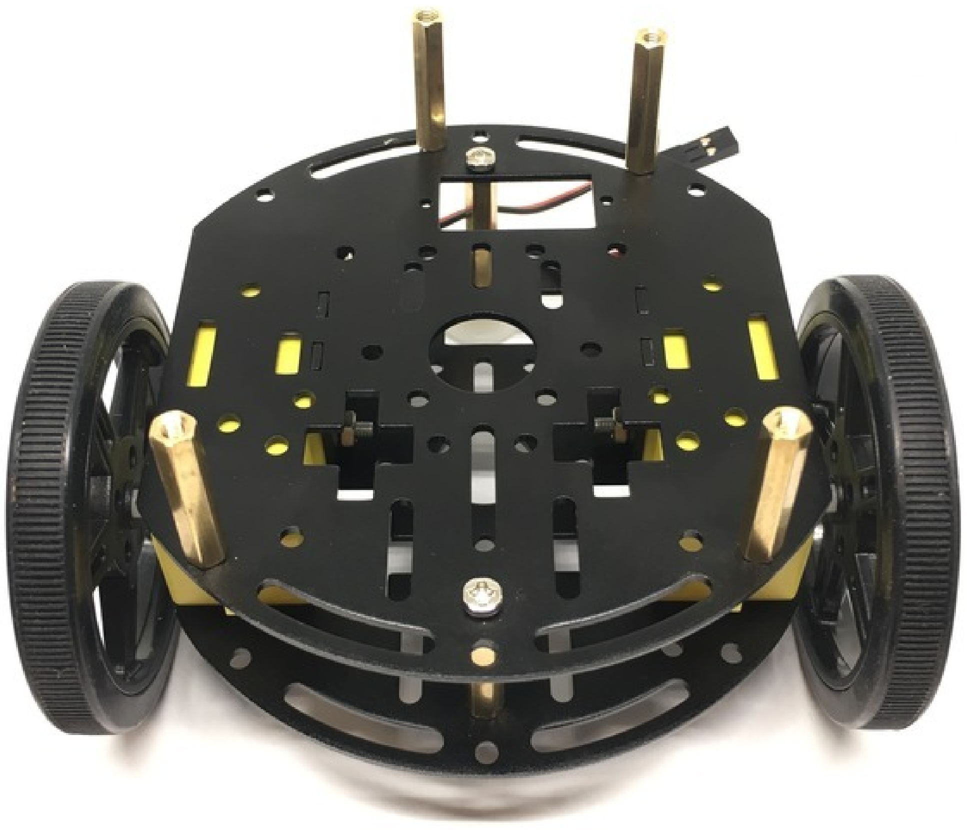
Flip the battery box over and place the scrap piece of the bumper material in the middle. This will help hold the battery box nice and tight between the top and middle plate of your robot.

Back to the Robot Chassis

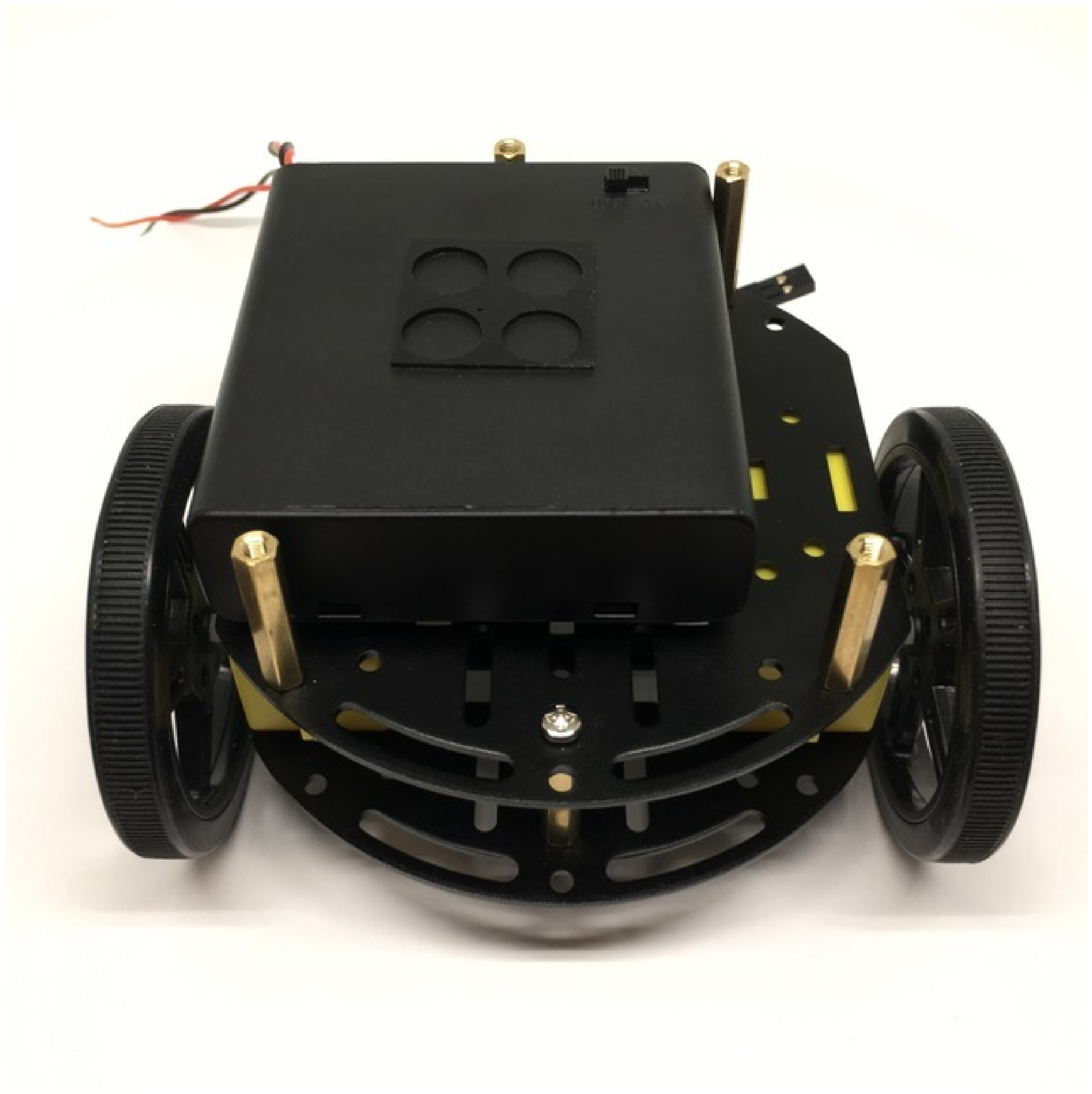
Remember how I told you that you can stop when you get to the part where you assemble the top layer of your robot in the chassis assembly guide? Well, in this step we are going to change the location of where we put some of the posts to make room for our AA battery box.

Take a look at the image below and install the brass stand-offs in the same positions. You can insert the stand-off screws through the middle plate and hand tighten the stand-offs while putting a bit of pressure on the screw with your finger. Or, as a tip, you can screw in the stand-off screws with the flat end of the screwdriver, which can reach through the holes in the bottom plate.

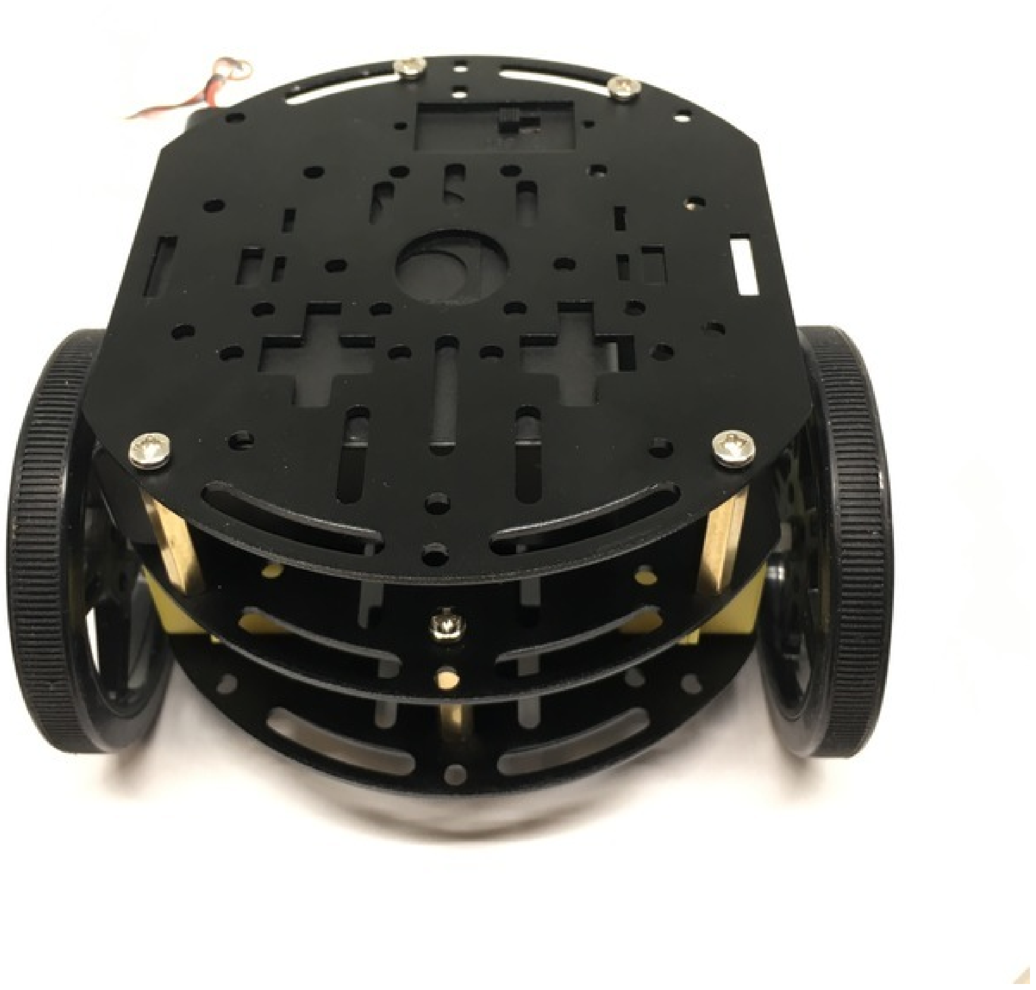
Tip: it may be easier to use the flat side of the screwdriver to screw in the stand-off screws.



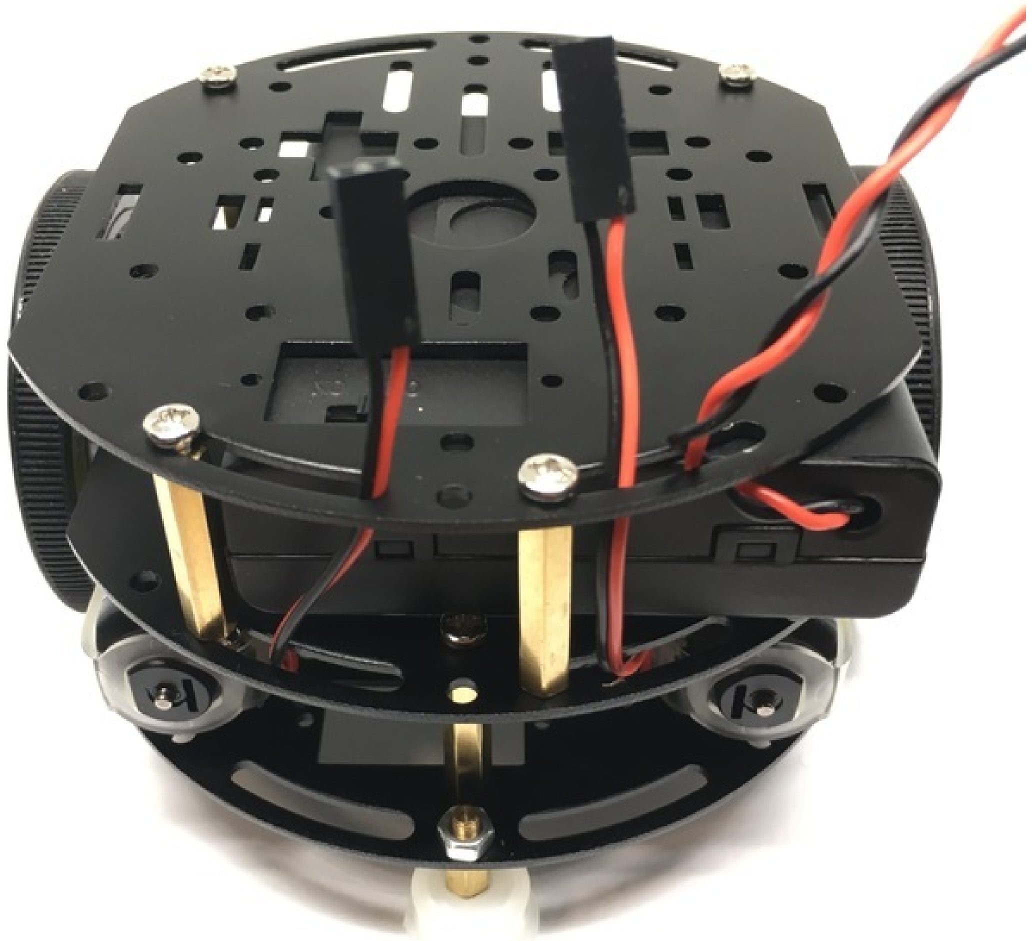
Once you have the stand-offs in place as shown in the image above, let's place the battery box in the correct spot as shown in the image below.



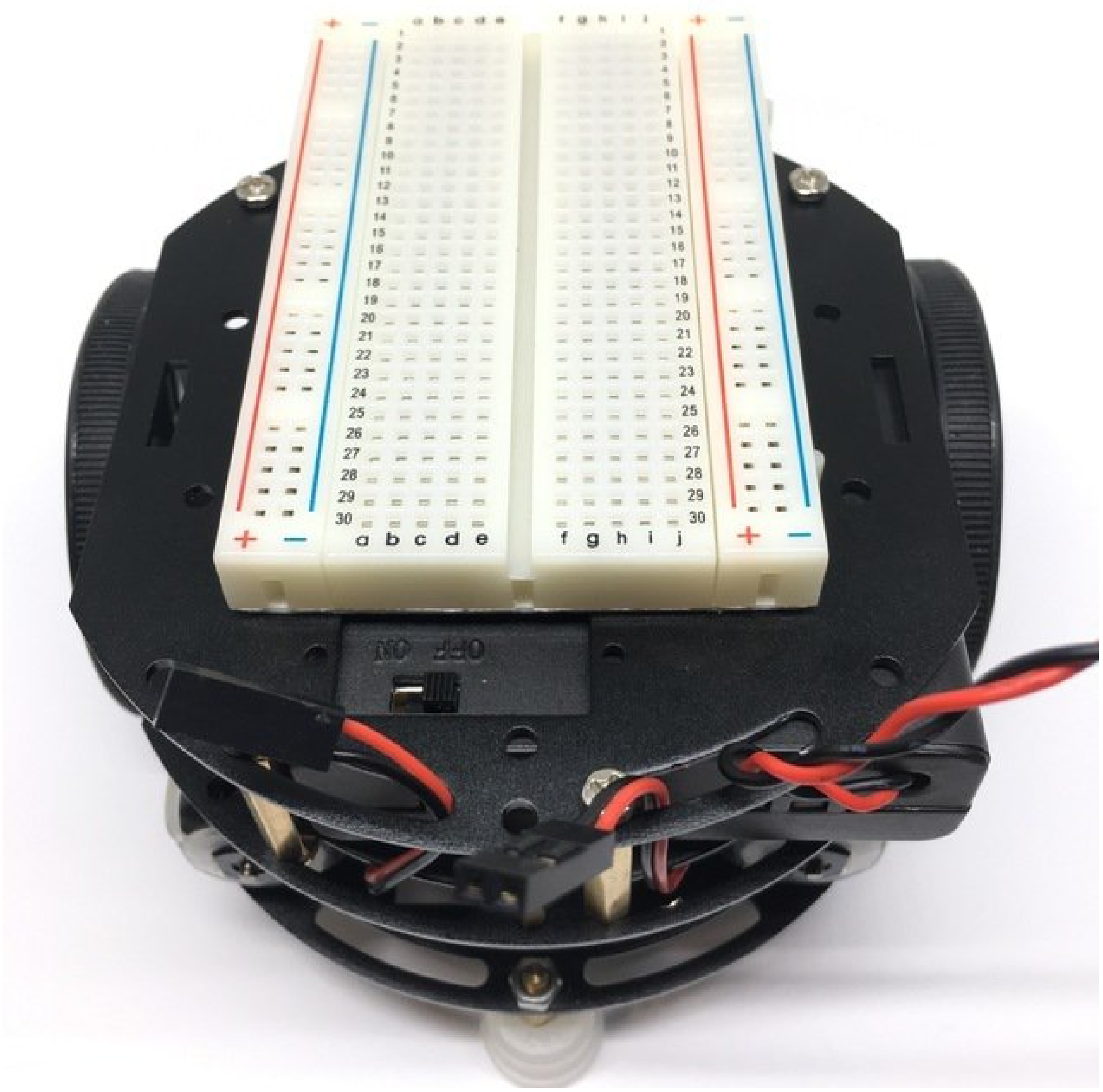
Notice the battery box is lined up on the left side of the middle chassis plate. It should be just in-between the upper left stand-off and the lower left stand-off (not touching either). This will make sure the on-off switch lines up just right in the hole of the top plate. Go ahead and install the top plate now.



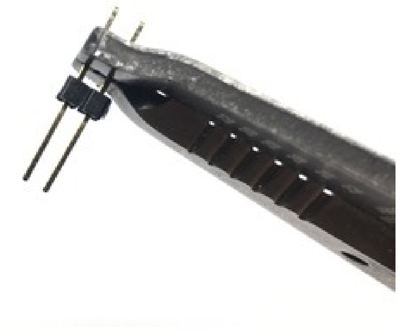
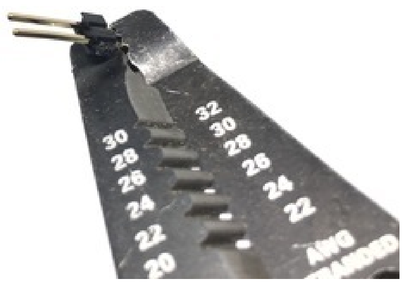
See how the on-off switch is now accessible through the top plate? Now let's take just a minute to route the wires from the motors and the battery box through the chassis like shown in the image below.



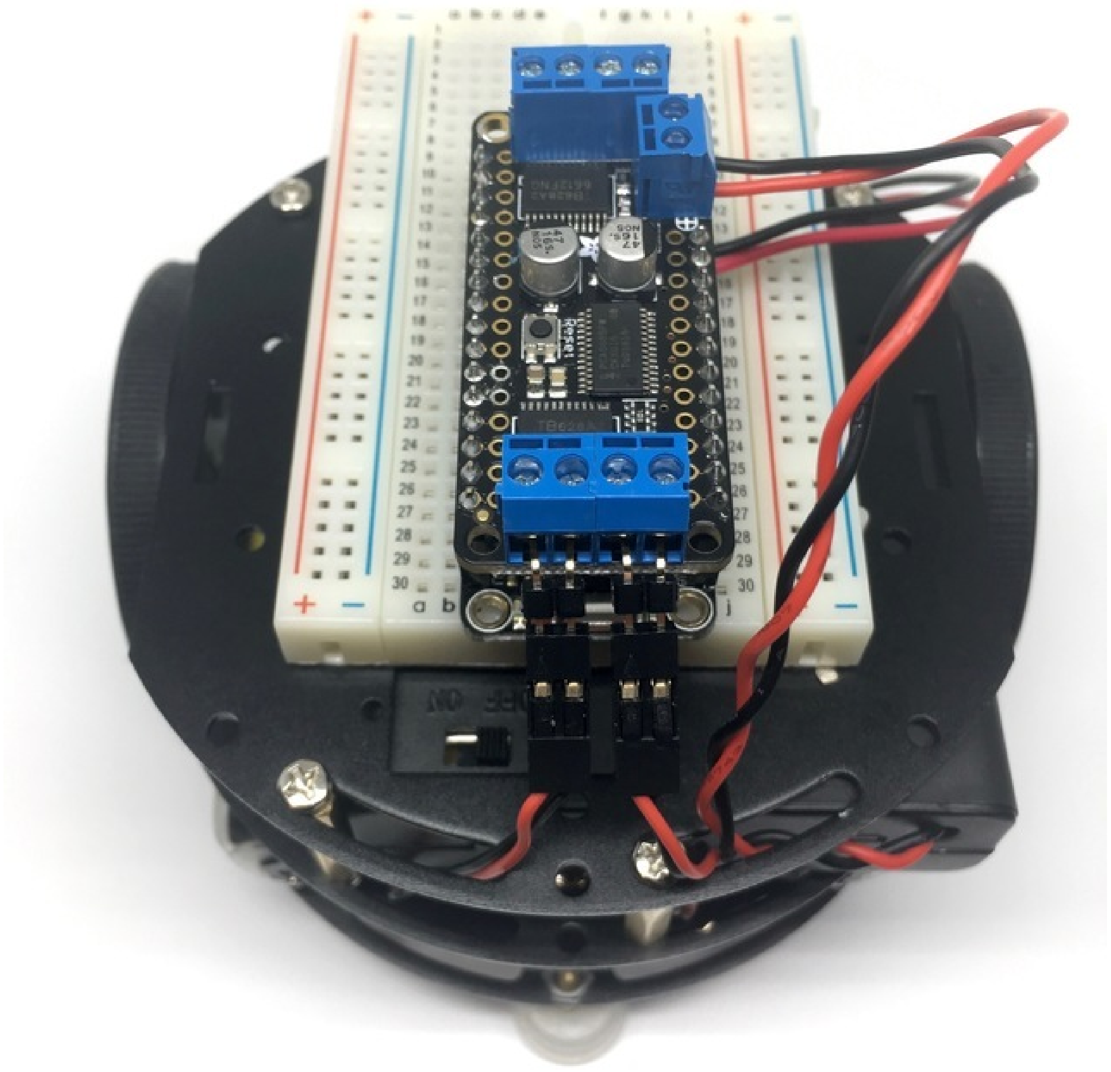
We are getting close! Now let's install the breadboard on top of the robot. Peel off the backing from the foam sticker on the bottom of the breadboard. Take a look at the image below and stick it exactly as shown. Be sure to install the breadboard just as shown in the image below and not sideways or the motor wires won't reach the motor driver Feather Wing.



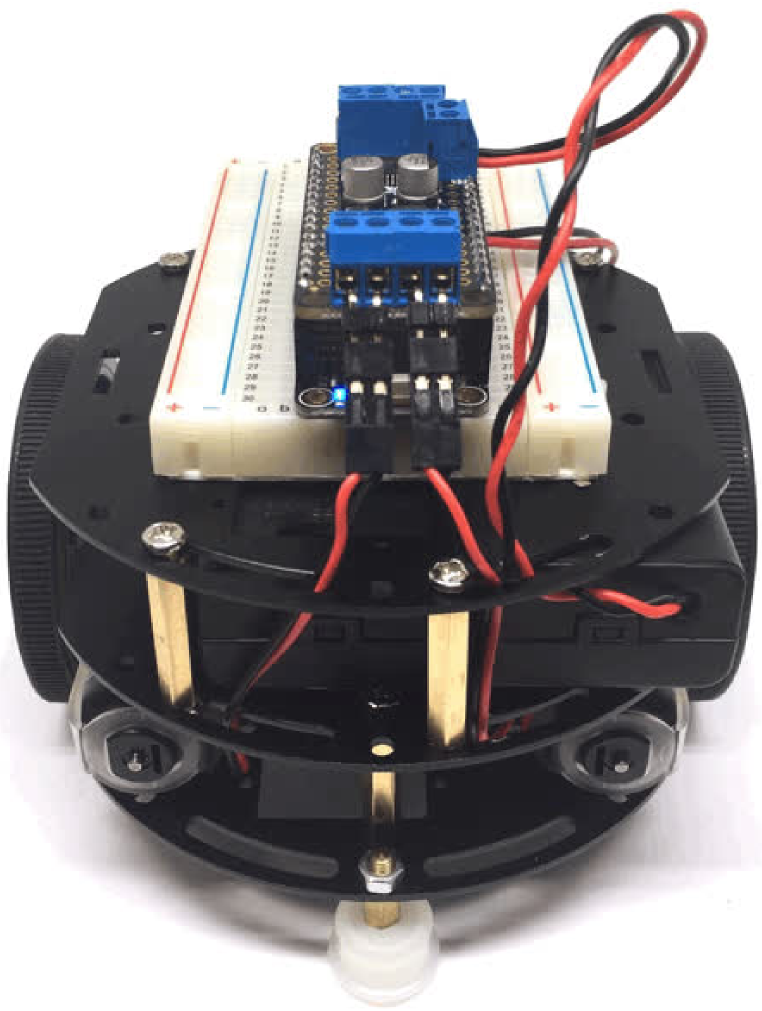
Prepare the Electronics

- 
  For this next step, you will need a set of pliers. First, grab the 6 header pins all attached together and break them into 3 sets of 2 header pins. Then, grab them in the pliers like shown here, and then slightly bend them so they look like the next picture.
- 

With the breadboard in place, go ahead and grab your stacked Bluefruit Feather, and Motor Driver Feather Wing and install it in the breadboard with the USB port facing away from the wires like shown in the image below. Then use your screwdriver and install the bent header pins like shown in the image. Finally, attach the motor wires and power wires as shown in the image. It is important that you have the red and black wires in the correct position. Please reference the image below and triple check that they are installed correctly.



That's it for the first part of getting your robot assembled and wired. Now, let's get this robot moving! On to the code!



How Your Robot Works: The Basics

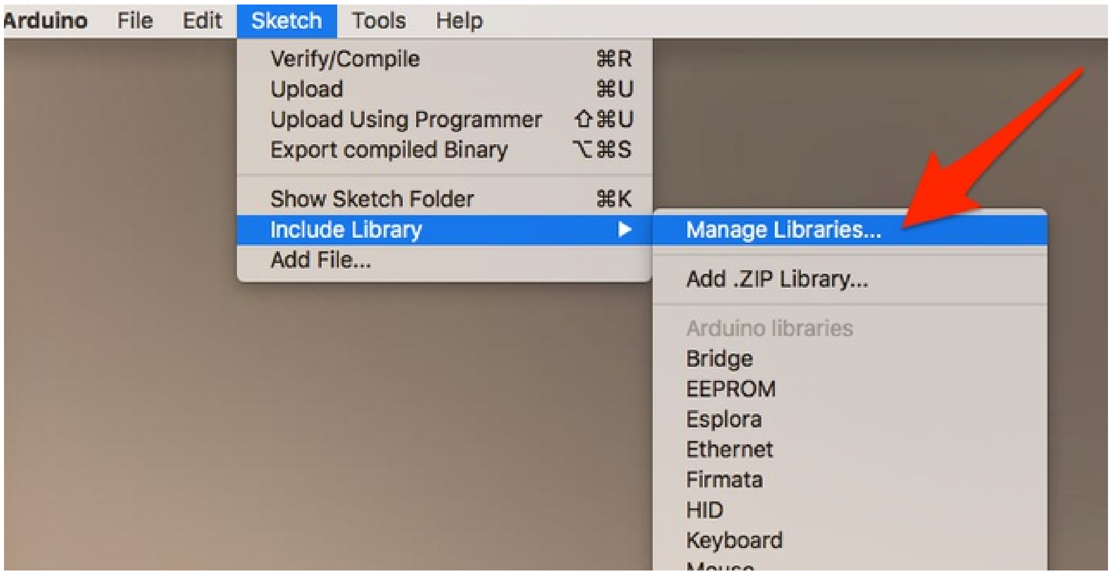
Before we dig into the more complicated code, let's take a minute to break down the simple motor controller code, and how it works to control your robot's motors.

Before going any further, make sure you have a basic understanding of how to program and use an Arduino. Thankfully, we have a lot of great tutorials on how this whole thing works. Click here to get started with Arduino (http://adafru.it/pcg), and then come back to this guide to continue.

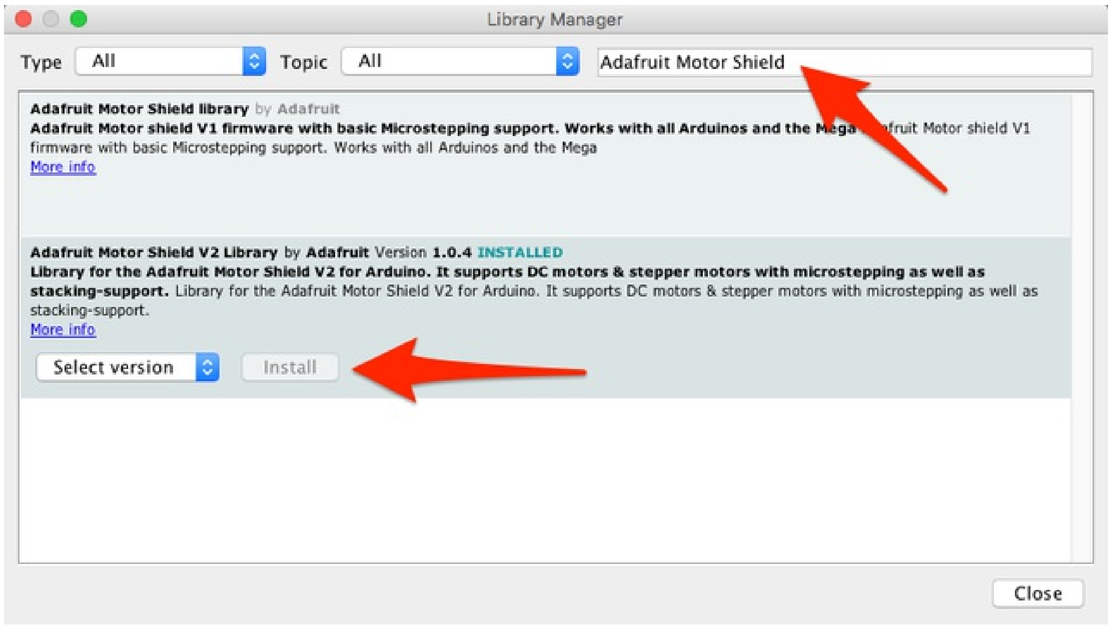
Before going any further, make sure you have a basic understanding of how to program and use an Arduino

Libraries

For your robot to work correctly, you will need to install a couple libraries. To install the libraries, we will use Arduino's handy library manager. Navigate to the library manager like shown in the screenshot below.



Then, all you need to do is search for the library you want to install. For this robot, start by searching for 'Adafruit Motor Shield', you should see two options like this:



Adafruit Motor Shield library by Adafruit
Adafruit Motor shield V1 firmware with basic Microstepping support. Works with all Arduinos and the Mega Adafruit Motor shield V1 firmware with basic Microstepping support. Works with all Arduinos and the Mega
More info

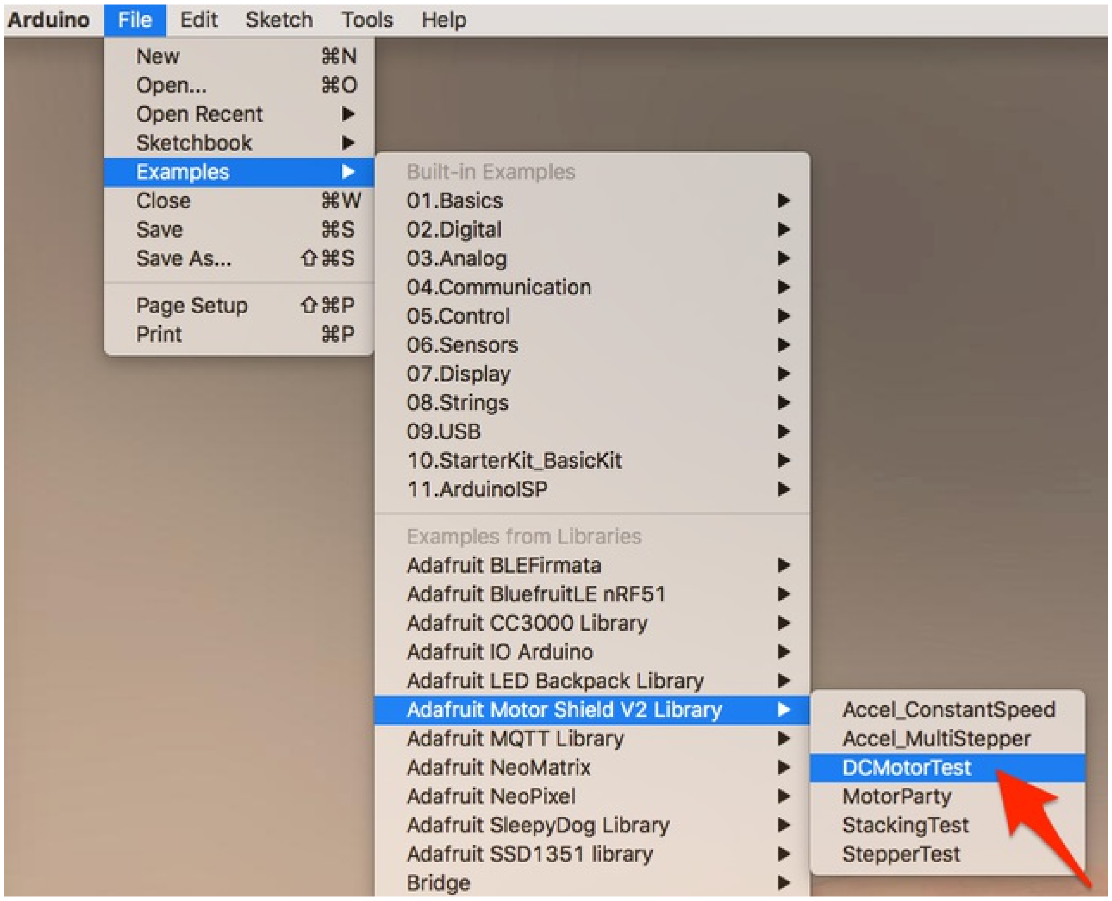
Adafruit Motor Shield V2 Library by Adafruit Version 1.0.4 INSTALLED
Library for the Adafruit Motor Shield V2 for Arduino. It supports DC motors & stepper motors with microstepping as well as stacking-support. Library for the Adafruit Motor Shield V2 for Arduino. It supports DC motors & stepper motors with microstepping as well as stacking-support.
More info

Under the Adafruit Motor Shield V2 Library, select the latest version from the dropdown, then click the install button.

While we won't need it quite yet, you might as well go ahead and search for 'Adafruit BluefruitLE nRF51', and install the latest version of that. We will use this library later on.

DC Motor Test

Now that you have the libraries installed, let's open up the example sketch to try out the DC motors on your robot. You can find the example sketch in the Arduino Examples menu:



Take a look through this example sketch. Before you upload the code and run it on your robot, you need to make a change or two. First off, you will need to change what port the motors are connected to. Let's start by just trying one of the motors. If you remember when we assembled the robot, we connected the motors to the back of the motor shield, in ports M3, and M4. So, in the top part of the code, go ahead and change out the port from 1, to 3 like this:

```
// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(3);
// You can also make another motor on port M2
//Adafruit_DCMotor *myOtherMotor = AFMS.getMotor(2);
```

When you see two forward slashes // in front of text, that is called commenting. Anything written after the // will be ignored by the Arduino. It is a way to communicate with whomever is reading your code.

Place your robot on top of a cup or mug so the wheels are not touching the ground. Select 'Adafruit Feather 32u4' as the board under the Arduino Tools menu, and upload this code to your Feather through the USB connector.

Notice how just the one wheel is going forward, then backward. If you update the code to this:

```
// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(4);
// You can also make another motor on port M2
//Adafruit_DCMotor *myOtherMotor = AFMS.getMotor(2);
```

With the motor port changed to 4, the other motor should now run. That is really the basics of how we will go about controlling the robot.

Let's Get Moving

Now that you know how to make either motor go forward and backward with the example sketch, let's learn the rest of the motor controller basics and write a sketch to get this robot moving. At the top of your sketch, you just need to call out both motors. To do this, you just need to uncomment the 'myOtherMotor' line of code by removing the two forward slashes before the code. Then, set the second motor to 3 like so:

```
// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(4);
// You can also make another motor on port M3
Adafruit_DCMotor *myOtherMotor = AFMS.getMotor(3);
```

In this case, we have myMotor set to M4, which is our right side motor, and myOtherMotor is set to M3, our left side motor.

Using the names myMotor, and myOtherMotor makes things really hard for us to remember which motor is which. So, we can simply change the variable name to whatever we want. Let's simply call them leftMotor, and rightMotor like this:

```
// Set up the left motor on port M4
Adafruit_DCMotor *leftMotor = AFMS.getMotor(4);
// Set up the right motor on port M3
Adafruit_DCMotor *rightMotor = AFMS.getMotor(3);
```

Of course, now that we have changed the variable name, we need to replace any instance of myMotor with leftMotor, and myOtherMotor with rightMotor.

Before we start driving forward, backward, or turning; we need to tell the motor controller how fast we want the motors to go. This is done using the setSpeed function. If we want to make both motors go full speed ahead, we would set them both to 255 like this:

```
// Set the speed to start, from 0 (off) to 255 (max speed)
  leftMotor->setSpeed(255);
  rightMotor->setSpeed(255);
```

To make your robot go forward, all we need to do is tell each motor to go forward like this:

```
leftMotor->run(FORWARD); //LEFT MOTOR FULL STEAM AHEAD!
rightMotor->run(FORWARD); //RIGHT MOTOR FULL STEAM AHEAD!
```

All right, let's put this into a full sketch and give it a shot. For this sketch, Go ahead and upload this sketch to your robot. I have slowed down the speed for safety, but you can update to whatever you want. **Even though it is slowed down, don't forget to set your robot on a mug or a cup to keep the wheels off the ground.**

```
/*
This is a test sketch for the Adafruit assembled Motor Shield for Arduino v2
It won't work with v1.x motor shields! Only for the v2's with built in PWM
control

For use with the Adafruit Motor Shield v2
----> http://www.adafruit.com/products/1438
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
```

```
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
// Or, create it with a different I2C address (say for stacking)
// Adafruit_MotorShield AFMS = Adafruit_MotorShield(0x61);

// Set up the left motor on port M4
Adafruit_DCMotor *leftMotor = AFMS.getMotor(4);
// Set up the right motor on port M3
Adafruit_DCMotor *rightMotor = AFMS.getMotor(3);

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Adafruit Motorshield v2 - Robot Test!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz
}

void loop() {
  // Set the speed to start, from 0 (off) to 255 (max speed)
  leftMotor->setSpeed(100);
  rightMotor->setSpeed(100);

  leftMotor->run(FORWARD); //LEFT MOTOR FULL STEAM AHEAD!
  rightMotor->run(FORWARD); //RIGHT MOTOR FULL STEAM AHEAD!
}
```

Is your robot still not going forward? Be sure to flip the switch on your battery box to 'on'

Ok, so we have the robot going forward. To put the robot in reverse, you just need to change FORWARD to BACKWARD. Turning is just as easy. To turn right, you just need to turn off the right motor, and go FORWARD with the left motor. This brings us to the next bit of motor controller code you need to know, RELEASE. Instead of FORWARD, or BACKWARD, you can also use RELEASE. RELEASE is like pulling the plug on the motor. It will ignore speeds, and direction, and just stop what it is doing. The code for RELEASE looks like this:

```
leftMotor->run(RELEASE);
rightMotor->run(RELEASE);
```

If we take that to our sketch, we can now make the robot turn in circles by releasing the right motor, and going forward with the left motor:

```
/*
This is a test sketch for the Adafruit assembled Motor Shield for Arduino v2
It won't work with v1.x motor shields! Only for the v2's with built in PWM
control

For use with the Adafruit Motor Shield v2
----> http://www.adafruit.com/products/1438
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
// Or, create it with a different I2C address (say for stacking)
// Adafruit_MotorShield AFMS = Adafruit_MotorShield(0x61);

// Set up the left motor on port M4
Adafruit_DCMotor *leftMotor = AFMS.getMotor(4);
// Set up the right motor on port M3
Adafruit_DCMotor *rightMotor = AFMS.getMotor(3);

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Adafruit Motorshield v2 - Robot Test!");

  AFMS.begin();  // create with the default frequency 1.6KHz
  //AFMS.begin(1000);  // OR with a different frequency, say 1KHz
}

void loop() {
  // Set the speed to start, from 0 (off) to 255 (max speed)
  leftMotor->setSpeed(100);
  rightMotor->setSpeed(100);

  leftMotor->run(FORWARD); //LEFT MOTOR FULL STEAM AHEAD!
  rightMotor->run(RELEASE); //RIGHT MOTOR FULL STOP!
}
```

That just about covers the basics, now let's take this to the next level...

Code for Your Robot

To take this robot to the next level, we are going to modify James De Vito's awesome code for the similar Red Robot Rover (http://adafru.it/kBm). In his code, he has multiple control methods where you can either use the Bluefruit App controller or use your phone's accelerometer to drive your robot. For this example, I am going to simplify things and we will just use the control pad. This will free up the four auxiliary buttons for some customization which we will cover in a couple steps.

Libraries

For this code to work, we will need a couple libraries. See the previous step on a really easy way to install these libraries using the Arduino Library manager.

First up is the Adafruit BLE library. Learn more about this library and download it here. (http://adafru.it/lDS)

Next, we will need the Adafruit MotorShield library. Learn more about this library and download it here. (http://adafru.it/t8A)

The Code

Once you have the libraries installed, you will need to download the updated rover code. Click the button below to download.

Download The Code
http://adafru.it/t8B

If you take a good look through this code, you will see it isn't so much more complicated than what we learned in the previous step. There is some complicated code that deals with the bluetooth connection to your phone (which we will use in the next step). One neat feature we added is a way to slowly speed up the motors to full speed so it doesn't pop-a-wheelie when you take off going forward with this bit of code:

```
// speed up the motors
   for (int speed=0; speed < maxspeed; speed+=5) {
    L_MOTOR->setSpeed(speed);
    R_MOTOR->setSpeed(speed);
    delay(5); // 250ms total to speed up
   }
```

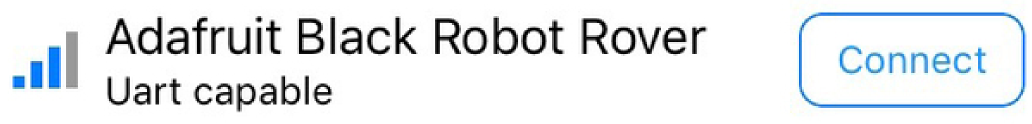
We just use a simple for loop to slowly ramp up the speed until it hits max speed. Learn more about for loops here (http://adafru.it/tb9).

Driving Your Robot

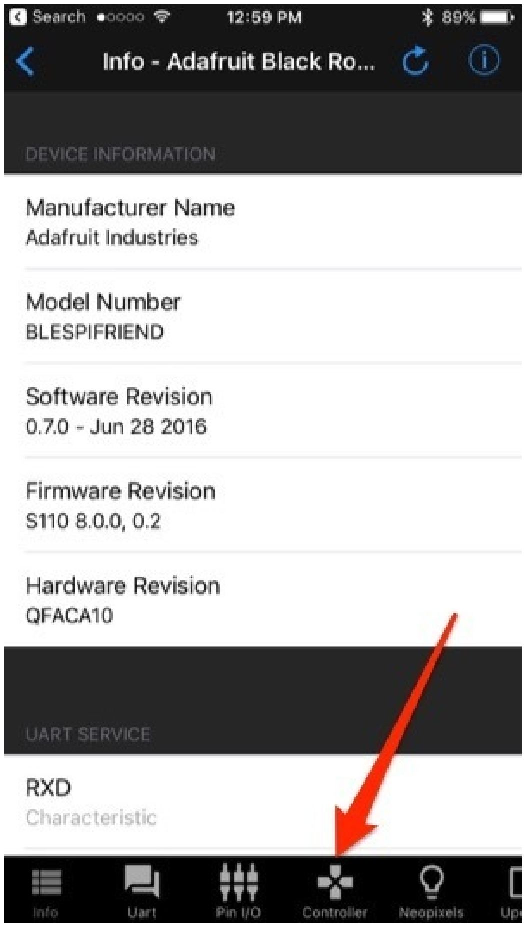
Now it is time to take control of your robot. We will be using the Adafruit Bluefruit LE Connect app. Go ahead and download it on your preferred device.

Bluefruit app for iOS
http://adafru.it/ddu
Bluefruit app for Android
http://adafru.it/f4G

Load up the Bluefruit LE Connect app, and the first thing you will see is a list of available Bluetooth devices to connect to. Find the one that says Adafruit Black Robot Rover, and tap the Connect button.



Once you connect, you will see a whole bunch of device information. At the bottom of the app, tap the Controller button.



Manufacturer Name
Adafruit Industries

Model Number
BLESPIFRIEND

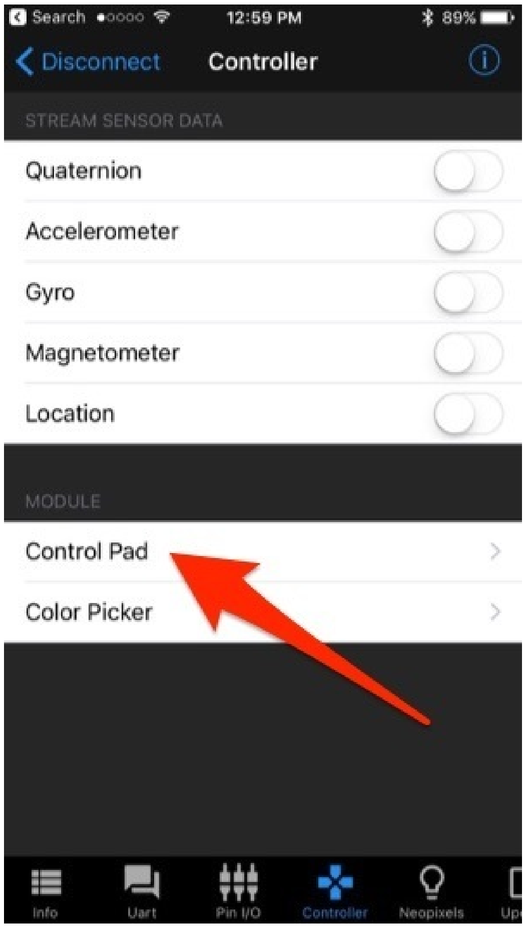
Software Revision
0.7.0 - Jun 28 2016

Firmware Revision
S110 8.0.0, 0.2

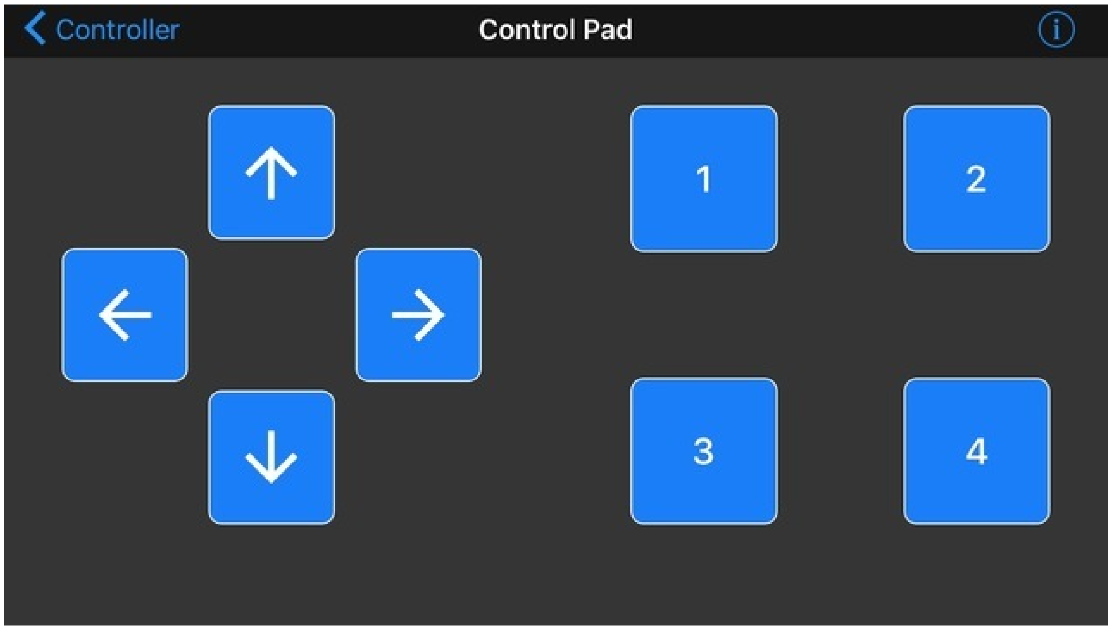
Hardware Revision
QFACA10

RXD
Characteristic

On the next screen, you will see some advanced features of the app. For now, click the Control Pad link.



The Control Pad should load up, and if you turn your phone sideways, it will expand to fill the screen.



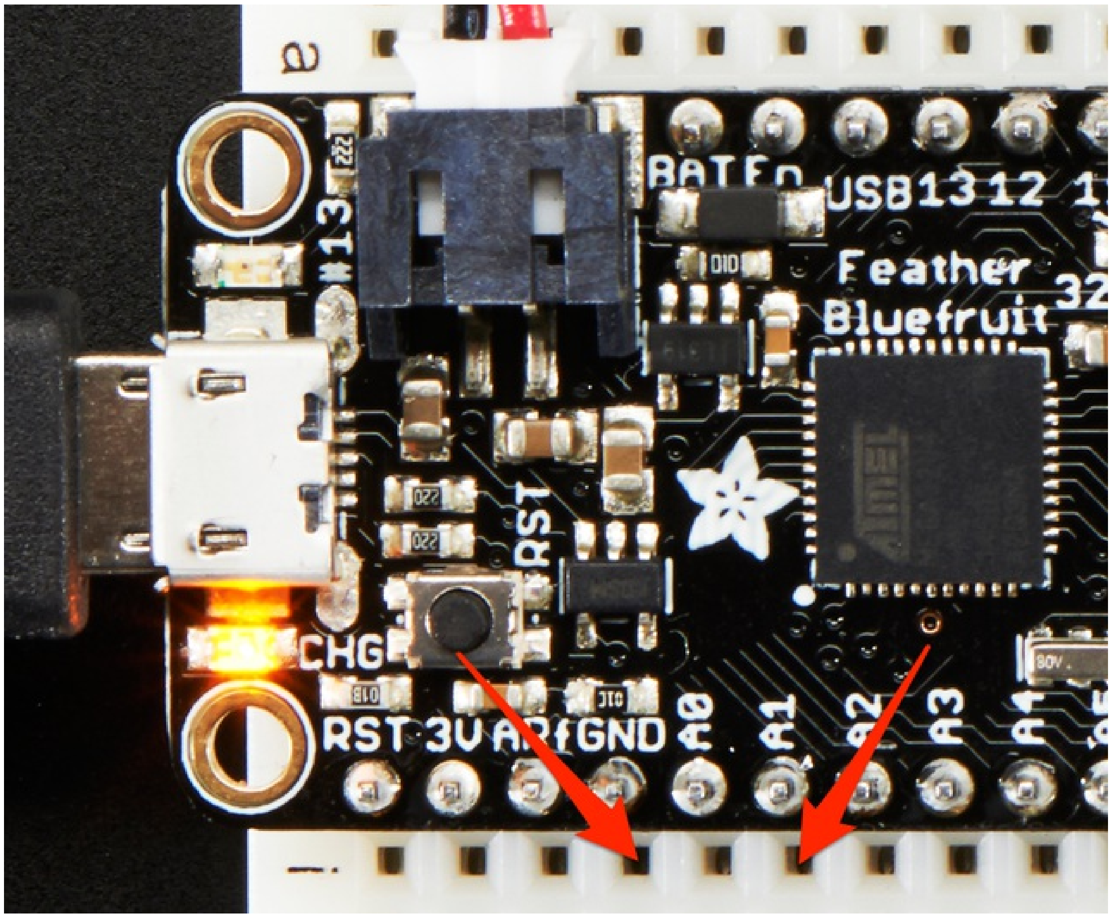
Go ahead and use the arrows to drive your rover. Be careful, this little guy is fast! Be sure to place it on the floor first!

Make Your Robot Sing

Packed in your AdaBox 002 is a little piezo buzzer. This will allow you to make your robot beep and boop.

Want to learn more about piezo buzzers and how they work? Click here.
http://adafru.it/t8C

The spacing of the piezo pins is perfect to put one side in the GND and the other in A1 as shown below. It doesn't matter which side of the buzzer goes into which spot.



The Code

For the fun of it, lets take the code from the previous step and make it play the Super Mario Bros. theme whenever you press button 1 on the controller. First, lets take a look at the code for the music.

We are going to use a variation of Becky's code from thewearable piezo project (http://adafru.it/t8D) (Becky's code is a varition of Tiaga Galo's code (http://adafru.it/aWC) - yay open source).

```
#define toneC    1911
#define toneC1    1804
#define toneD    1703
#define toneEb    1607
```

```
#define toneE    1517
#define toneF    1432
#define toneF1    1352
#define toneG    1276
#define toneAb    1204
#define toneA    1136
#define toneBb    1073
#define toneB    1012
#define tonec       955
#define tonec1      902
#define toned       851
#define toneeb      803
#define tonee       758
#define tonef       716
#define tonef1      676
#define toneg       638
#define toneab      602
#define tonea       568
#define tonebb      536
#define toneb       506

#define tonep       0

int speaker = A1;
long vel = 20000;
boolean hasplayed = false;

void setup() {
  pinMode(speaker, OUTPUT);
  delay(2000);
}

int melod[] = {tonec, toneG, toneE, toneA, toneB, toneBb, toneA, toneG, tonee, toneg, tonea, tonef, toneg, tonee, tonec, toned, toneB};
int ritmo[] = {18, 18, 18, 12, 12, 6, 12, 8, 8, 8, 12, 6, 12, 12, 6, 6, 6};

void loop() {
  if (hasplayed == true){ return;}
  for (int i=0; i<17; i++) {
    int tom = melod[i];
    int tempo = ritmo[i];

    long tvalue = tempo * vel;

    tocar(tom, tvalue);

    delayMicroseconds(1000);
  }     delay(1000);

  hasplayed = true;
}

void tocar(int tom, long tempo_value) {
  long tempo_gasto = 0;
  while (tempo_gasto < tempo_value) {
    digitalWrite(speaker, HIGH);
    delayMicroseconds(tom / 2);

    digitalWrite(speaker, LOW);
    delayMicroseconds(tom/2);
    tempo_gasto += tom;
  }
}
```

Now, if we combine that code with the BLE controller code, and have it play the music when you tap button 1 on the controller...

```
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#if not defined (_VARIANT_ARDUINO_DUE_X_)
  #include <SoftwareSerial.h>
#endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#include <Wire.h>
#include <Adafruit_MotorShield.h>

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// And connect 2 DC motors to port M3 & M4 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(3);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(4);

//Name your RC here
String BROADCAST_NAME = "Adafruit Black Robot Rover";

String BROADCAST_CMD = String("AT+GAPDEVNAME=" + BROADCAST_NAME);

Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

#define toneC    1911
#define toneC1    1804
#define toneD    1703
#define toneEb    1607
#define toneE    1517
#define toneF    1432
#define toneF1    1352
#define toneG    1276
#define toneAb    1204
#define toneA    1136
#define toneBb    1073
#define toneB    1012
#define tonec       955
#define tonec1      902
#define toned       851
#define toneeb      803
#define tonee       758
#define tonef       716
#define tonef1      676
#define toneg       638
#define toneab      602
#define tonea       568
#define tonebb      536
#define toneb       506

#define tonep       0
```

```
int speaker = A1;
long vel = 20000;
boolean hasplayed = false;

// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// function prototypes over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// the packet buffer
extern uint8_t packetbuffer[];

char buf[60];

/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
void setup(void)
{
  Serial.begin(9600);

  AFMS.begin();  // create with the default frequency 1.6KHz

  // turn on motors
  L_MOTOR->setSpeed(0);
  L_MOTOR->run(RELEASE);

  R_MOTOR->setSpeed(0);
  R_MOTOR->run(RELEASE);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Robot Controller Example"));
  Serial.println(F("-----------------------------------------"));

  /* Initialize the module */
  BLEsetup();

  pinMode(speaker, OUTPUT);

  L_MOTOR->setSpeed(155);
  R_MOTOR->setSpeed(155);
}

int melod[] = {tonec, toneG, toneE, toneA, toneB, toneBb, toneA, toneG, tonee, toneg, tonea, tonef, toneg, tonee, tonec, toned, toneB};
int ritmo[] = {18, 18, 18, 12, 12, 6, 12, 8, 8, 8, 12, 6, 12, 12, 6, 6, 6};

void loop(void)
{
    // read new packet data
    uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  readController();
}

bool readController(){

 // Buttons
  if (packetbuffer[1] == 'B') {

    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';

    // Serial.println(buttnum);

    if (pressed) {
      if(buttnum == 1){
        if (hasplayed == true){ return;}
          for (int i=0; i<17; i++) {
            int tom = melod[i];
            int tempo = ritmo[i];

            long tvalue = tempo * vel;

            tocar(tom, tvalue);

            delayMicroseconds(1000);
            }      //delay(1000);

            hasplayed = true;
      }

      if(buttnum == 2){

      }

      if(buttnum == 3){

      }

      if(buttnum == 4){

      }

      if(buttnum == 5){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(FORWARD);
      }

      if(buttnum == 6){
        L_MOTOR->run(BACKWARD);
        R_MOTOR->run(BACKWARD);
      }

      if(buttnum == 7){
        L_MOTOR->run(RELEASE);
        R_MOTOR->run(FORWARD);
      }

      if(buttnum == 8){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(RELEASE);
      }
  }
  else {
      L_MOTOR->run(RELEASE);
      R_MOTOR->run(RELEASE);
      hasplayed = false;
    }
}
}

void BLEsetup(){
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));
  }
  Serial.println( F("OK!") );

  /* Perform a factory reset to make sure everything is in a known state */
  Serial.println(F("Performing a factory reset: "));
  if (! ble.factoryReset() ){
       error(F("Couldn't factory reset"));
  }

  //Convert the name change command to a char array
  BROADCAST_CMD.toCharArray(buf, 60);

  //Change the broadcast device name here!
  if(ble.sendCommandCheckOK(buf)){
    Serial.println("name changed");
  }
  delay(250);

  //reset to take effect
  if(ble.sendCommandCheckOK("ATZ")){
    Serial.println("resetting");
  }
  delay(250);

  //Confirm name change
  ble.sendCommandCheckOK("AT+GAPDEVNAME");

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game controller, etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this point!

  /* Wait for connection */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("******************"));
  // Set Bluefruit to DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("******************"));
}

void tocar(int tom, long tempo_value) {
  long tempo_gasto = 0;
  while (tempo_gasto < tempo_value) {
    digitalWrite(speaker, HIGH);
    delayMicroseconds(tom / 2);

    digitalWrite(speaker, LOW);
    delayMicroseconds(tom/2);
    tempo_gasto += tom;
  }
}
```

Upgrade Your Robot

One great perk of the AdaBox is the included 10% off discount. What better way to take advantage of the discount than to upgrade your new robot friend?

Redeeming your coupon code is really easy. Just log in to your Adafruit account, and click on the My Account link in the upper right of the top nav bar.

Once you are in My Account, click on the 'Gift Certificates and Coupons' link in the left sidebar.

Your coupon code will appear under 'account coupons.'

This code is associated with your AdaBox order and can be used once per AdaBox subscription. You must be logged into the Adafruit account associated with your AdaBox order to redeem your coupon code. If you received AdaBox as a gift, the code will appear in your account once you link your Adafruit account to the AdaBox you received.

Shipping costs do not count towards free items. Discount codes do not apply to shipping costs. Discount codes do not apply to gift certificates and software. Only one discount code can be applied if available. Discount does not apply to orders placed before the sale time start. Discount can not combine with reseller, educational, or any other discounted orders.

Now for the fun part, lets upgrade your robot...

Make Your Robot Autonomous

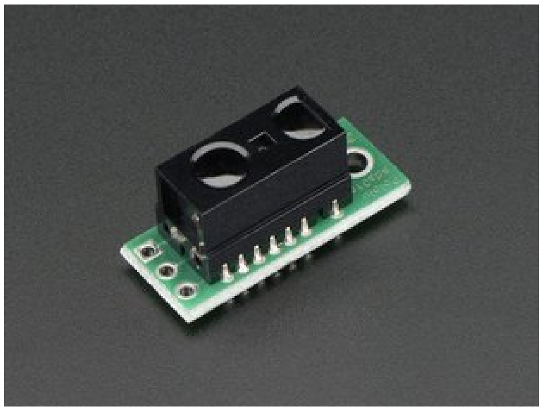
While it is definitely neat to use your phone to control your robot, it is time to set your little robot free. The first step in making your robot autonomous is to add proximity sensors so it can avoid obstacles.

The simplest way to do this is to add a couple robot whiskers to sense when it physically runs into a wall or object. You simply write some code to listen for when the switch has been triggered, stop, turn around, and go forward again. These Micro Switches with a wire are perfect for this.

Micro Switch w/Wire - Three Terminals

PRODUCT ID: 820
Micro-switches are often found in arcade buttons and joysticks but they're also really handy in any kind of mechatronics project or when you need a basic sensor. They are always 'tactile'...
http://adafru.it/t8E
$2.95
IN STOCK

While this is a great way to navigate around obsticles, I prefer to use a something a bit more intelligent so my little robot doesn't actually have to run into things to navigate. There are plenty of distance sensor options out there (there is a whole category of them (http://adafru.it/t8F) on the Adafruit shop), but the sensor I am going to use is the neat little IR sensor from from Sharp.



Sharp GP2Y0D810Z0F Digital Distance Sensor with Pololu Carrier

PRODUCT ID: 1927
If you're like me, you've always wanted a smaller and faster way to sense an object between 2 and 10 centimeters away. If you're really like me, you've wanted the sensor...
http://adafru.it/t9a
$6.95
IN STOCK

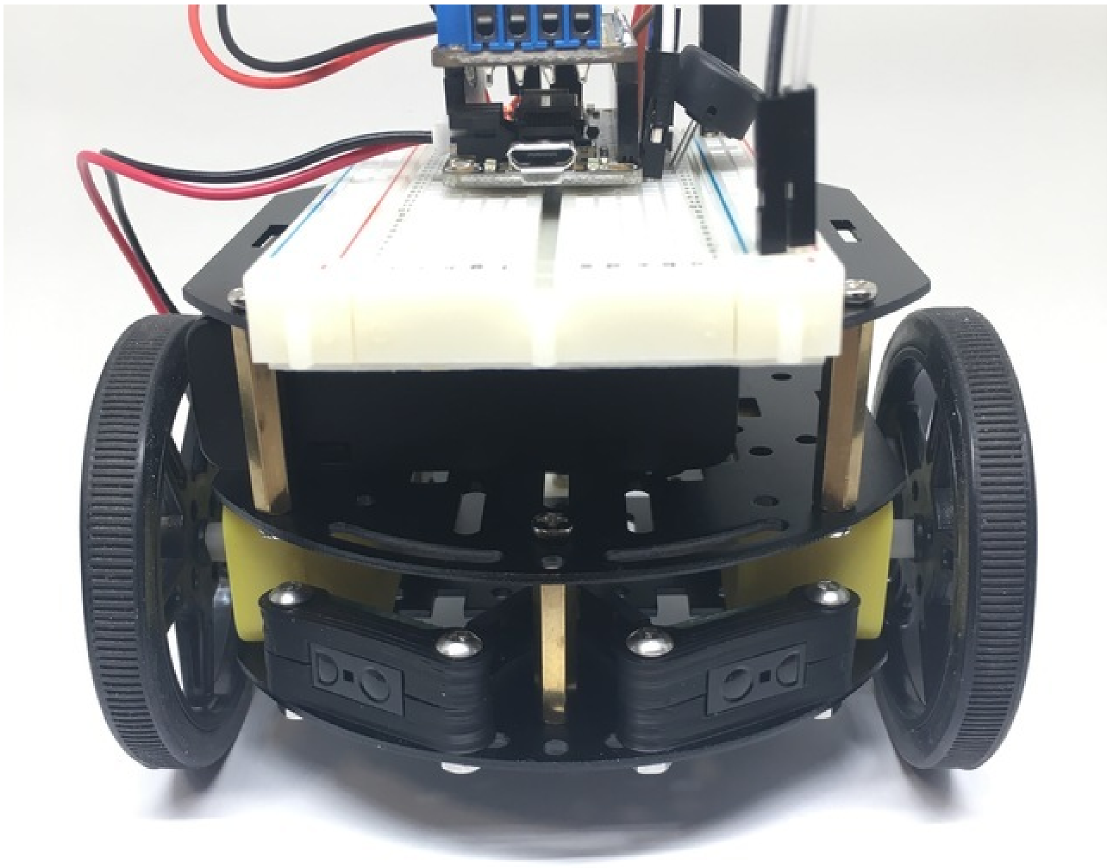
The obvious benefit here is the size, but also the price. This little sensor will sense an object about 10 centimeters away, and acts like a normal switch. There is a pin on the board that is normally high and switches to low when it senses an object. Because of the small size, we can put two of these on our little robot.

Mounting the Sensors

There are a ton of ways you can mount this sensor to your robot. The easiest way would be to just a bit of double sided foam tape to stick it in place, but I decided to make a super simple 3D printed mount.

Click Here to Download the Mount
http://adafru.it/t9b

I used a couple M2.5 screws to secure the mounts to the robot (required a drill bit to clean up the holes in the 3D print). I wired up the left sensor to pin A4, and the right sensor to pin A5.



The Code

The code to make your robot take advantage of its new eyes is very straight forward. For now, we are going to focus on the Adafruit MotorShield library.

[Learn More About the Adafruit MotorShield Library](http://adafru.it/t9c)
http://adafru.it/t9c

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// And connect 2 DC motors to port M3 & M4 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(3);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(4);

// And connect the Sharp distance sensors
int leftSensor = A4;
int rightSensor = A5;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Adafruit Motorshield v2 - DC Motor test!");

  pinMode(leftSensor, INPUT); // set up distance sensor pins
  pinMode(rightSensor, INPUT);

  AFMS.begin();  // create with the default frequency 1.6KHz

}

void loop() {
  L_MOTOR->setSpeed(200);
  R_MOTOR->setSpeed(200);
  L_MOTOR->run(FORWARD);
  R_MOTOR->run(FORWARD);

  while (digitalRead(rightSensor) == LOW){
    L_MOTOR->setSpeed(100);
    R_MOTOR->setSpeed(100);
    L_MOTOR->run(BACKWARD);
    R_MOTOR->run(RELEASE);
  }

  while (digitalRead(leftSensor) == LOW){
    L_MOTOR->setSpeed(100);
    R_MOTOR->setSpeed(100);
    L_MOTOR->run(RELEASE);
    R_MOTOR->run(BACKWARD);
  }
}
```

As you can see from the above code, there isn't a whole lot going on here. All we are doing is reading one of the distance sensors, and if it senses an object we reverse the opposite side motor until the object is no longer detected. We also slow things down quite a bit, as this little robot is so quick it loves to pop wheelies when it starts and stops quickly.

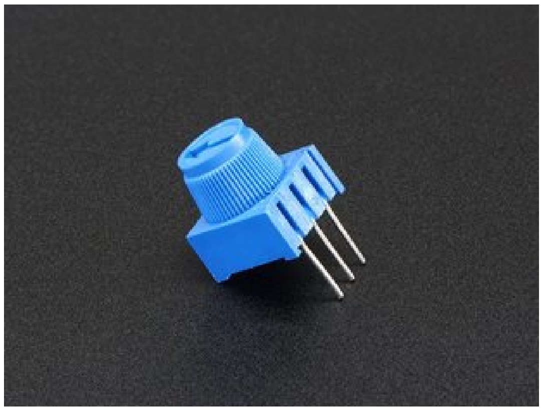
This is just the beginning of what you can do with distance sensing. The next steps are up to you. You can integrate this code into the bluetooth controller code to turn on and off auto mode with a button press. What other ideas can you think of?

Control Your Robot's Speed

Instead of hard coding a specific speed for your robot, here we will use a couple different ways to adjust your robot speed without having to constantly upload new code.

AdaBox 001 Bonus! Use the potentiometer from AdaBox 001 for this modification.

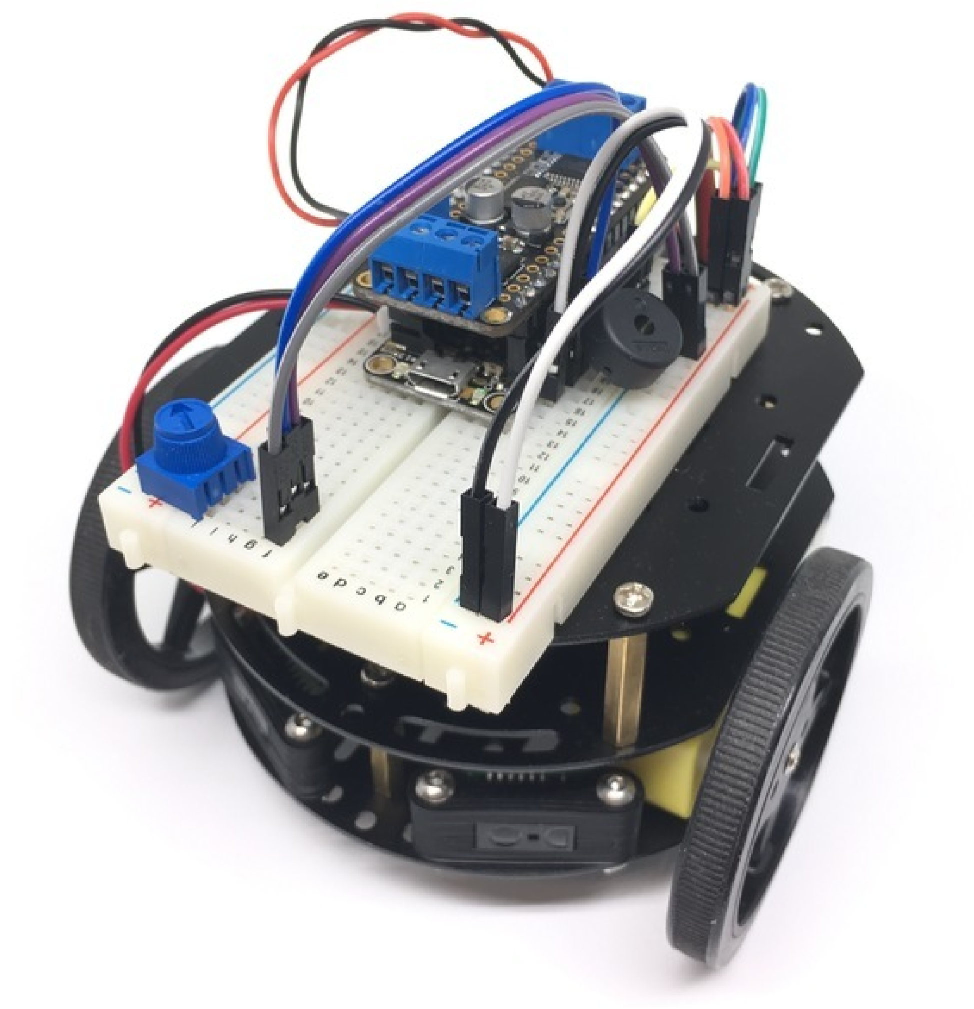
The first way we are going to adjust the speed is with a simple breadboard trim potentiometer. If you were lucky enough to get the first AdaBox, you have one of these already. If not, don't worry, they are super cheap! Pick one up on the Adafruit Shop:



Breadboard trim potentiometer

PRODUCT ID: 356
These are our favorite trimpots, perfect for breadboarding and prototyping. They have a long grippy adjustment knob and with 0.1" spacing they plug into breadboards or perfboards...
http://adafru.it/tba
$1.25
IN STOCK

Potentiometers, or Pots for short, are variable resistors that allow us to send different voltages to the Feather analog pin. Wiring it up is super simple. Just connect one of the outside pins to a 3.3V pin, and the other outside pin to ground. Then, connect the middle pin to the A0 pin on the Feather. For a reminder on which pin is which on the Bluefruit Feather, click here (http://adafru.it/tbb).



The code to get this all working is incredibly simple. Just copy and paste in the code below to the top of your main loop.

```
//Set your motor speed
int reading  = analogRead(A0);
L_MOTOR->setSpeed(map(reading, 0, 1023, 0, 255));
R_MOTOR->setSpeed(map(reading, 0, 1023, 0, 255));
```

So, it should look something like this now:

```
void loop(void)
{

  //Set your motor speed
  int reading  = analogRead(A0);
  L_MOTOR->setSpeed(map(reading, 0, 1023, 0, 255));
  R_MOTOR->setSpeed(map(reading, 0, 1023, 0, 255));


  // read new packet data
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
  readController();

}
```

What we are doing here is reading that analog pin, and the Feather is going to pull a number from 0 to 1023, depending on which direction the arrow is facing on your pot. Because the motor controller needs a value from 0 to 255, we are using the map() function (http://adafru.it/tbc).

Go ahead and upload this code to your robot, and turn the pot, then press forward on the controller to see how it works. Keep changing to direction of the arrow on the pot to adjust the speed.

Using the Controller to Control Speed

If you would rather just control your robot's speed using the extra 4 buttons on the controller, it is also pretty easy to set up. First off, we need to set up a global speed variable. Anywhere above the sketch setup, add in something like this:

```
int robotSpeed = 100;
```

In the main loop, we can then use that variable to set the motor speed using:

```
L_MOTOR->setSpeed(robotSpeed);
R_MOTOR->setSpeed(robotSpeed);
```

Then, all we need to do is increment the speed up every time we press the 1 button, and down every time we press the 3 button.

```
if(buttnum == 1){
 if(robotSpeed <= 245){
  robotSpeed = robotSpeed + 10;
    }
}

if(buttnum == 2){
}

if(buttnum == 3){
 if(robotSpeed >=10){
  robotSpeed = robotSpeed - 10;
 }
}

if(buttnum == 4){
}
```

Here is the whole block of code for you to copy and paste in:

```
/*********************************************************************
 This is an example for our nRF51822 based Bluefruit LE modules

 Modified to drive a 3-wheeled BLE Robot Rover! by http://james.devi.to

 Pick one up today in the Adafruit shop!

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!
 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#if not defined (_VARIANT_ARDUINO_DUE_X_)
  #include <SoftwareSerial.h>
#endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#include <Wire.h>
#include <Adafruit_MotorShield.h>

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// And connect 2 DC motors to port M3 & M4 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(3);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(4);

//Name your RC here
String BROADCAST_NAME = "Adafruit Black Robot Rover";

String BROADCAST_CMD = String("AT+GAPDEVNAME=" + BROADCAST_NAME);

Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// function prototypes over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// the packet buffer
extern uint8_t packetbuffer[];

char buf[60];

int robotSpeed = 100;

/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
void setup(void)
{
  Serial.begin(9600);

  AFMS.begin();  // create with the default frequency 1.6KHz

  // turn on motors
  L_MOTOR->setSpeed(0);
  L_MOTOR->run(RELEASE);

  R_MOTOR->setSpeed(0);
  R_MOTOR->run(RELEASE);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Robot Controller Example"));
  Serial.println(F("-----------------------------------------"));

  /* Initialize the module */
  BLEsetup();

  //Set your motor speed (255 Max)
  L_MOTOR->setSpeed(robotSpeed);
  R_MOTOR->setSpeed(robotSpeed);
}

void loop(void)
{
  L_MOTOR->setSpeed(robotSpeed);
  R_MOTOR->setSpeed(robotSpeed);

  // read new packet data
  uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);

  readController();

}


bool readController(){

 // Buttons
  if (packetbuffer[1] == 'B') {

    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';

    if (pressed) {
      if(buttnum == 1){
        if(robotSpeed <= 245){
          robotSpeed = robotSpeed + 10;
        }

      }

      if(buttnum == 2){

      }

      if(buttnum == 3){
        if(robotSpeed >=10){
          robotSpeed = robotSpeed - 10;
        }
      }
      if(buttnum == 4){

      }

      if(buttnum == 5){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(FORWARD);
      }

      if(buttnum == 6){
        L_MOTOR->run(BACKWARD);
        R_MOTOR->run(BACKWARD);
      }

      if(buttnum == 7){
        L_MOTOR->run(RELEASE);
        R_MOTOR->run(FORWARD);
      }

      if(buttnum == 8){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(RELEASE);
      }
  }
  else {
      L_MOTOR->run(RELEASE);
      R_MOTOR->run(RELEASE);
    }
}
}

void BLEsetup(){
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));
  }
  Serial.println( F("OK!") );

  /* Perform a factory reset to make sure everything is in a known state */
  Serial.println(F("Performing a factory reset: "));
  if (! ble.factoryReset() ){
       error(F("Couldn't factory reset"));
  }

  //Convert the name change command to a char array
  BROADCAST_CMD.toCharArray(buf, 60);

  //Change the broadcast device name here!
  if(ble.sendCommandCheckOK(buf)){
    Serial.println("name changed");
  }
  delay(250);

  //reset to take effect
  if(ble.sendCommandCheckOK("ATZ")){
    Serial.println("resetting");
  }
  delay(250);
  //Confirm name change
  ble.sendCommandCheckOK("AT+GAPDEVNAME");

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game controller, etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this point!

  /* Wait for connection */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("******************************"));

  // Set Bluefruit to DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("******************************"));
}
```