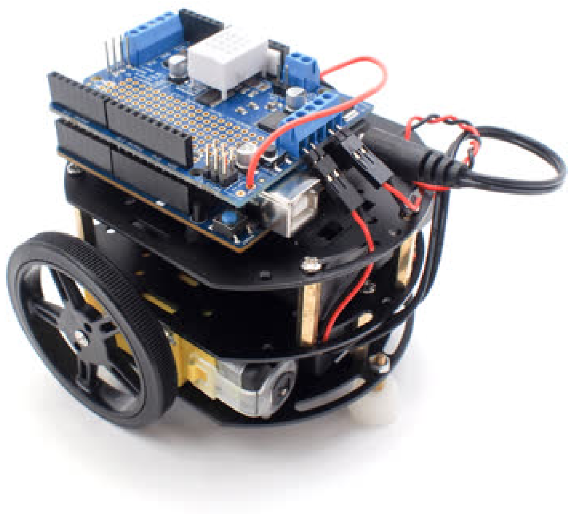
adafruit learning system

CurieBot: Arduino 101 Mini Robot Rover

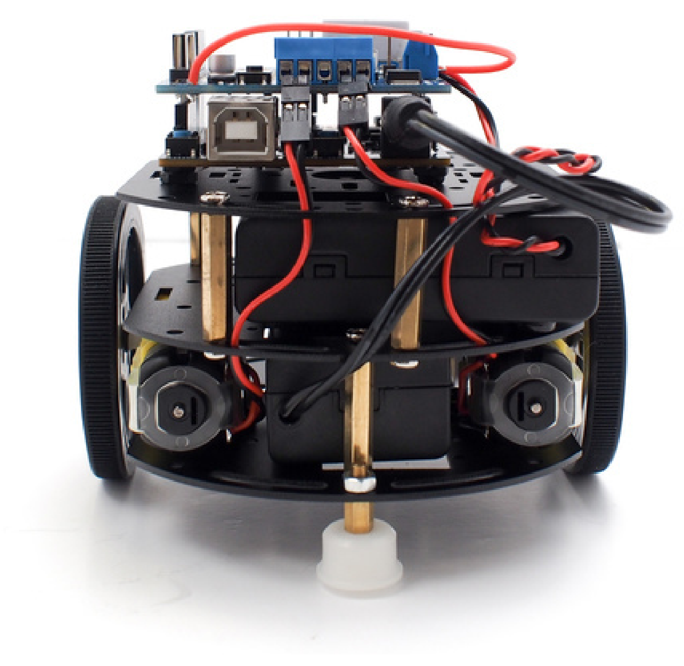
Created by John Park



Last updated on 2018-08-22 03:58:48 PM UTC

Guide Contents

Introduction

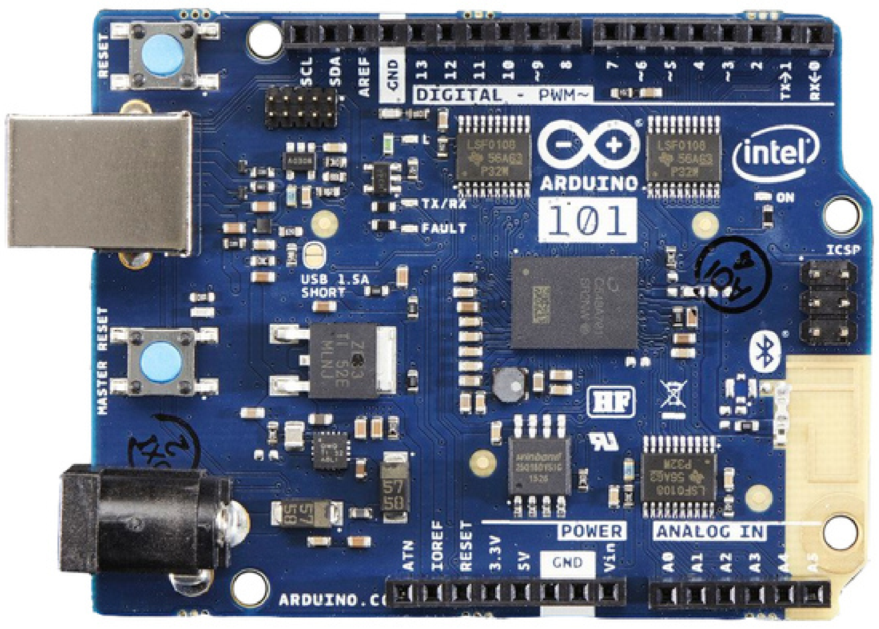


Greetings Human Friend!

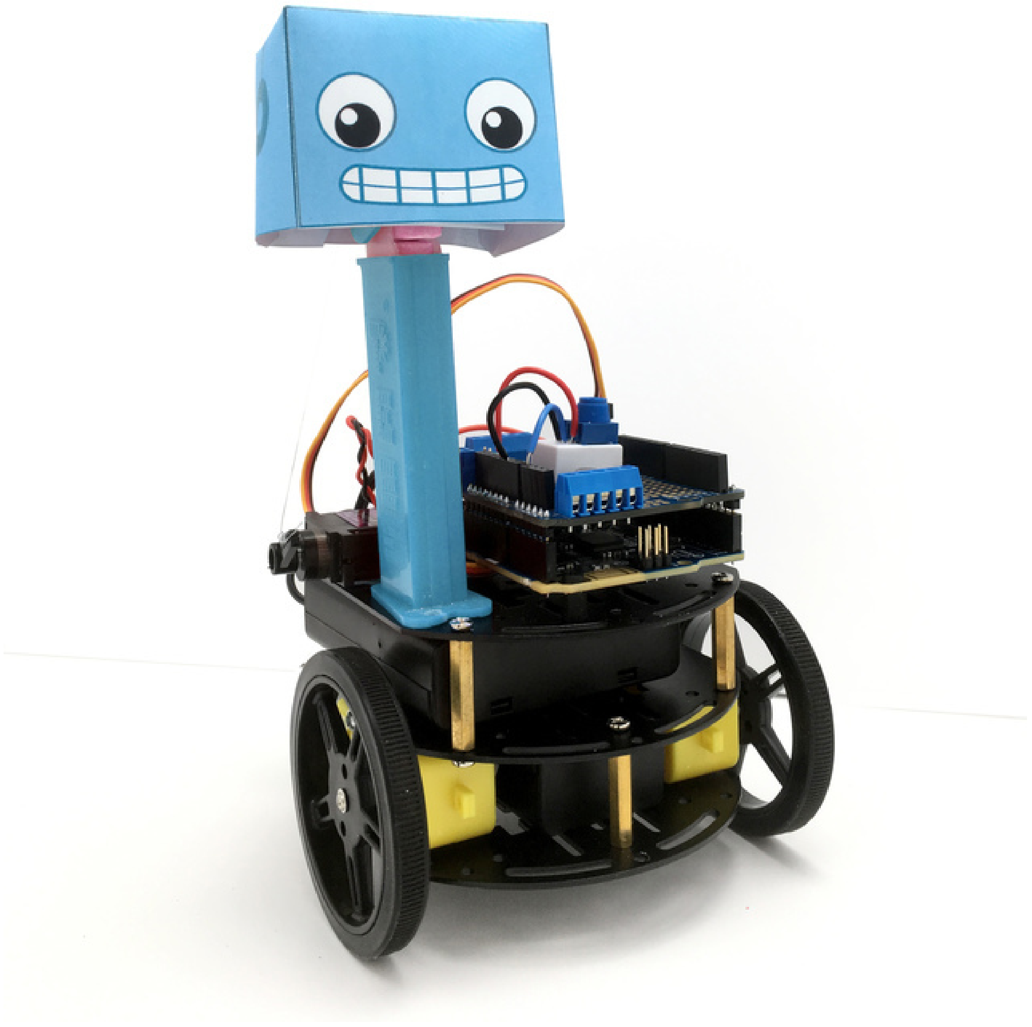
Welcome to **CurieBot**, your own mobile, customizable, and *adorable*, 32-bit, Bluetooth-enabled driving robot!

Based upon our popular three-layer round robot platform, CurieBot brings some special sauce to the table -- delicious **Arduino 101** sauce, that is! This powerful board comes in a familiar Arduino form factor, so you can use many of your favorite shields, such as the included **Adafruit MotorShield v2.**

Instead of the typical processor, this one is packing an Intel Curie 32-bit chip, which enables some very powerful on-board processing. The Arduino 101 also has a 6-axis accelerometer/gyro, and that's not all -- it's got Bluetooth LE built directly onto the board, which enables direct control of CurieBot right from your Bluetooth-capable phone or tablet!



Build and control your robot right away, and then begin exploring all of the possibilities for modifications and expansion with this powerful new platform.



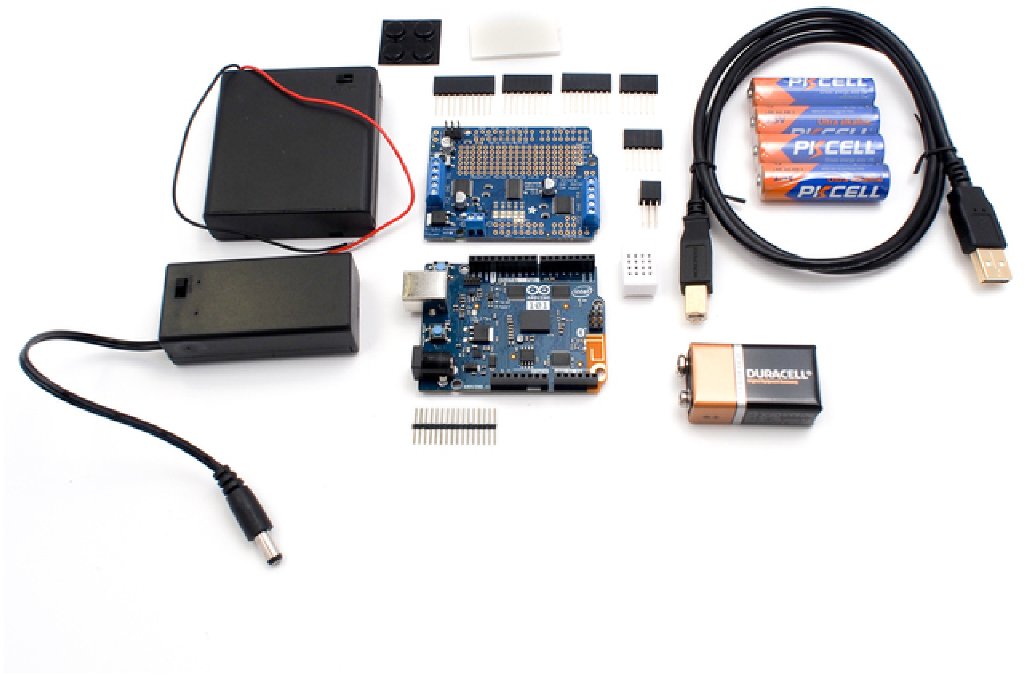
Unboxing CurieBot

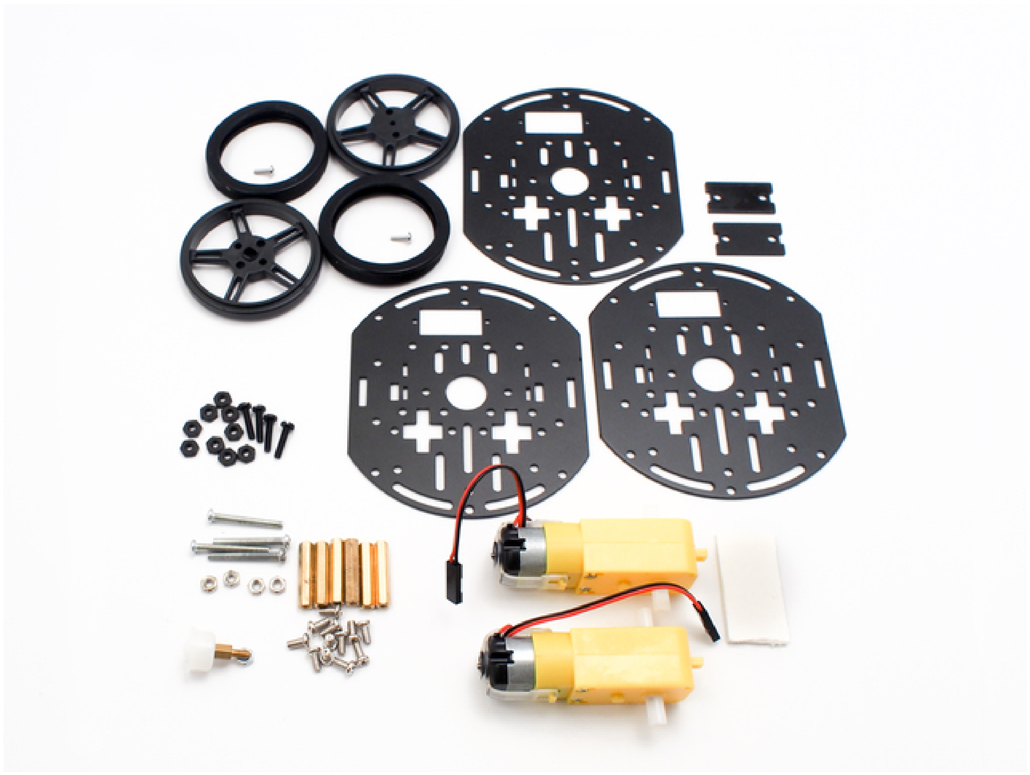
CurieBot is designed to introduce you to the joys of making with electronics. We decided to come up with a fun pack of parts that:

- Could introduce a beginner to making
- Teach soldering, 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:





Arduino 101, USB Cable, & Batteries

- 1x Arduino 101 with Intel Curie (http://adafru.it/3033) the brains of your bot! It's a board that combines the universal appeal of Arduino with the latest technologies - like the Intel Curie module (https://adafru.it/ueG), Bluetooth LE capabilities, and a 6-axis accelerometer/gyro.

- 1x USB Cable - Standard A-B - 3 ft/1m (http://adafru.it/62) - use this to install new code onto your Arduino 101 (from any computer)
- 1x 9V battery (http://adafru.it/1321) - this battery will power your Arduino 101 (but not the motors)
- 4x AA Batteries (http://adafru.it/3349) - Use these to power your the motors of your super awesome little robot

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 Motor/Stepper/Servo Shield v2 (http://adafru.it/1438) - Lets you drive up to 4 DC or 2 stepper motors- certainly enough motors to power the chassis kit!

Prototyping Parts and Components

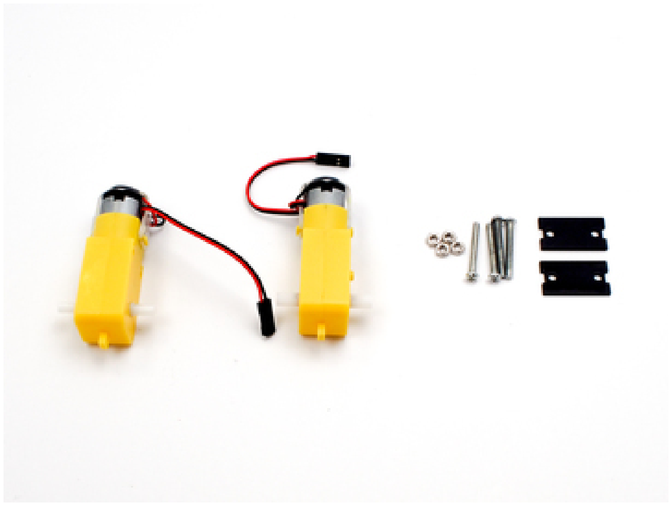
- 1x - Mini Solderless Breadboard - 4x4 points (http://adafru.it/2463) - Perfect fit for your chassis kit. Add it to the prototyping area to upgrade your robot with more sensors, lights, servos and beyond!
- 1x 9V Battery Holder with switch & plug (http://adafru.it/67) - Plugs into your Arduino 101's DC jack to power the CurieBot on and off with a flick of the switch
- 1x 4xAA Battery Holder w/ On/Off Switch (https://adafru.it/sfq) - A nice portable battery holder for your robot's motor batteries.
- 1x Shield stacking headers for Arduino (http://adafru.it/85) - Solder these to your MotorShield to allow access to all pins for further upgrades later!
- 1x Rubber Bumper Feet (https://adafru.it/dLG) - Helps keep the battery packs safe and secure
- 1x Foam tape rectangle - Helps mount the battery packs
- 4x Nylon 2.5mm pan screws - For mounting your Arduino 101 to the top deck of the robot (http://adafru.it/3299)
- 9x Nylon hexnuts - Also for Arduino 101 fastening (http://adafru.it/3299)

Assembling and Wiring Your Robot

The wiring and assembly is pretty easy, and there is very little soldering required. You'll need a soldering iron and solder, diagonal cutters, a small screwdriver, and it wouldn't hurt to grab some pliers. You will also need a 9V battery, 4 x AA batteries and a USB cable to upload code to the Arduino 101 board.

First, you'll assemble the robot chassis. All the parts needed for this are inside the brown box with the 'Custom Black 2WD Robot + extra layer' sticker on it.

Motors and Wheels and Tires

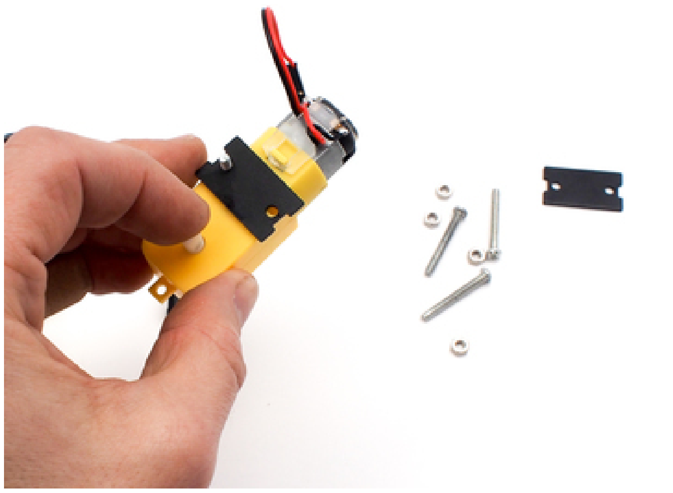


To start, take the two motors, four long screws, four nuts, and two black panels.

Screw the two black panels onto the motors.

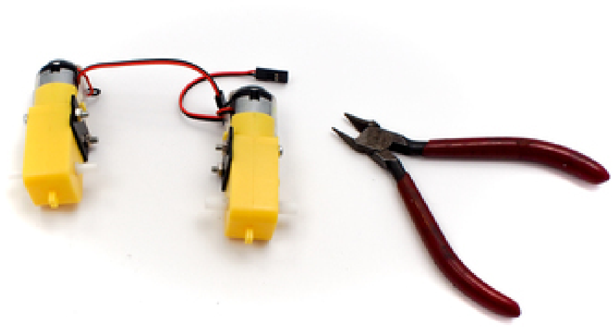
The metal panels go on the side with the red and black wires coming out.

Have the hex nuts on the metal panel side so they don't interfere with the wheel!

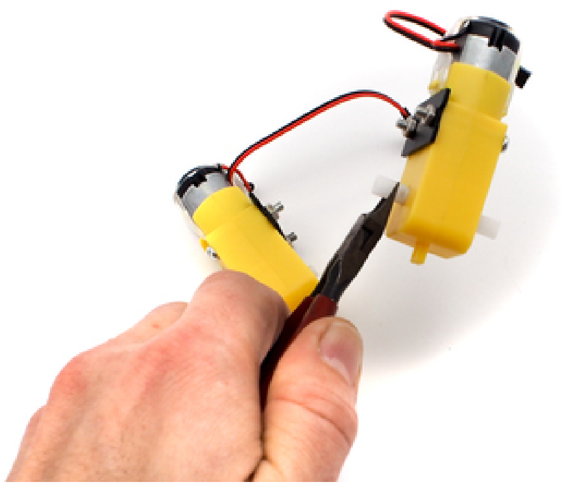


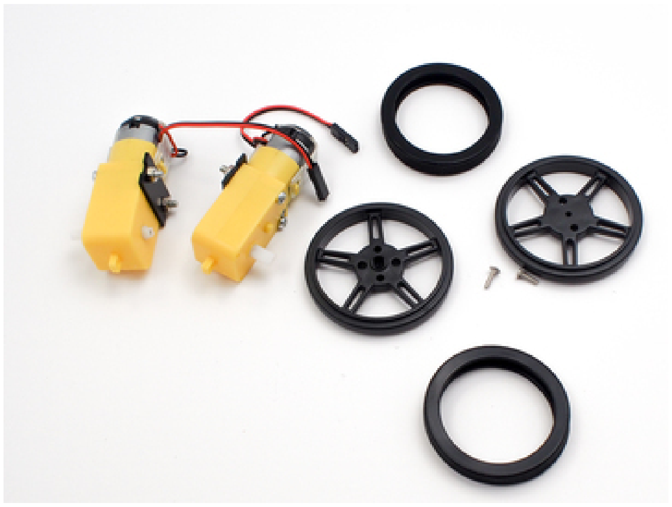
The metal panels go on the side with the red and black wires coming out.

Have the hex nuts on the metal panel side so they don't interfere with the wheel!



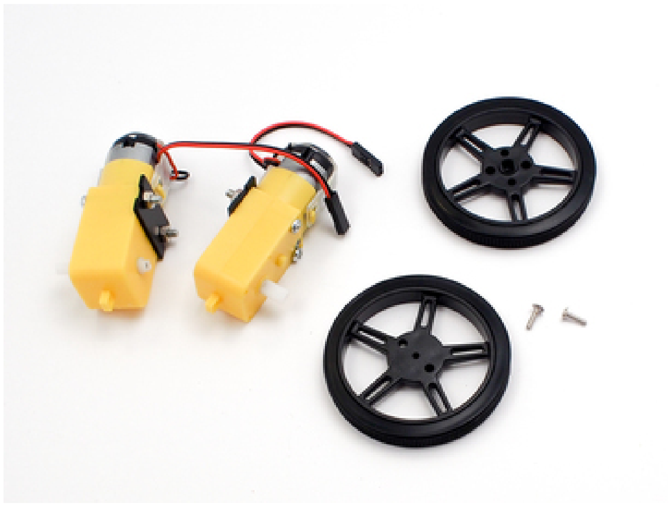
To make space for the 9V battery pack later, trim the excess off of the inner motor wheel shaft on each motor. Grab some diagonal cutters (scissors or a hobby knife will work in a pinch) and trim off about half the length of the motor wheel shaft that is on the same side as the black plate you affixed.

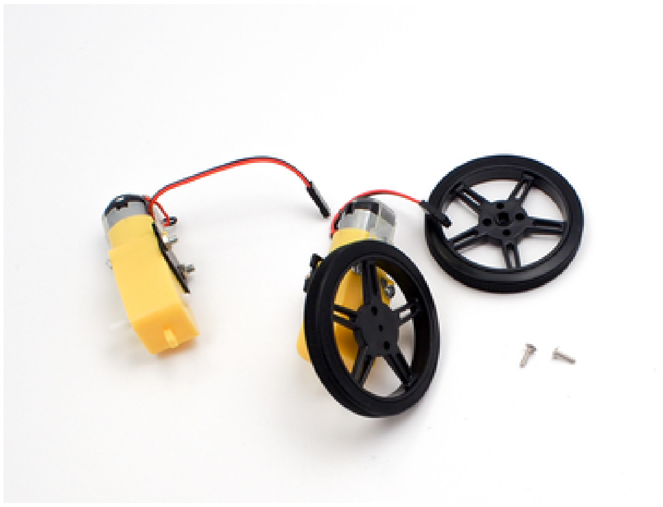




Take the two wheels, rubber treads, and 2x small screws found in the same bag as the wheels.

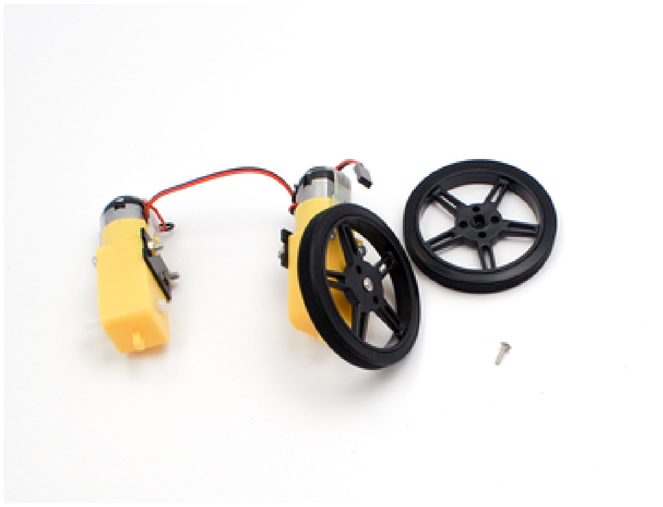
Put the rubber treads on the wheels. This is a lot of fun!

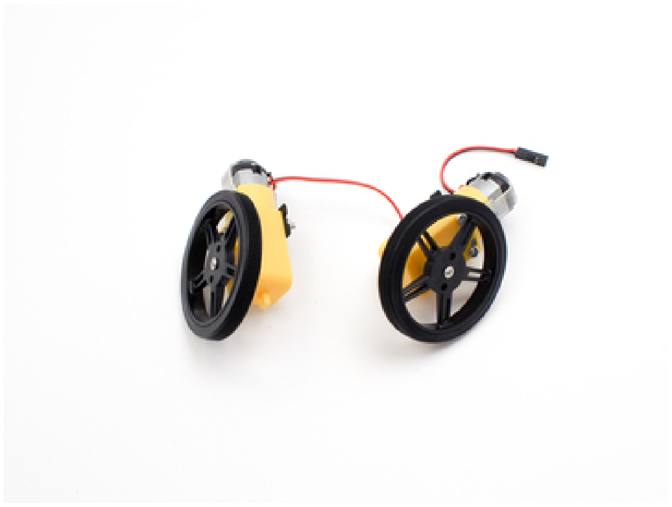




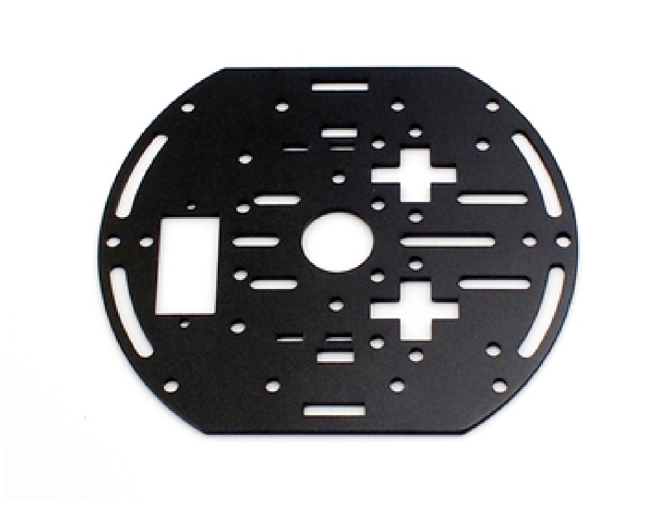
Fit the wheels onto the white knob on the motors, they will snap nicely onto the oval center.

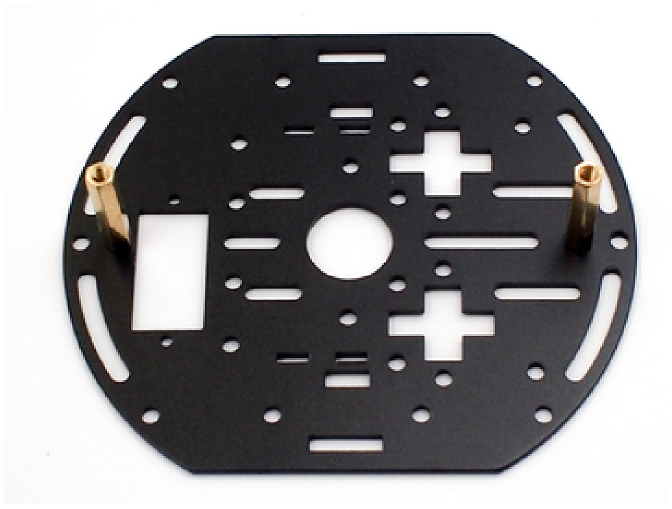
Attach the wheels into place with the tiny screws





Lower Chassis



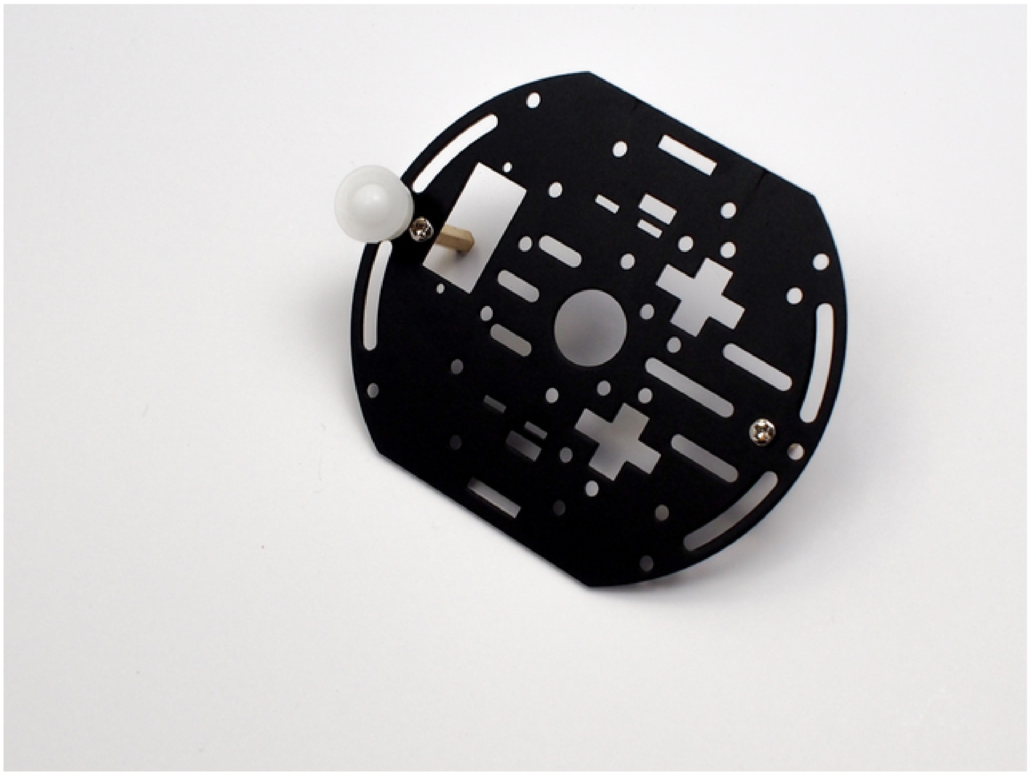


Take one of the black chassis layers. All three layers are identical.

Align it on your table as shown on the left. Note that the panel **is not symmetrical** - look on the left to see that rectangle cut out? Make sure it's aligned as you see here!

Attach two of the brass standoffs onto the black chassis layer.

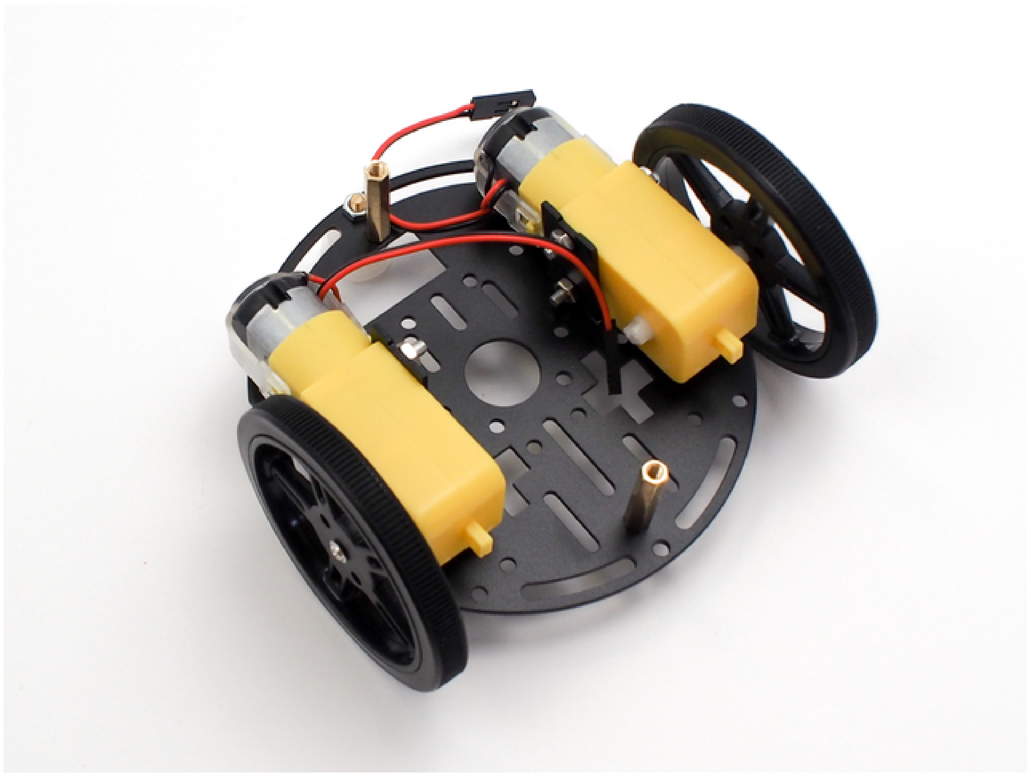
The standoffs should be screwed into the second set of holes from the outer edge - meaning the two interior holes.



Turn over the plate

Attach the white free-wheel into the exterior hole closest to the rectangular opening.

The white free-wheel should be on the opposite side of the chassis of the standoff.



Turn over the plate again

Take your assembled wheels and fit them into the chassis layer.

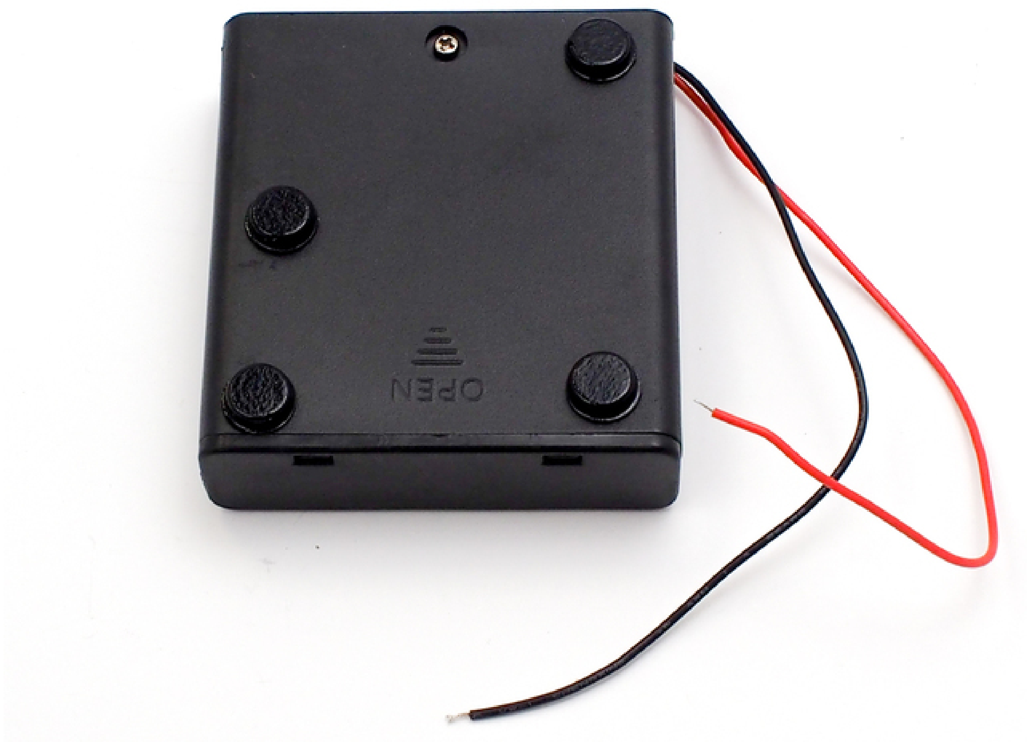
There are 2 slots on the black panels that you attached to your motor that should fit perfectly into the chassis layer.

The metal front of the motor will be pointing toward the side of the chassis where you placed your white freewheel

Prepare the Battery Boxes

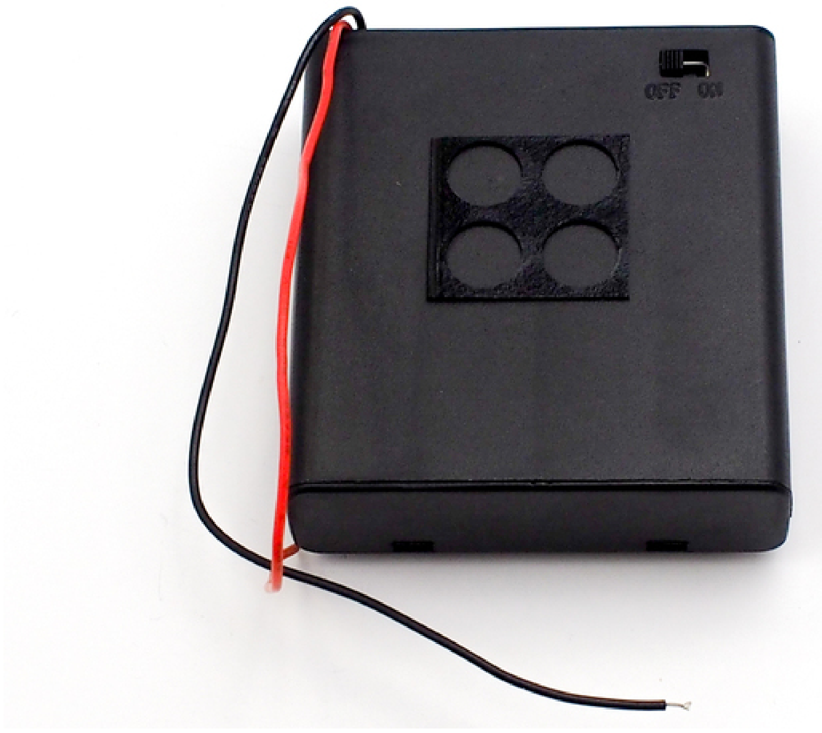
For this step, you will need the AA battery box, 4 x AA batteries, the 9V battery box, a 9V battery the screwdriver, a sheet of 4 rubber bumpers, and double-stick foam tape.

First, open each battery box, grab out the screw, insert your batteries, and then screw the boxes shut. **Oh, and make sure you have the boxes 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.

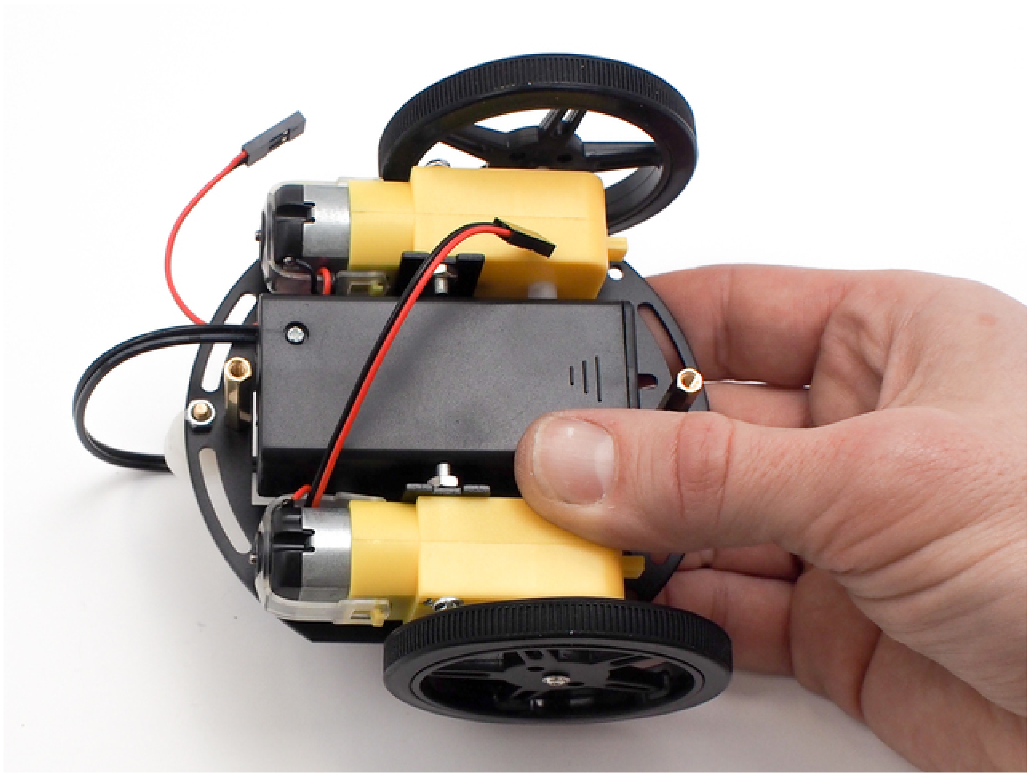


Do not discard the leftover piece of bumper material.

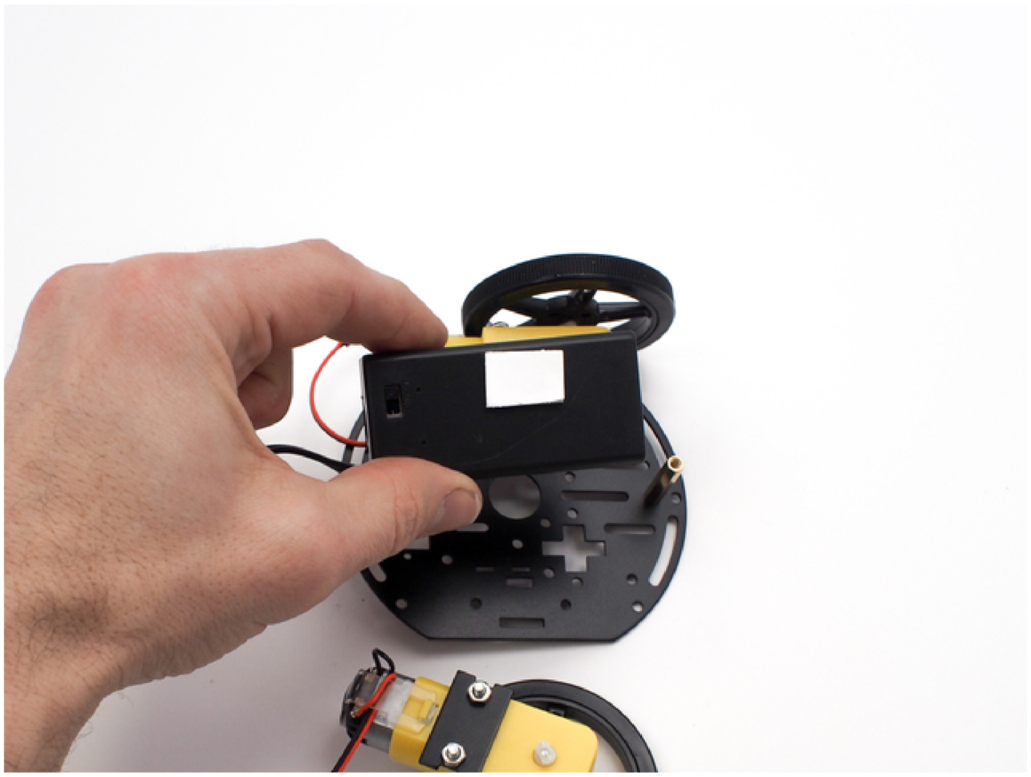
Flip the battery box over and place the scrap piece of the bumper material in the middle. This will help hold the battery box nice and tight between the top and middle plate of your robot.



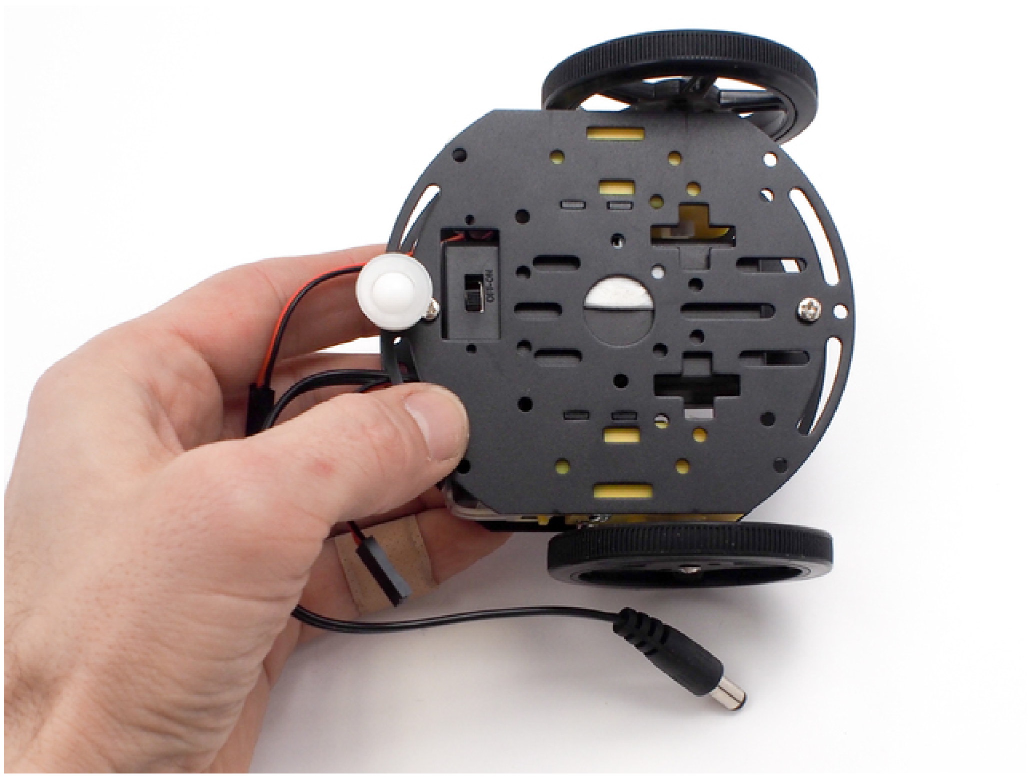
Add the 9V Battery Box to the Chassis



Check to see that the 9V box fits between the motors -- trim more of the inner shafts if needed.

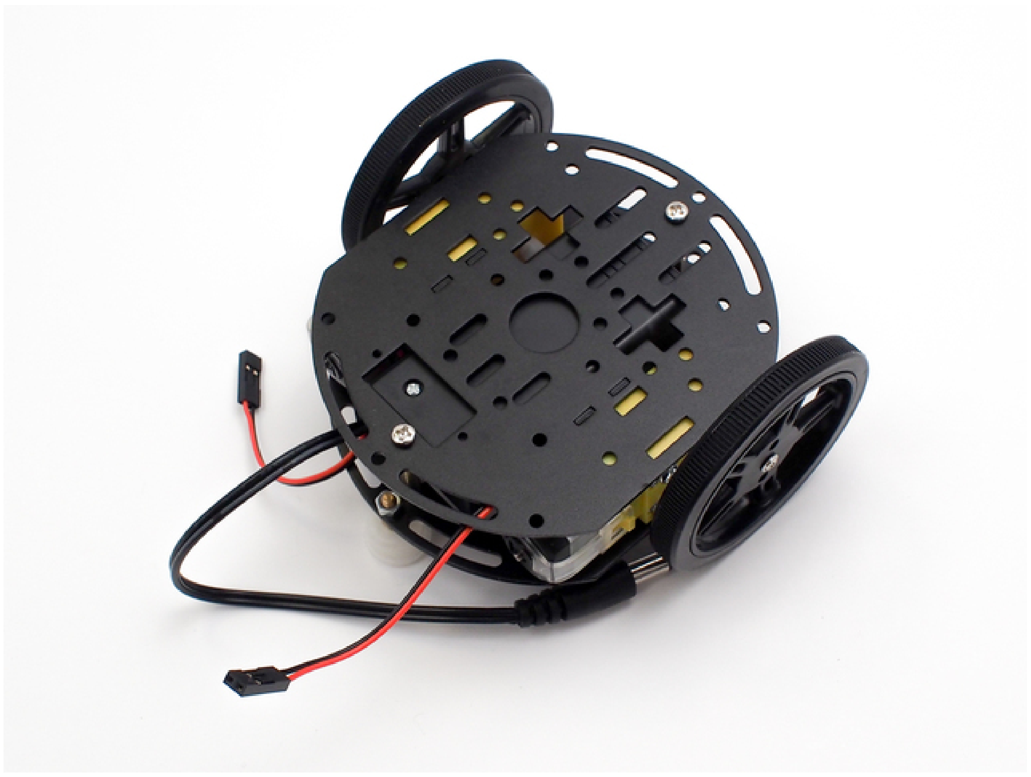


Cut a small strip of double stick tape and press it onto the side of the battery box with the 9V switch.

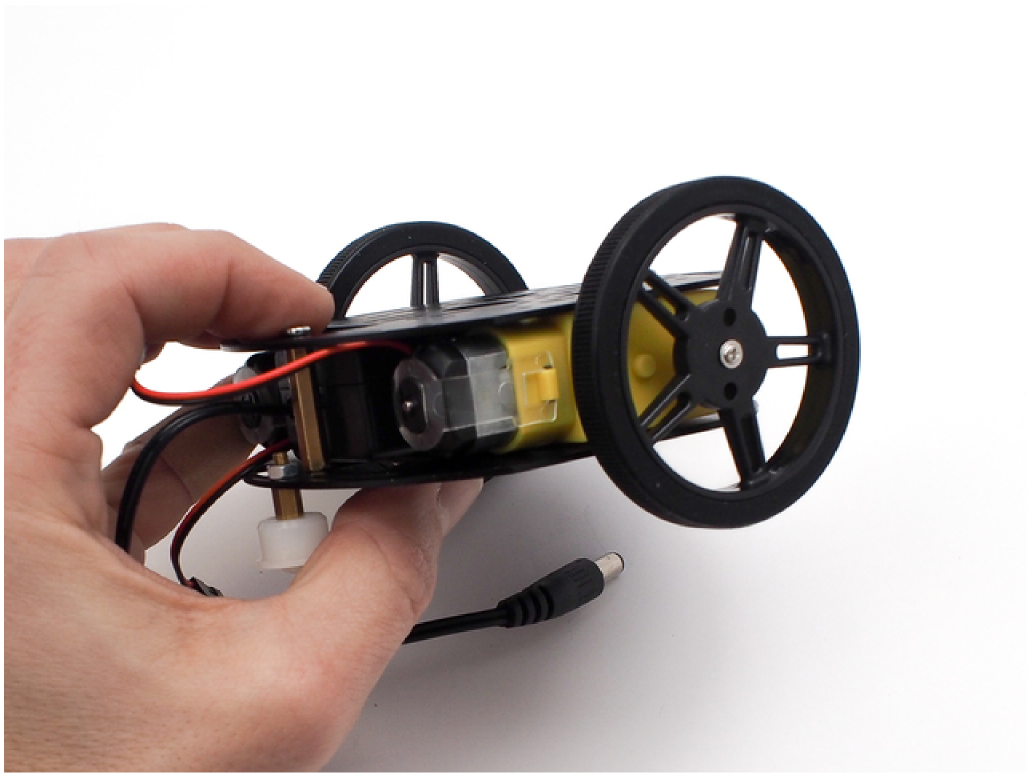


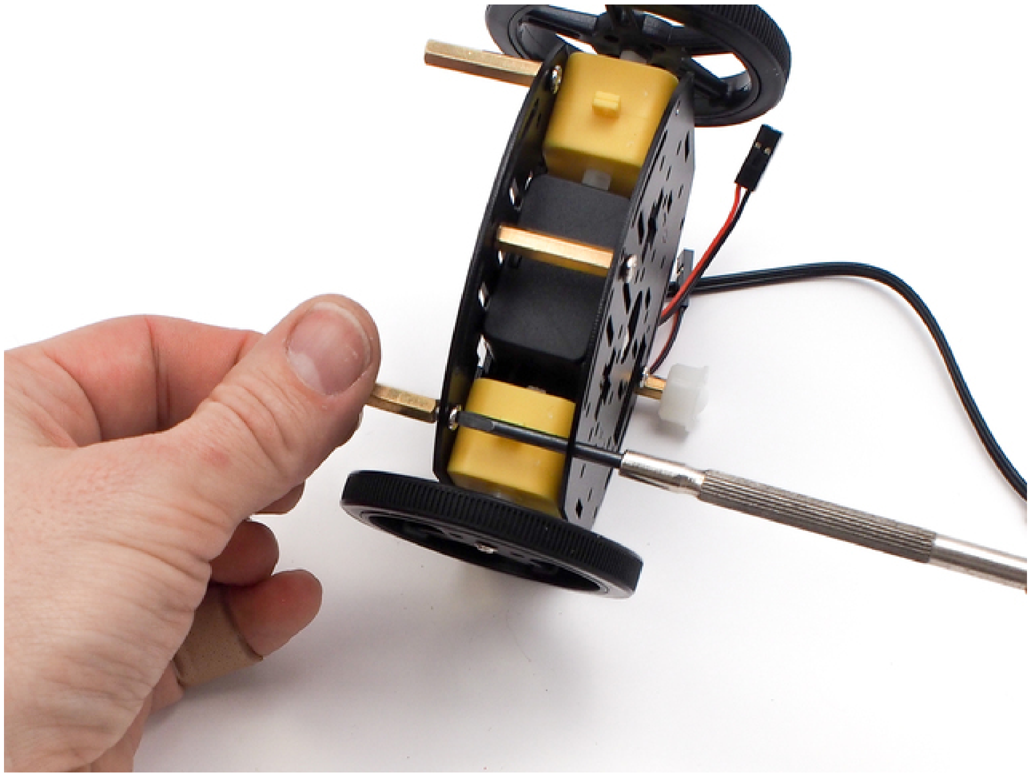
Place the battery box back in between the motors and stick it to the chassis

Middle Chassis Layer

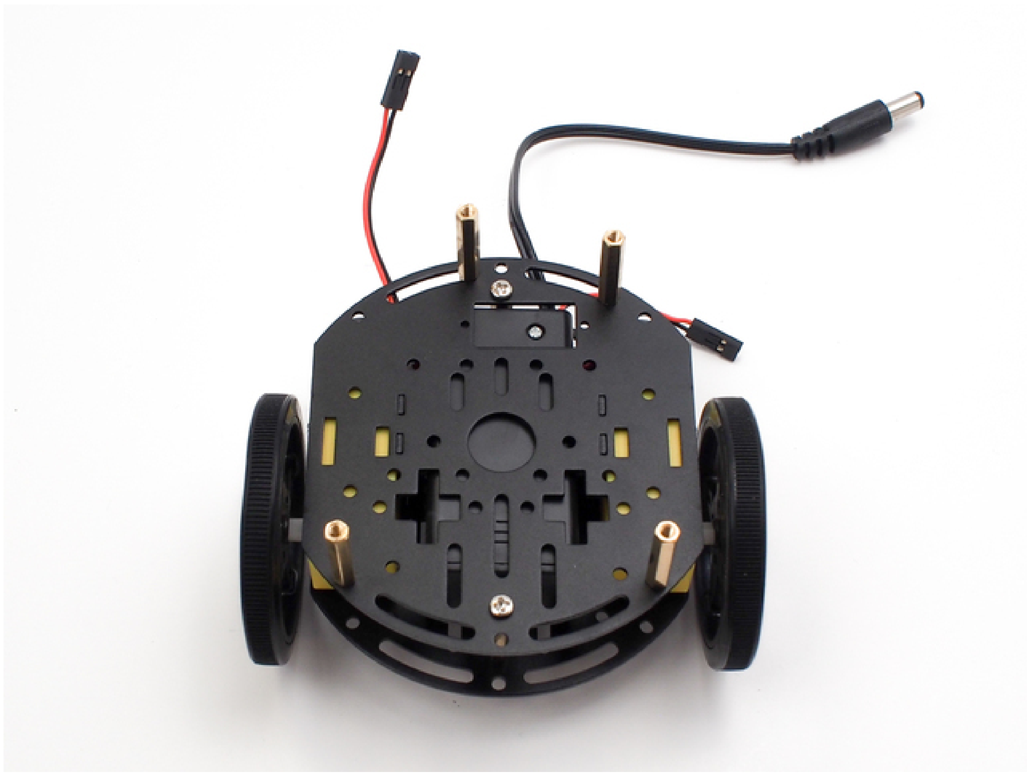


Place the middle chassis layer onto robot, making sure to fit the motor tabs into the slots of the layer, then screw in the two brass standoffs.

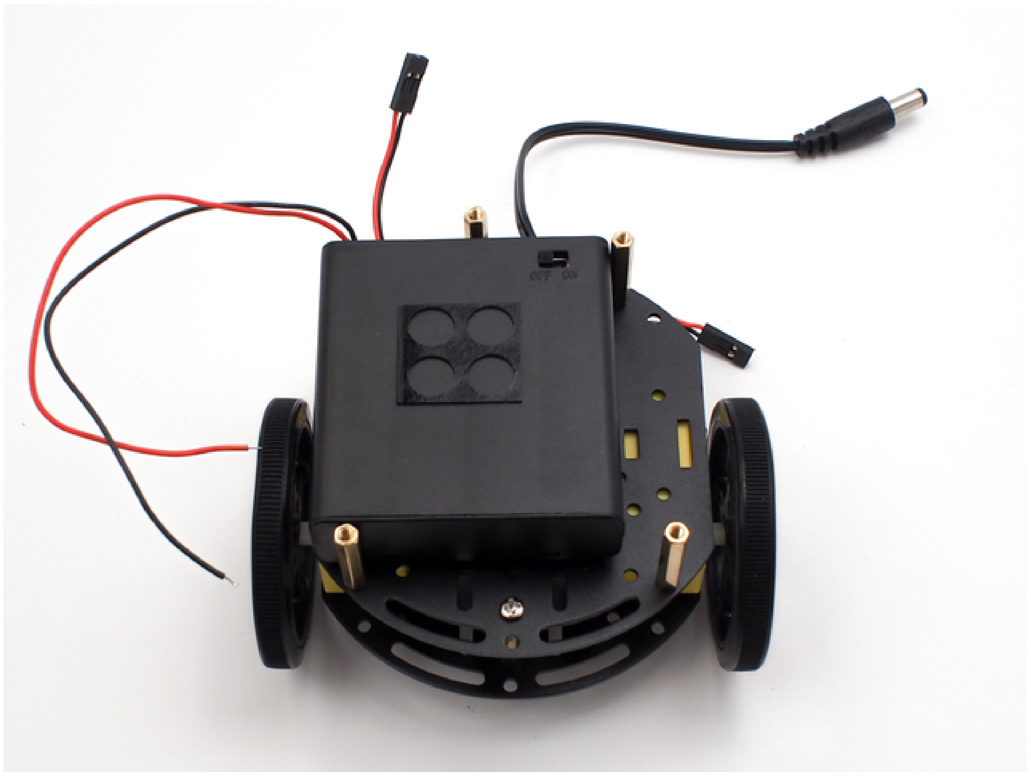




Take a look at the image below and install the brass stand-offs in the same positions. You can insert the stand-off screws through the middle plate and hand tighten the stand-offs while putting a bit of pressure on the screw with your finger. Or, as a tip, you can screw in the stand-off screws with the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



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.



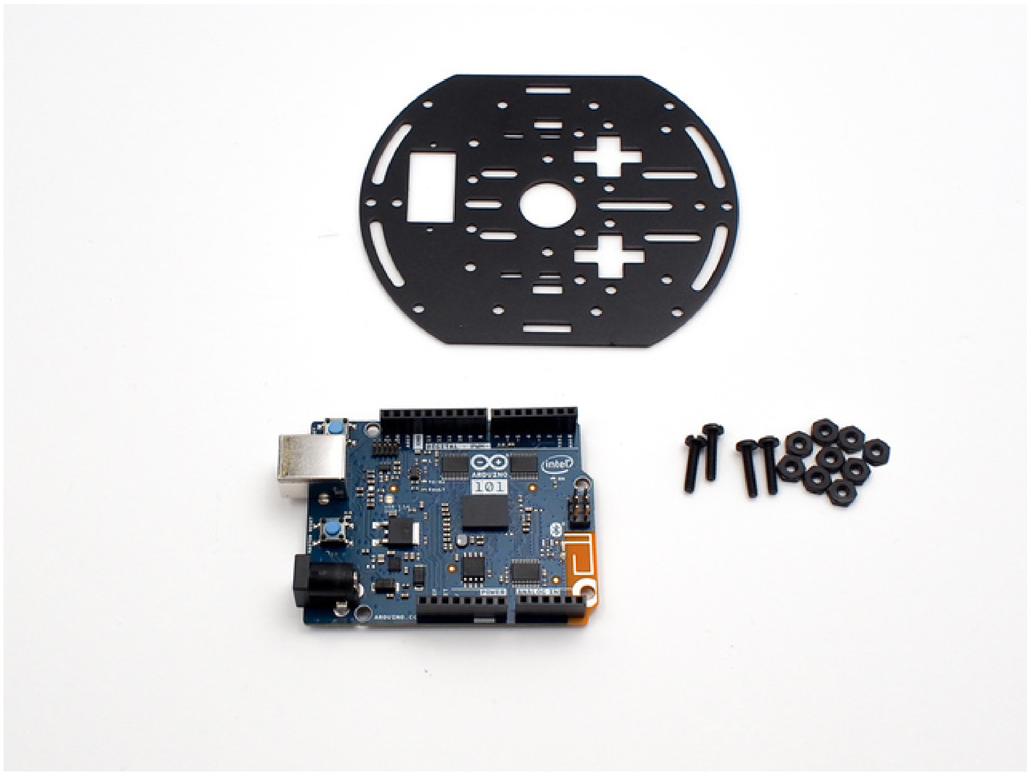
Notice the battery box is lined up on the left side of the middle chassis plate. It should be just in-between the upper left stand-off and the lower left stand-off (not touching either). This will make sure the on-off switch lines up just right in the hole of the top plate.

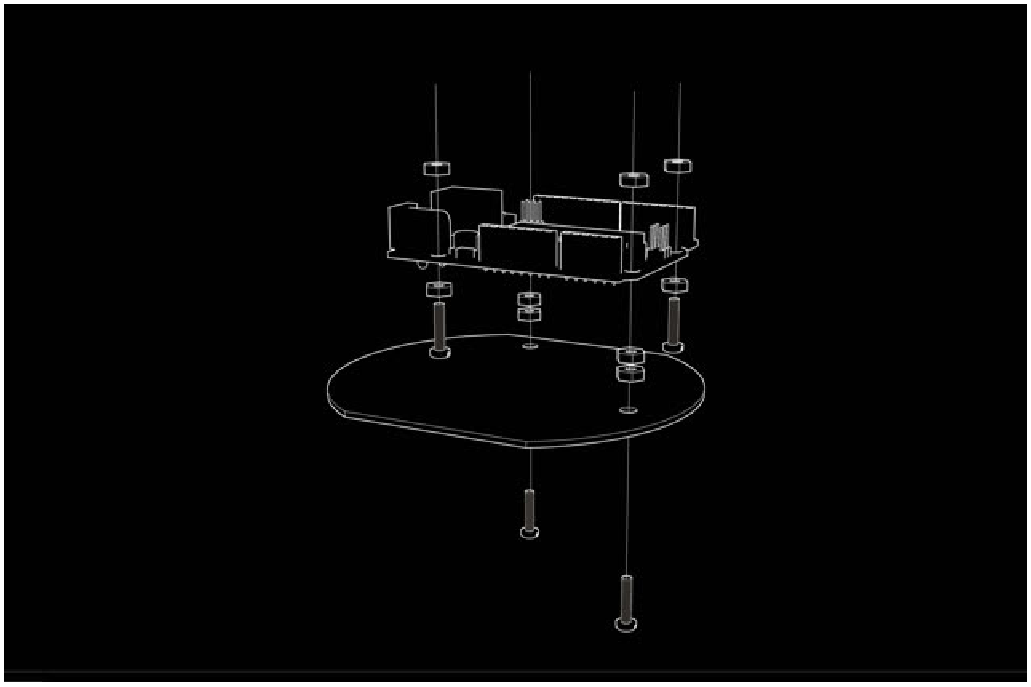
Mount the Arduino 101

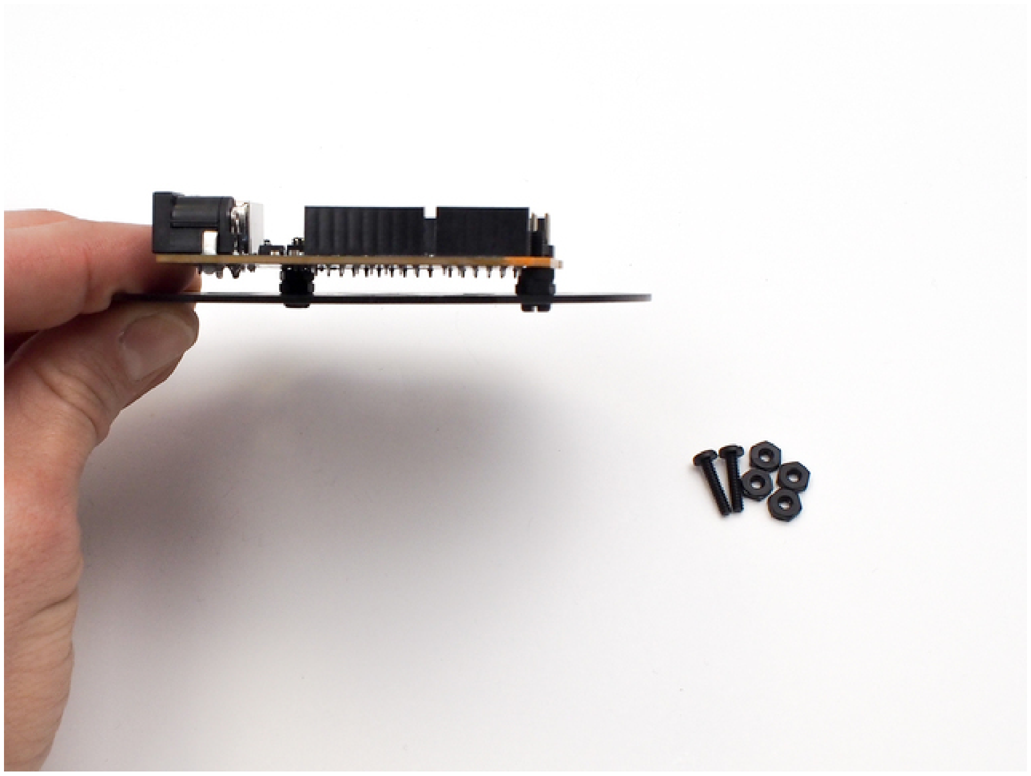
Next you mount the Arduino 101 to the top chassis plate. Before doing so, note the orientation of the plate, the end with the "+" signs is the same as the other two plates, but the large rectangle cutout is flipped to opposite side compared to the bottom layer. This is to accommodate the AA battery box switch.

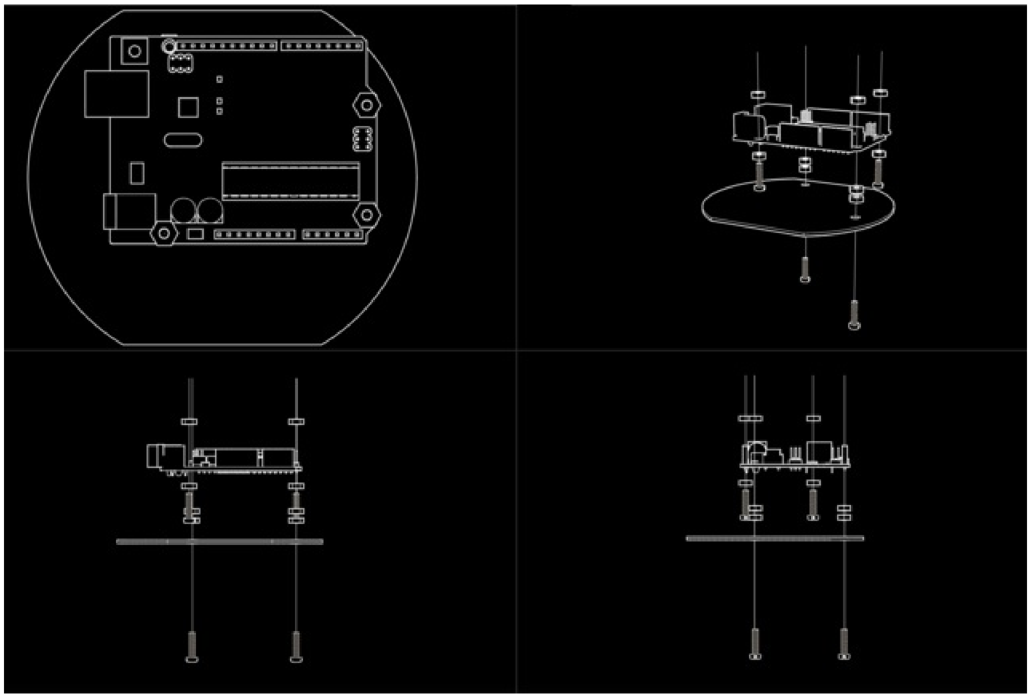
Use the black nylon screws and nuts to mount the Arduino 101 board to the top plate. Two of the Arduino mounting holes can be lined up with two of the plate holes and screwed in. You'll mount the other two screws on the Arduino only to use as "feet" to keep it from touching the board.

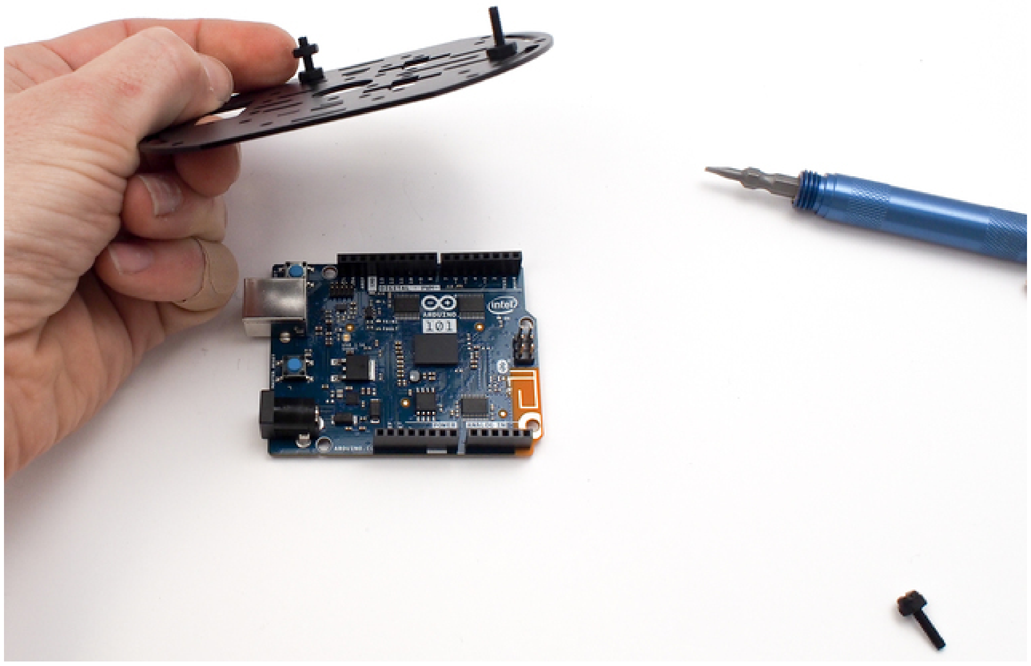
Follow the animated assembly diagram for the specific sequence of screws and nuts to use on each hole.

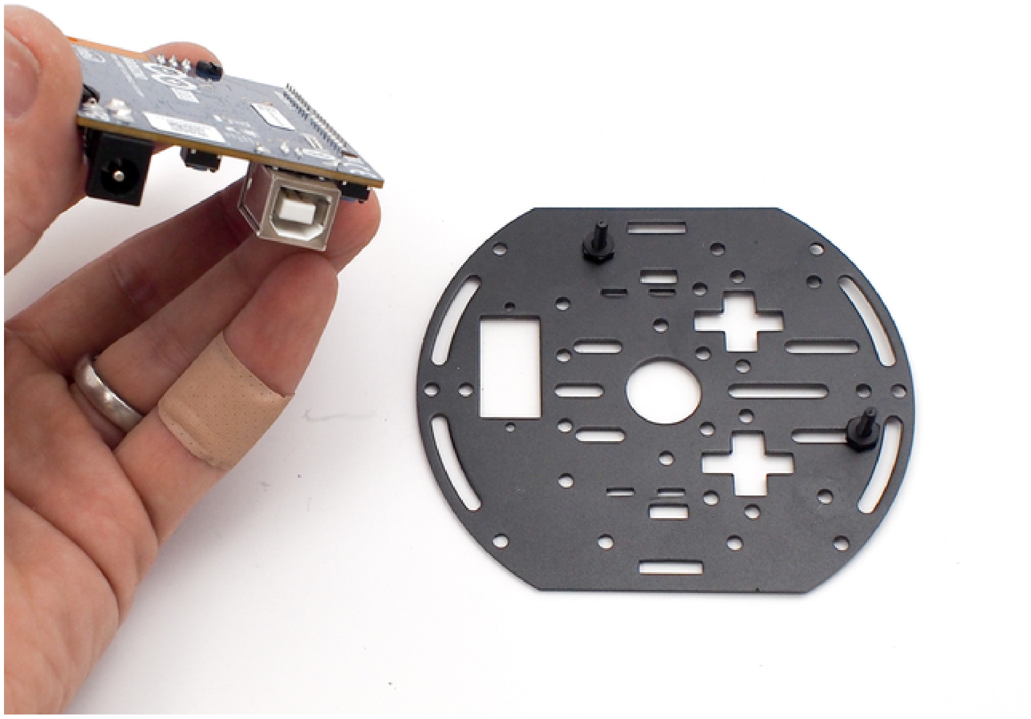


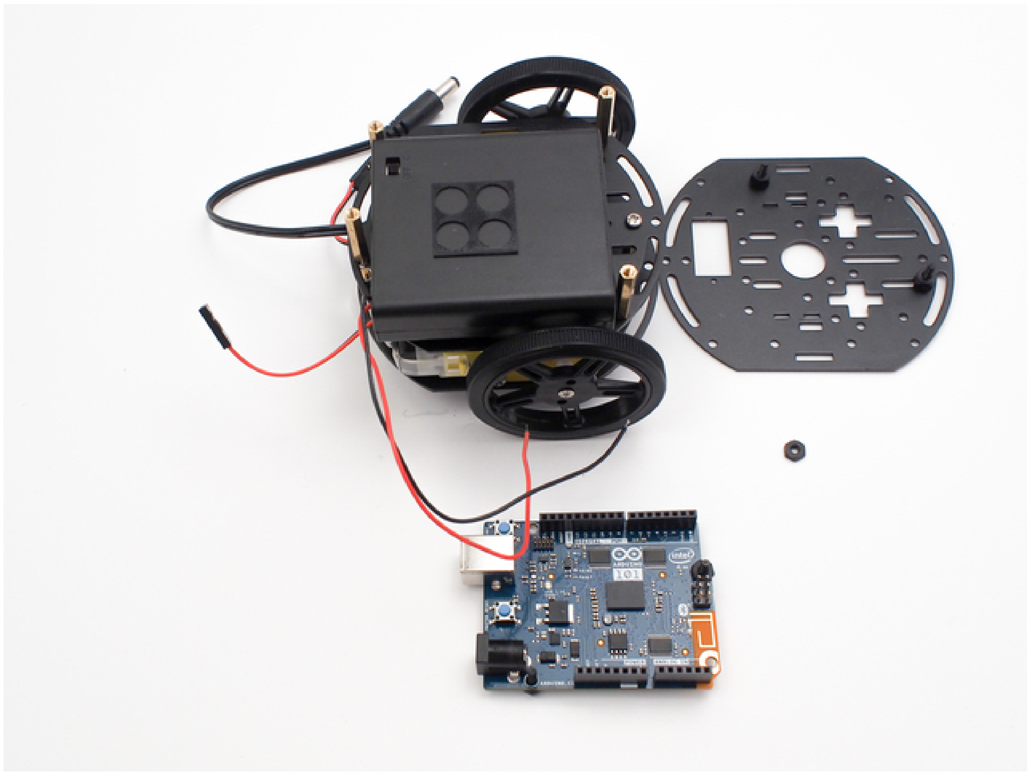




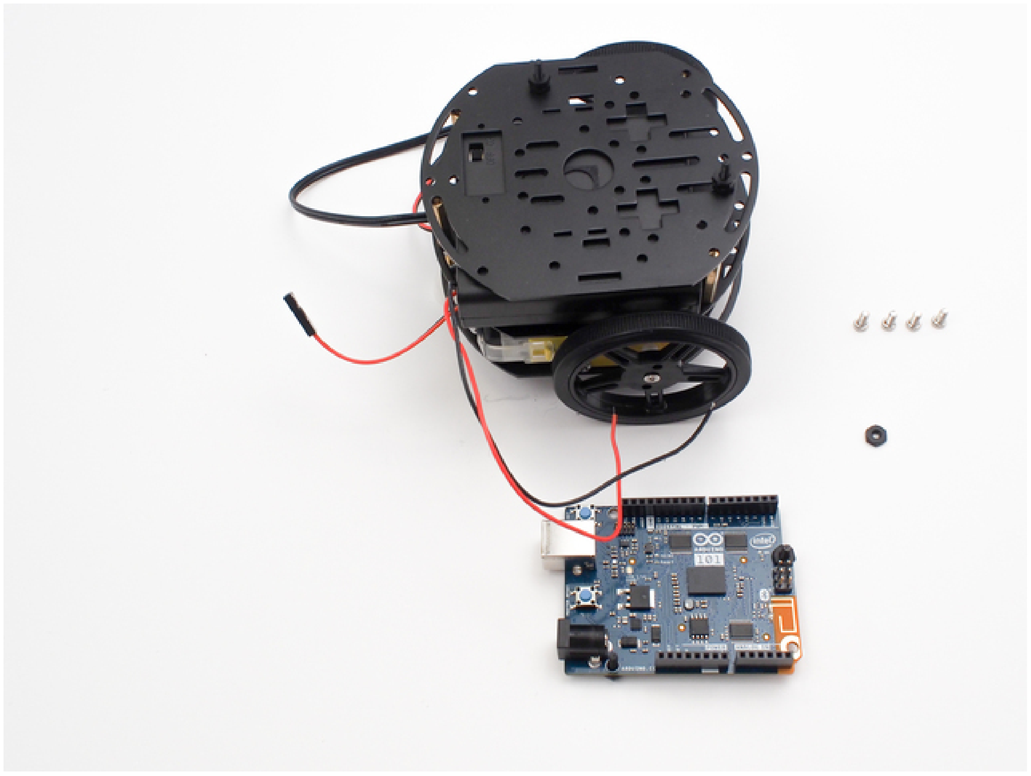






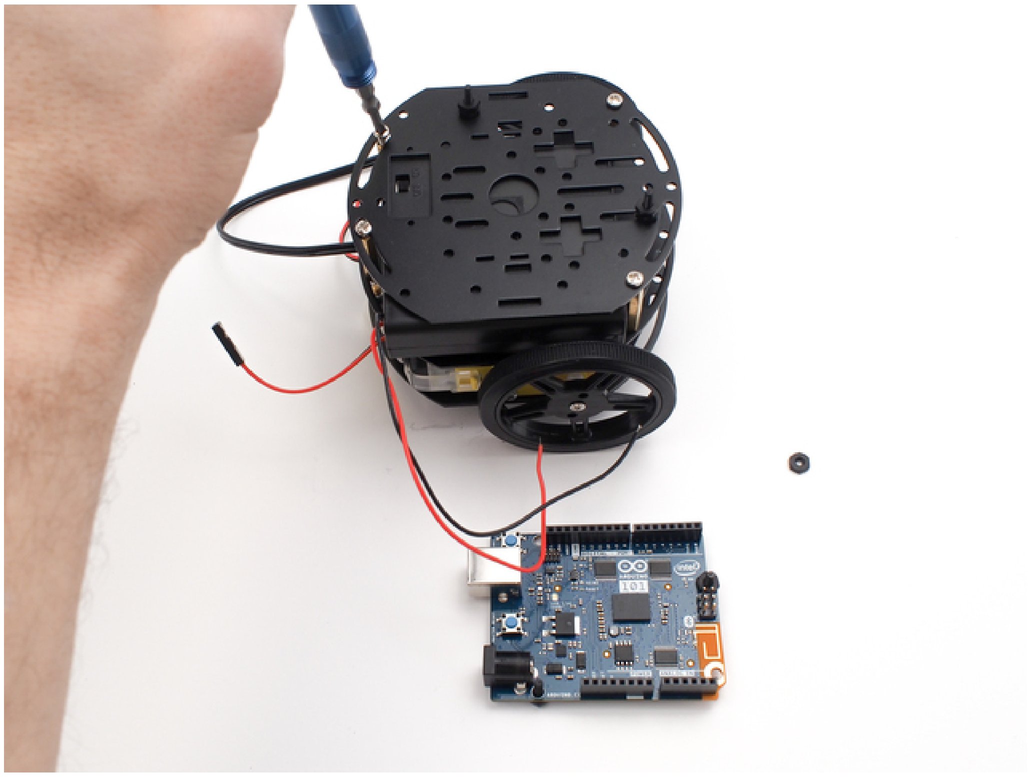


Screw in the two screws and four nuts that'll protrude from the top of the middle chassis layer as seen below, then place the layer on top of the AA battery box.

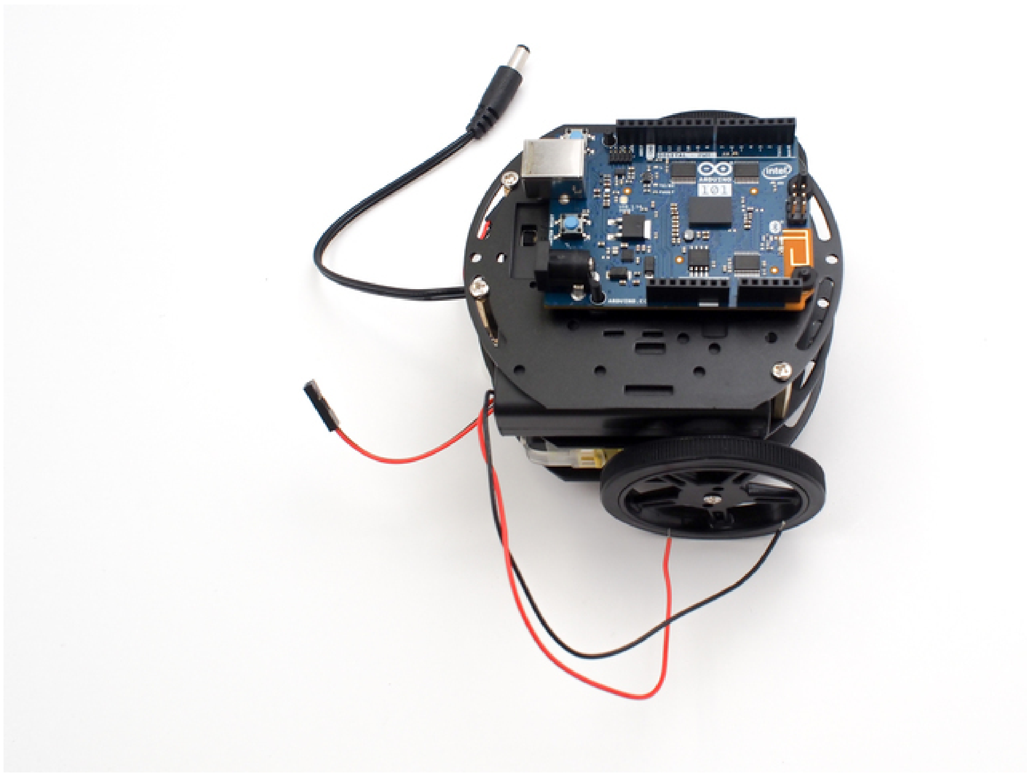


Screw the four brass standoffs into place with the small metal screws.

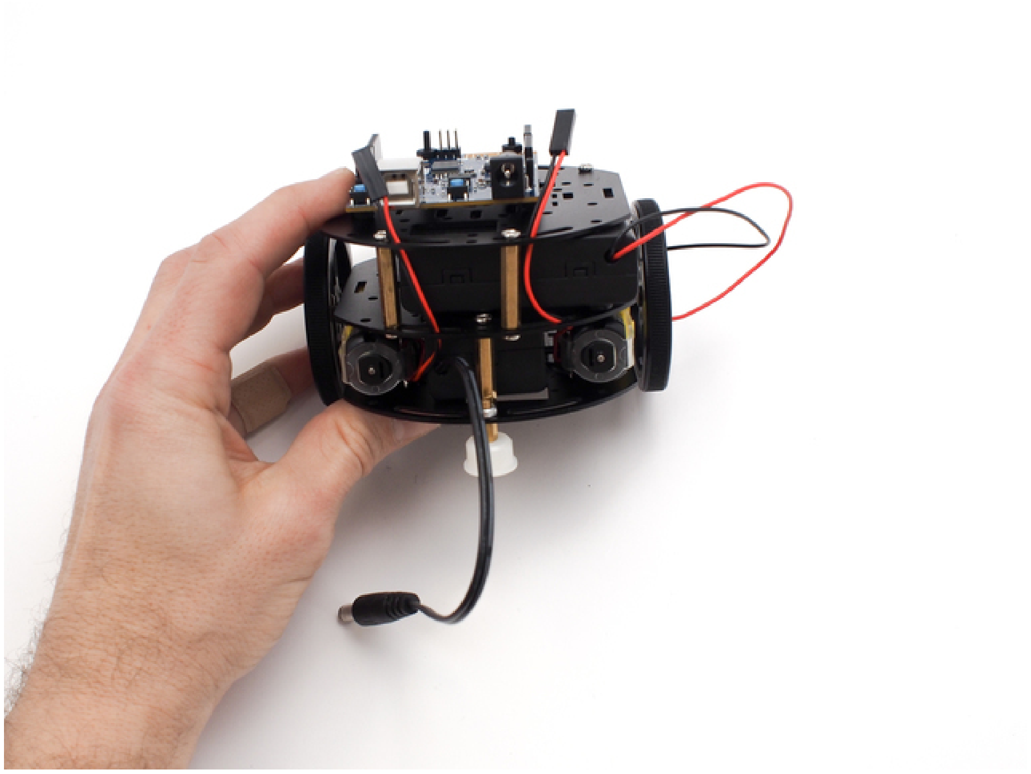
Be sure to screw the two remaining black nylon screws into the Arduino 101 as seen here, placing a one nut below and one nut above the board in each case as seen in the animated diagram.



Place the Arduino 101 onto the top chassis layer, lining up the screws, then place the remaining nut onto the lower right screw to hold it in place. The upper left screw can't accommodate a nut due to the header row, but the screw will prevent the board from rotating.



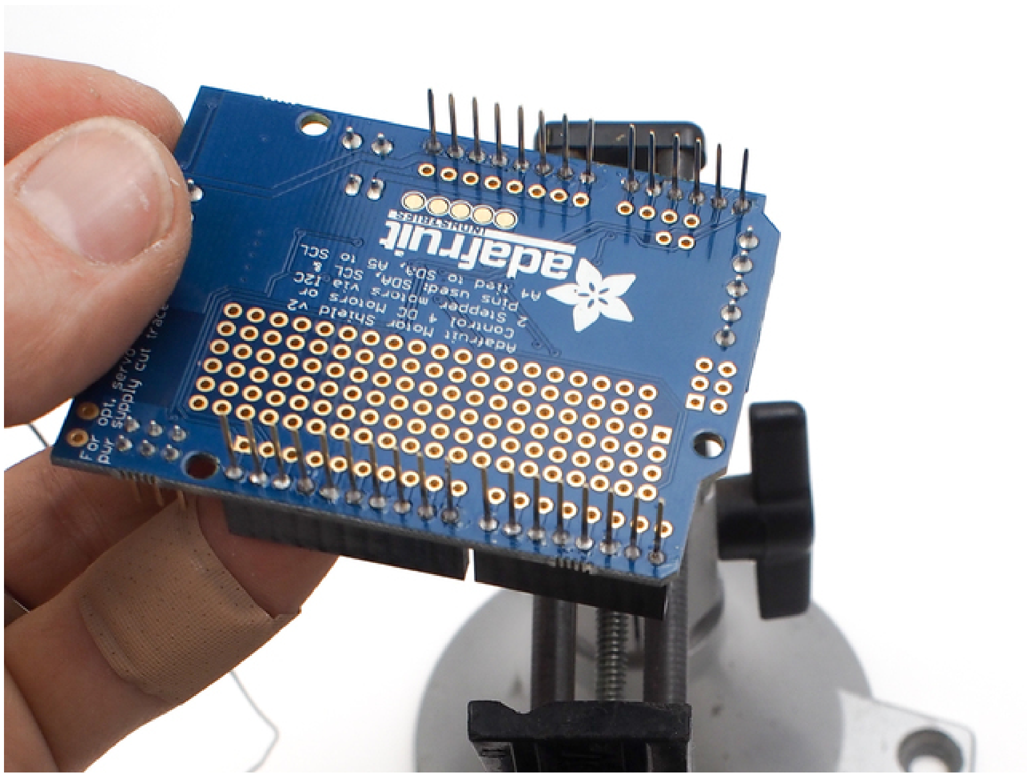
Route the wires up through the chassis layers for connecting them later to the MotorShield.



Solder the MotorShield Headers

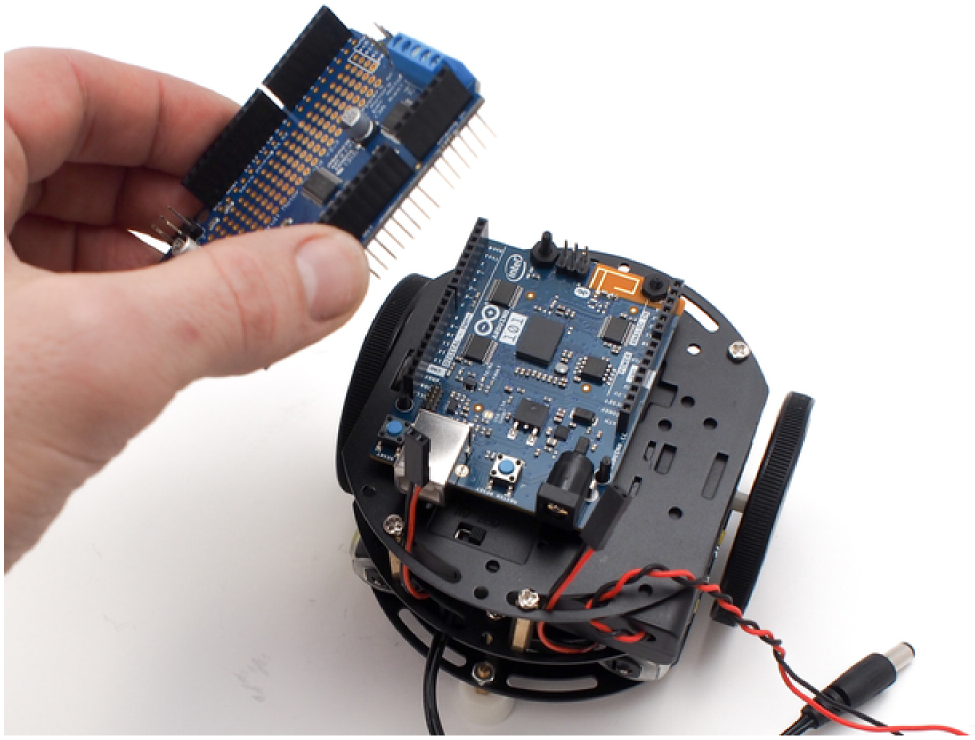
Now it's time to solder the stacking headers into the Motor Shield. These will be used to connect the MotorShield to the Arduino 101, while still allowing access to all pins for connecting other sensor and components later.

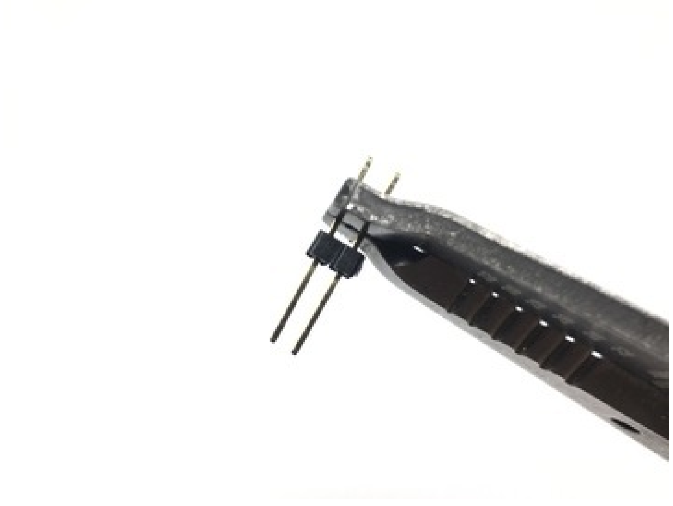
Solder the headers as seen below. You can follow this guide (https://adafru.it/ueH) for more details.


adafruit
Adafruit Motor Shield v2
Control 4 DC Motors or 2 Stepper motors via I2C
Pins used: SDA, SCL &
A4 tied to SDA, A5 to SCL
opt. servo supply cut trace
For pwr

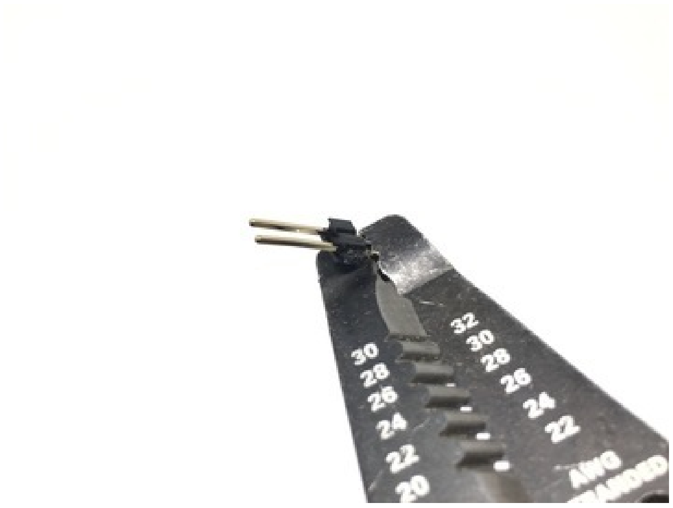
Once the headers are soldered in place, place the shield onto the Arduino 101.

You will NOT use the power jumper that came with the MotorShield, so do not connect it.

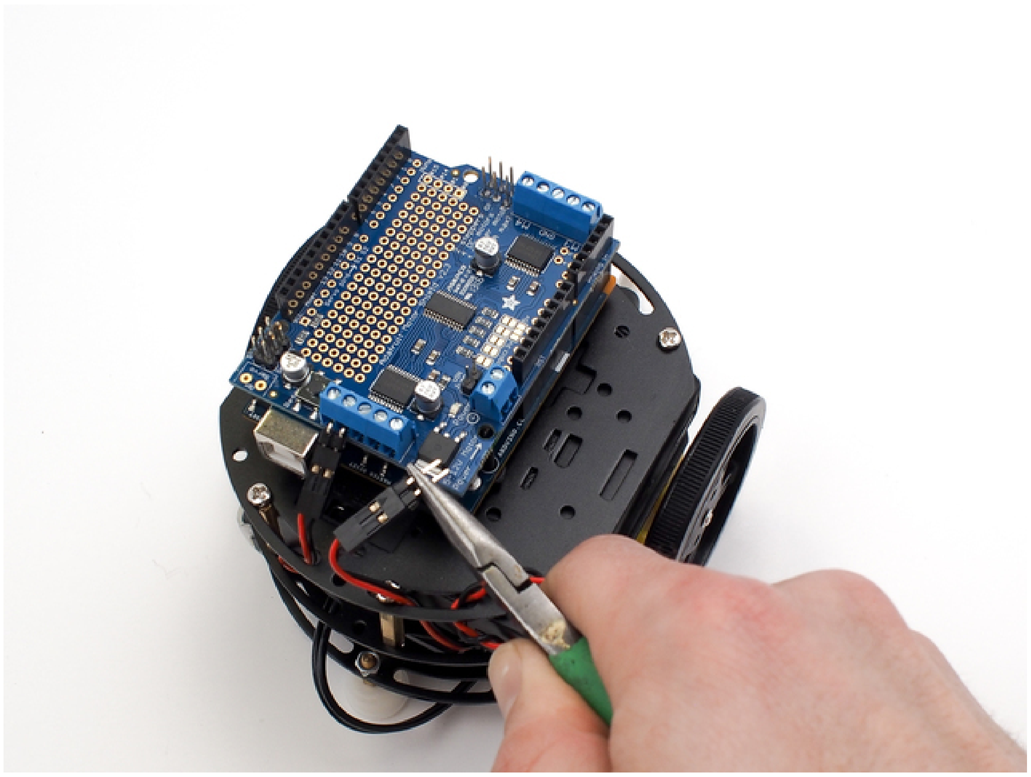


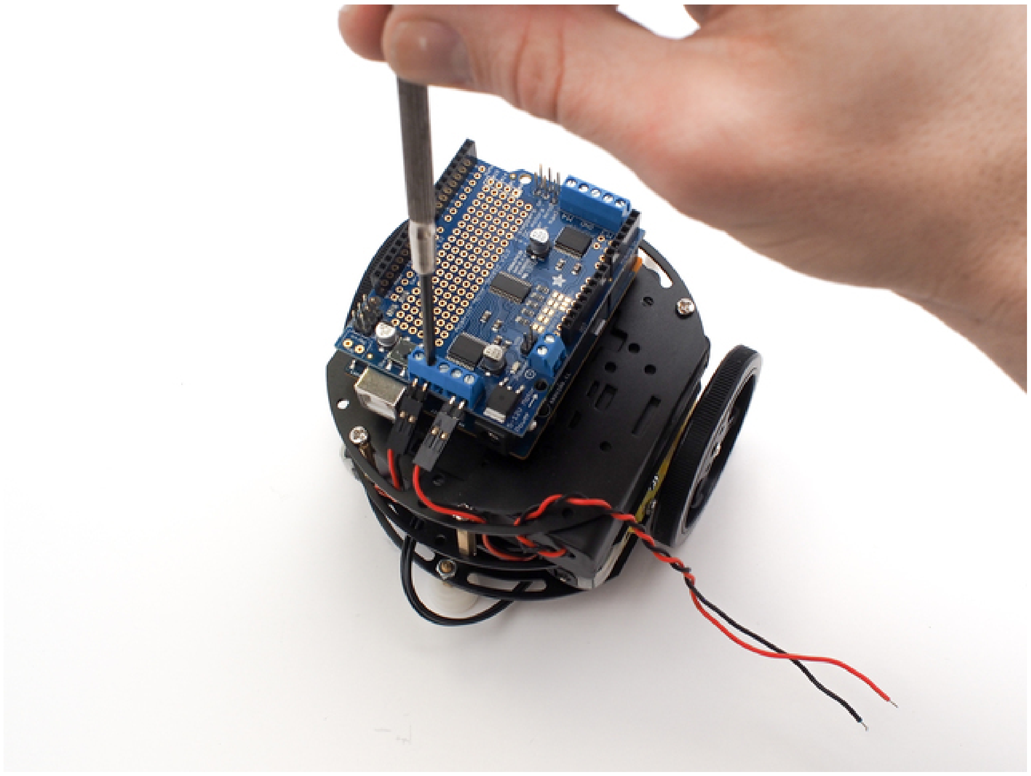


For this next step, you will need a set of pliers. First, grab the long header pins that are attached together and break them into 2 sets of 2 header pins. Then, grab them in the pliers like shown here, and then slightly bend them so they look like the next picture.



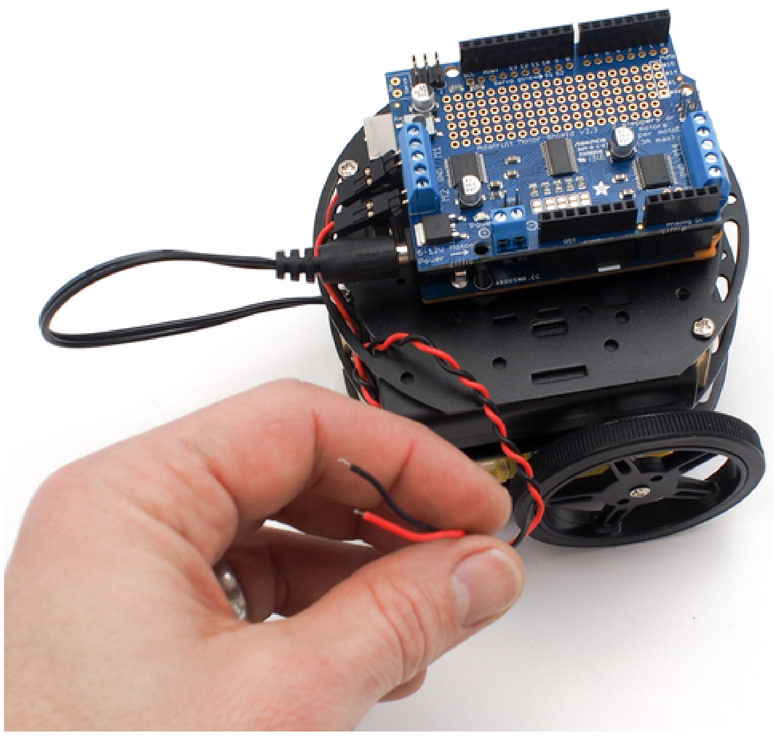
Insert the two sets of bent headers into the female crimp cables from the motors, and then into the MotorShield terminals M1 & M2, then tighten the screws down on them.

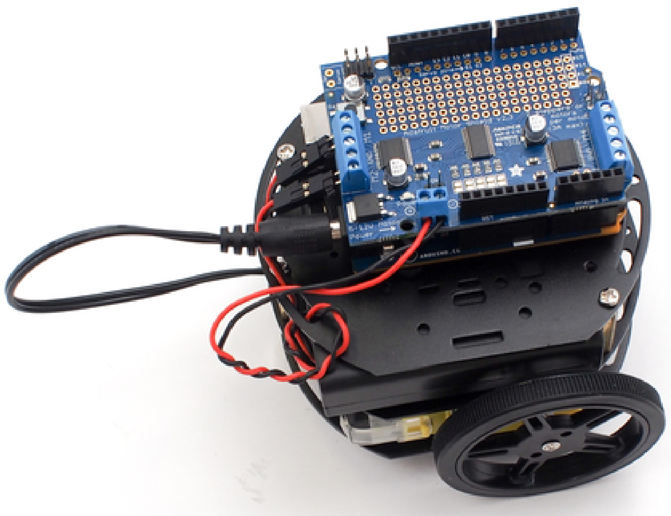


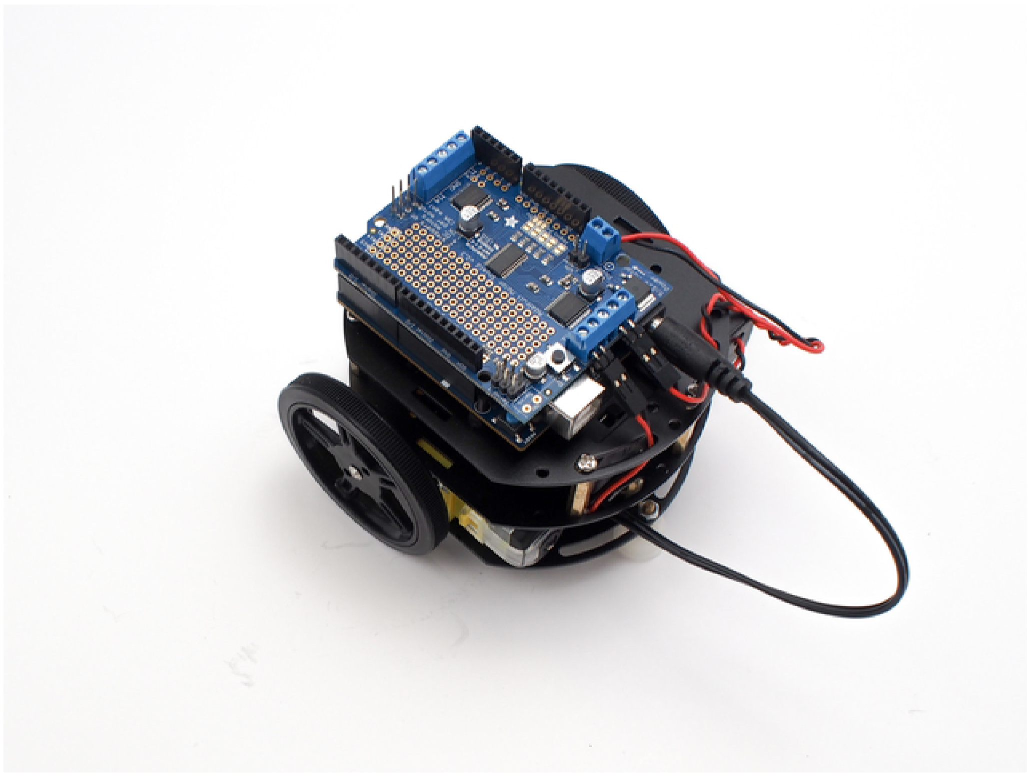


Make sure the battery packs are both turned off, then plug the 9V barrel plug into the jack on the Arduino 101.

Then, insert the red wire from the AA battery pack into the + power terminal on the MotorShield, and the black wire into the GND power terminal on the MotorShield, screwing them both down securely.





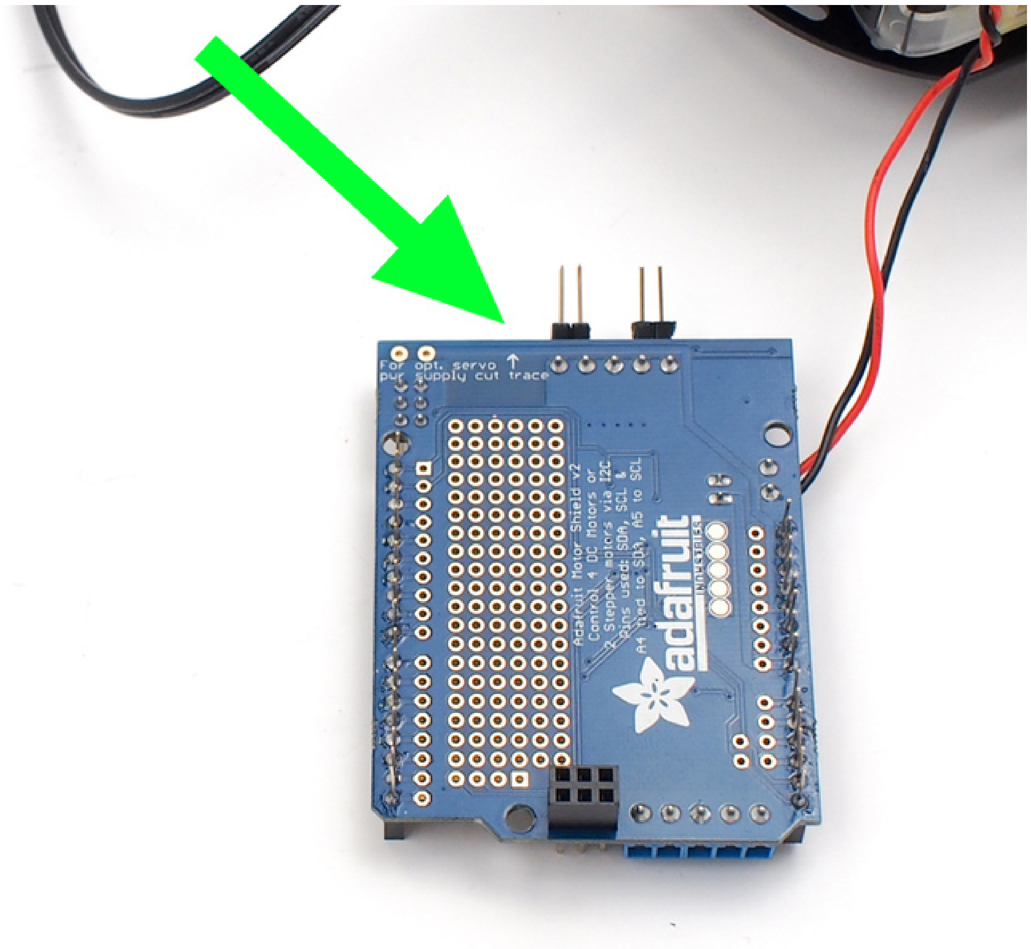


It is fine to skip this next step for normal use. However, if you think you'll be adding medium to large hobby servos to your CurieBot at some point, you can prepare the Motor Shield for it now.

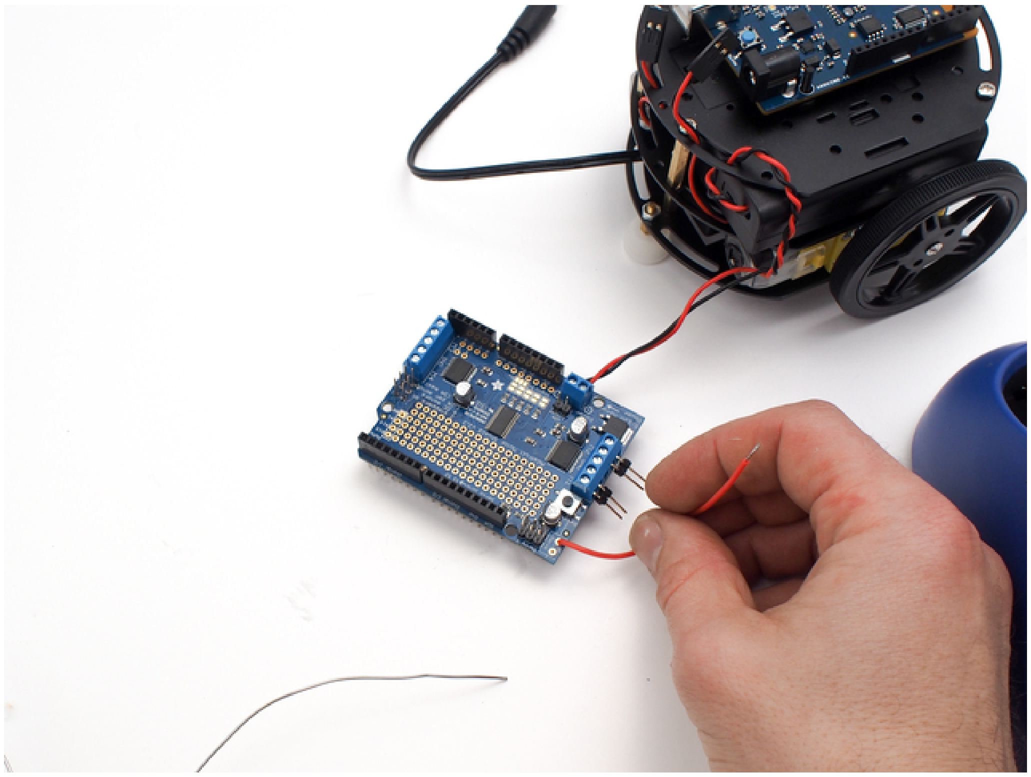
Servo Power

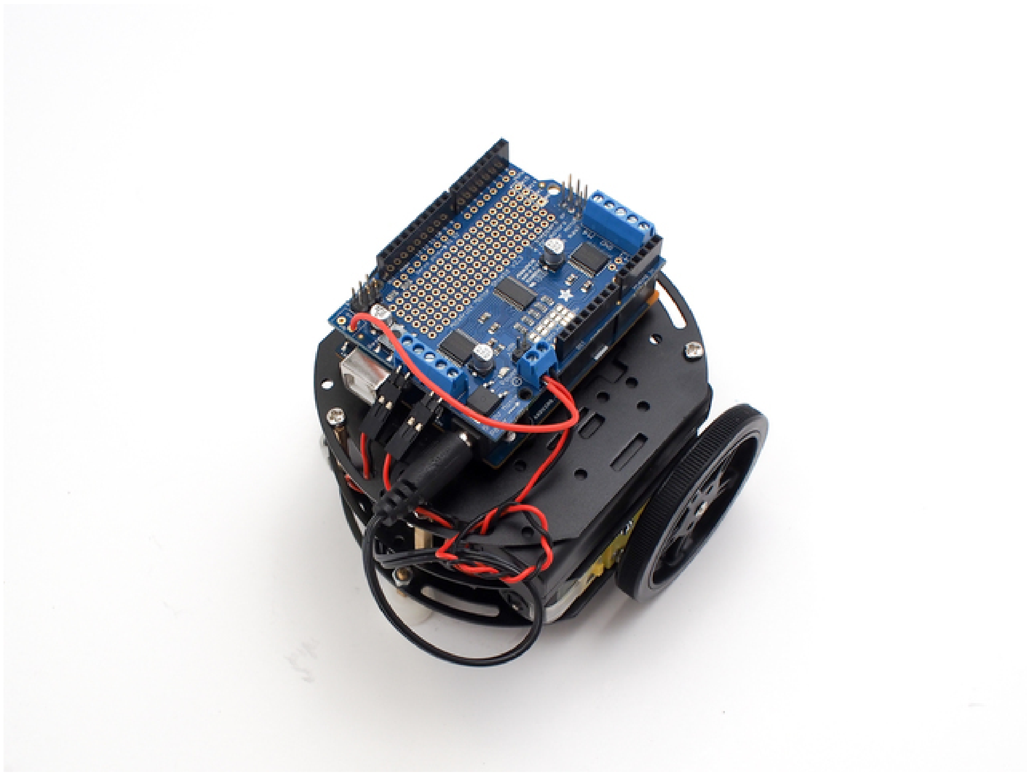
To prepare for moderate to heavy servo use, you'll want to have servos draw from the MotorShield power supply, not the supply of the Arduino 101, otherwise you may run into strange reset behavior when the Arduino power dips too low.

First, flip the MotorShield over and cut the power trace running to the servo power pin.



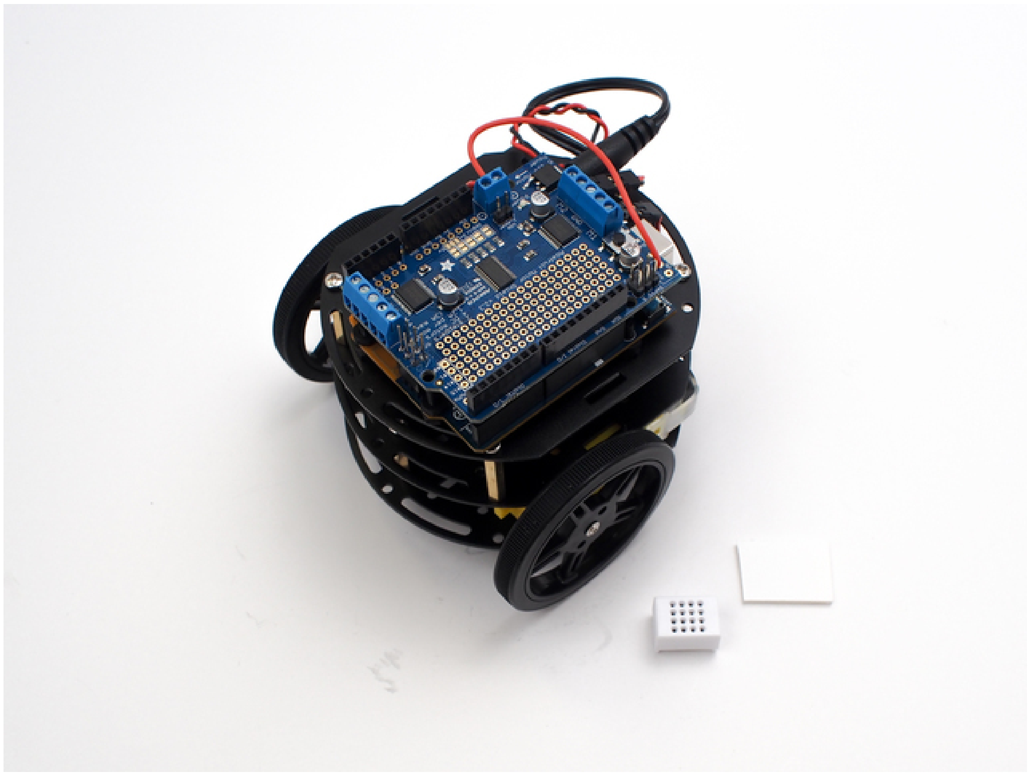
Then, solder a length of wire to the optional servo power pin (the "Opt. Servo" pin closest to the large capacitor and reset button") on the MotorShield that can be screwed into the + power terminal.



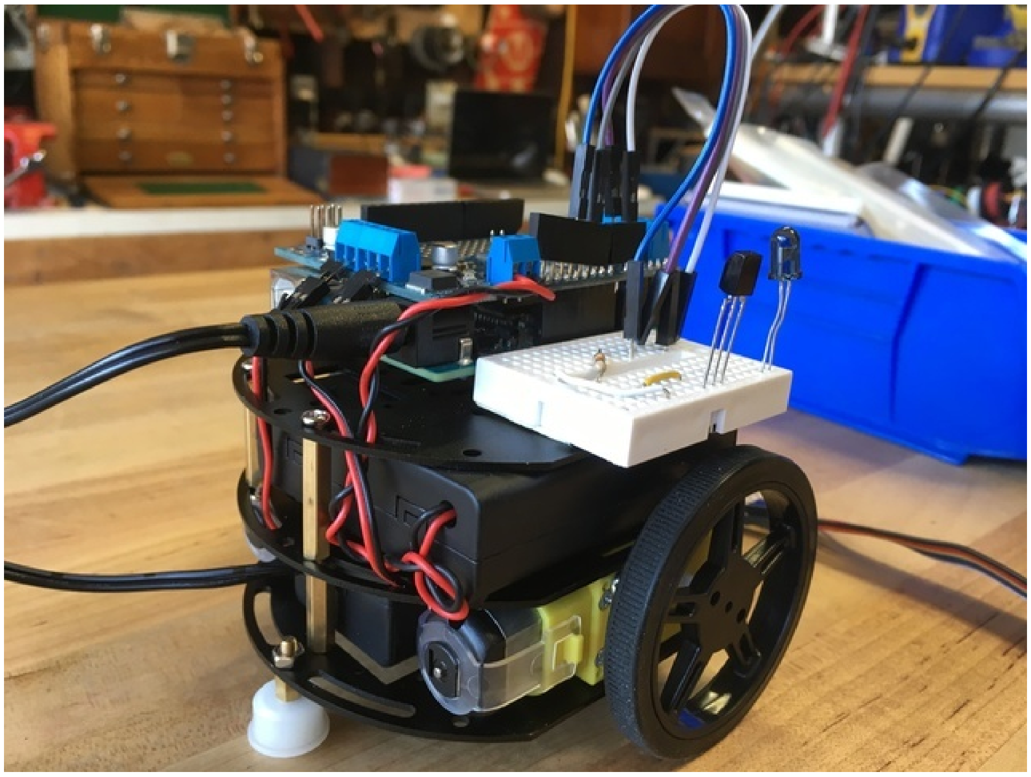


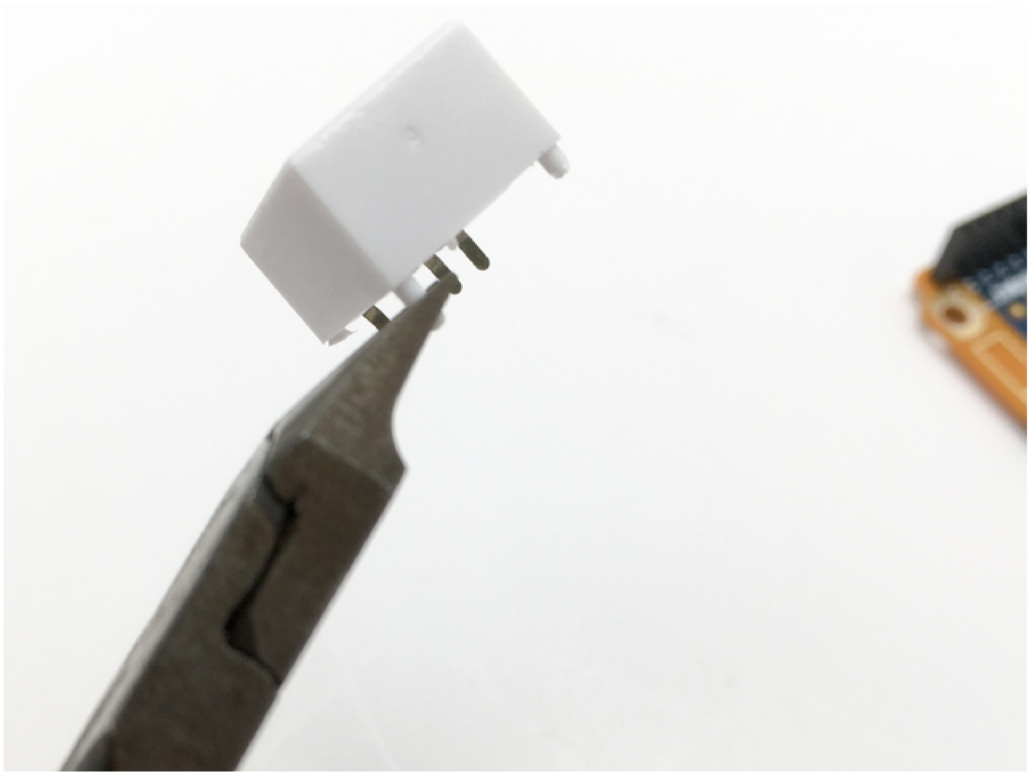
Finishing Touches

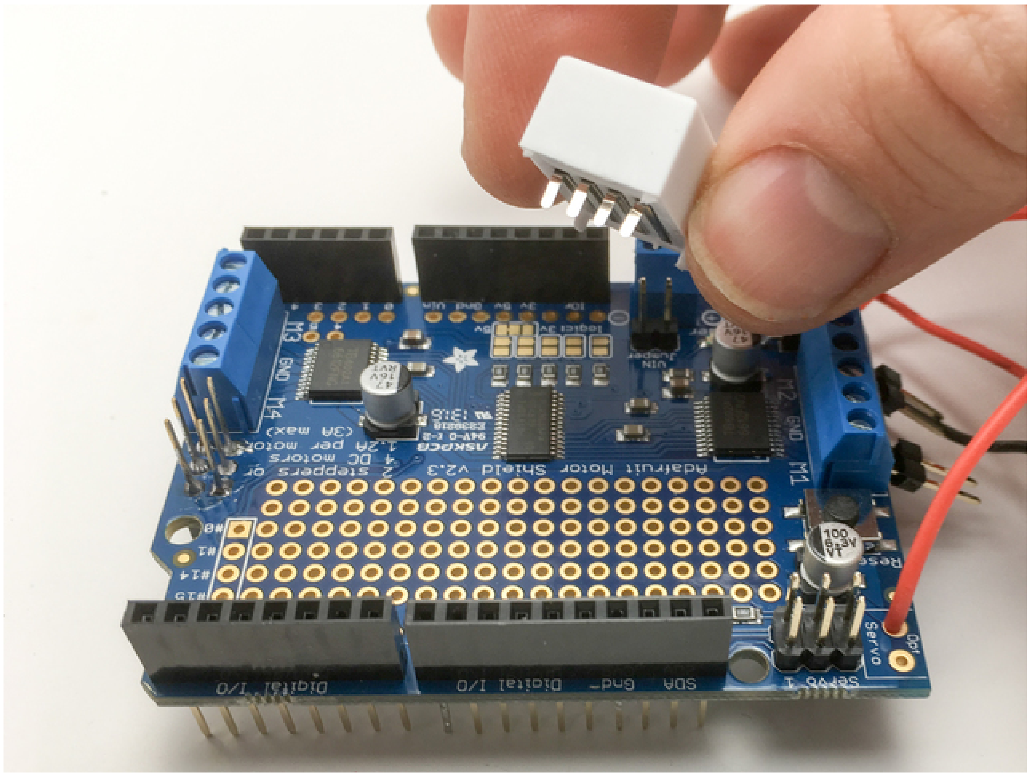
The CurieBot is nearly complete! To enable easy prototyping of small circuits (very small!) we've included a 4x4 baby breadboard. You can use the double stick tape to affix it to the board, or for a more permanent attachment, trim down the breadboard's four legs with some diagonal cutters and then solder its four metal tabs into the MotorShield's prototyping holes.

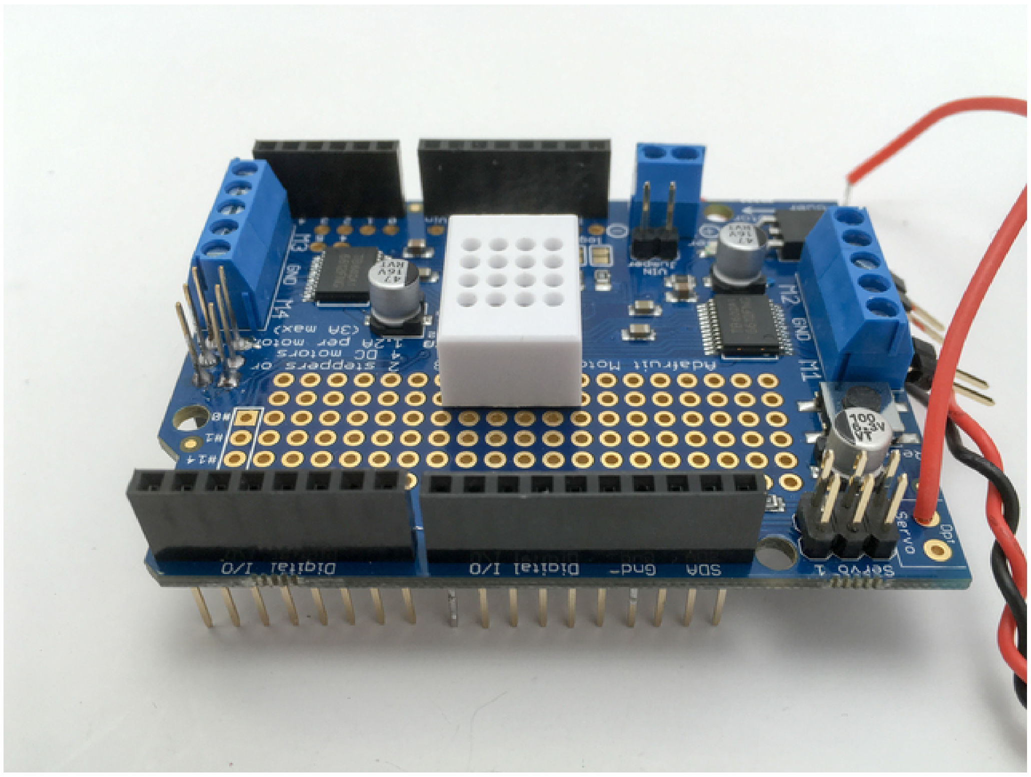


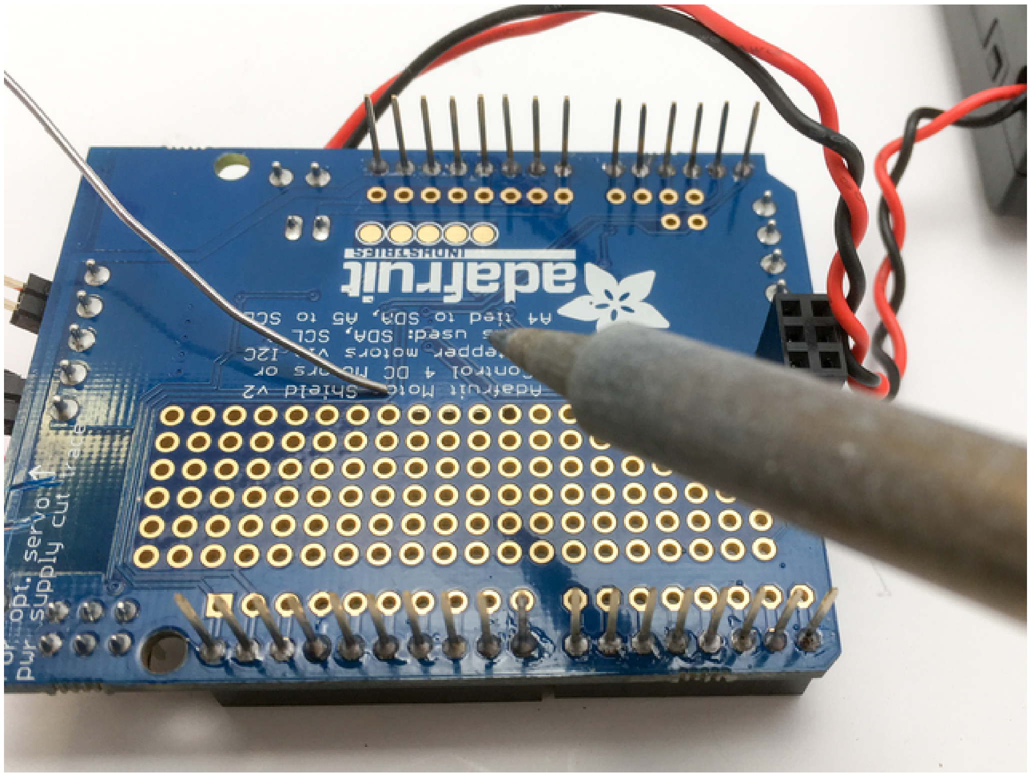
Some kits may include an alternate breadboard as seen here.

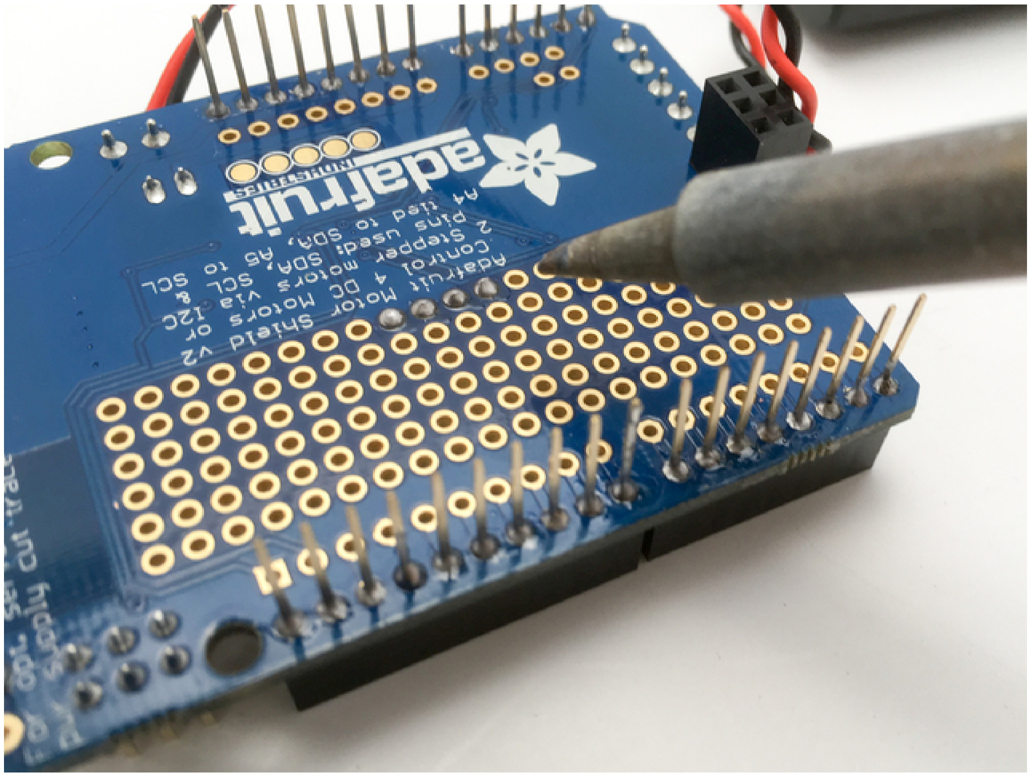


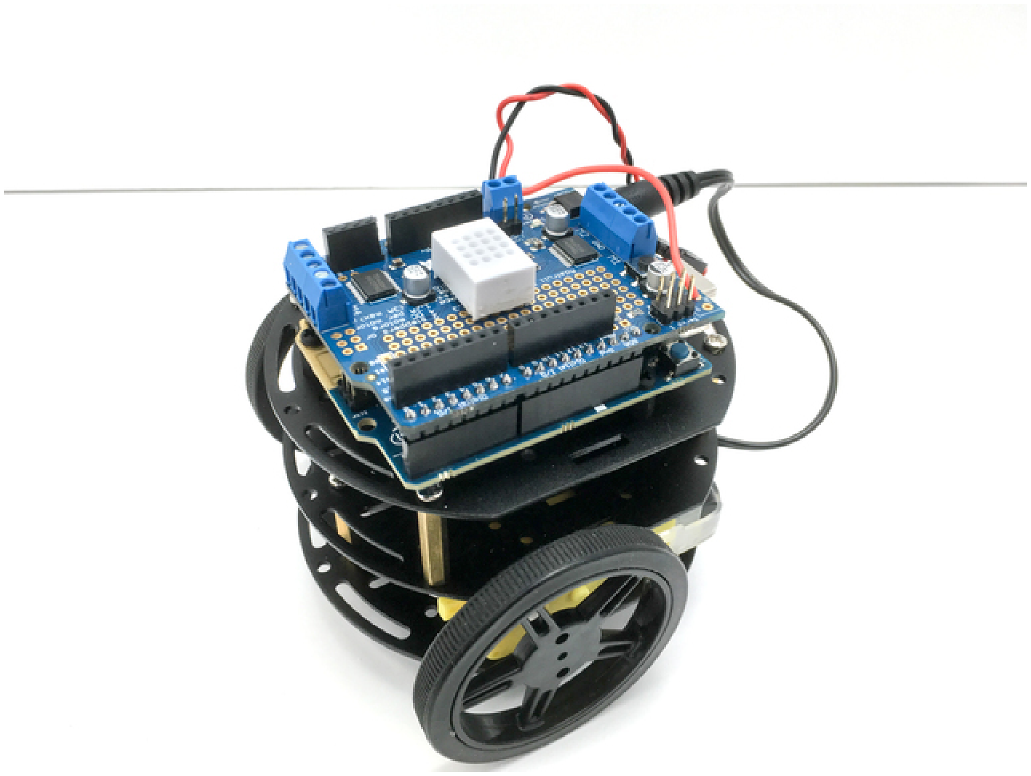



Adafruit Motor Shield v2.3
M1
M2
M3
M4
GND
Servo 1
Digital I/O
SDA

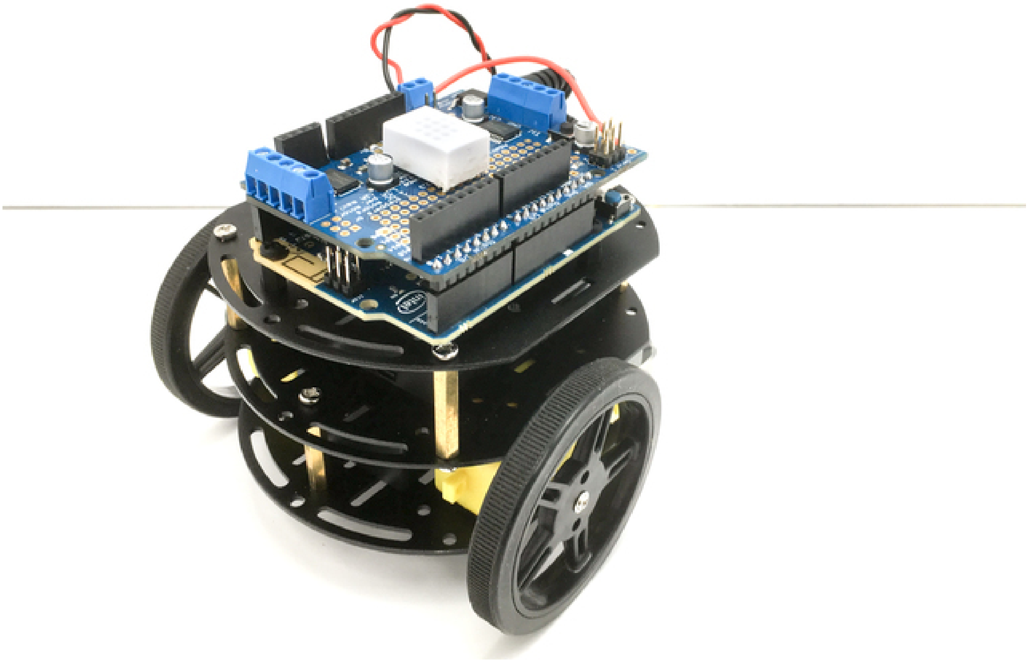


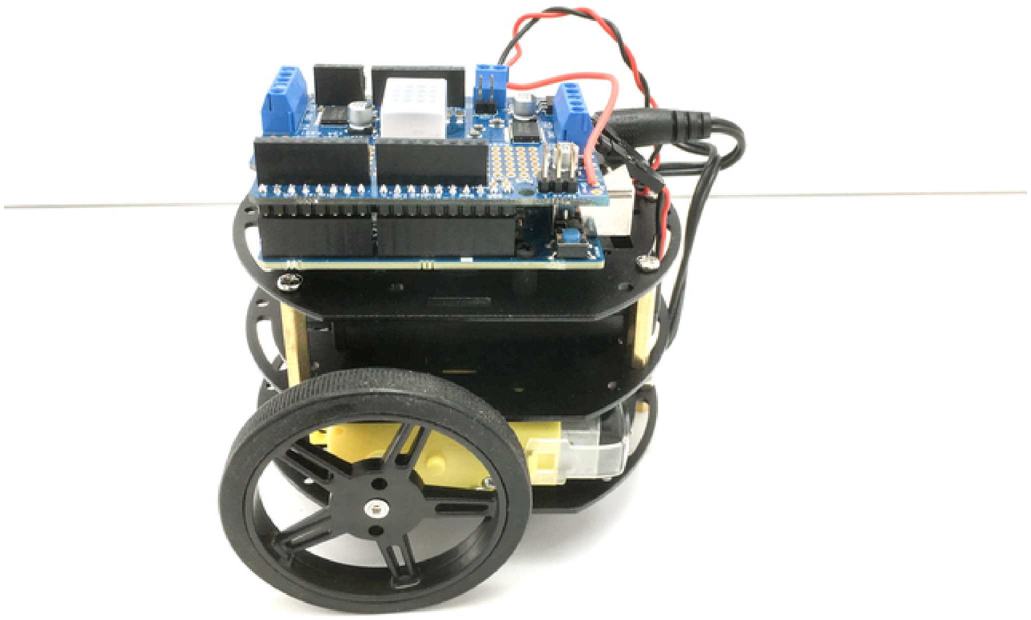


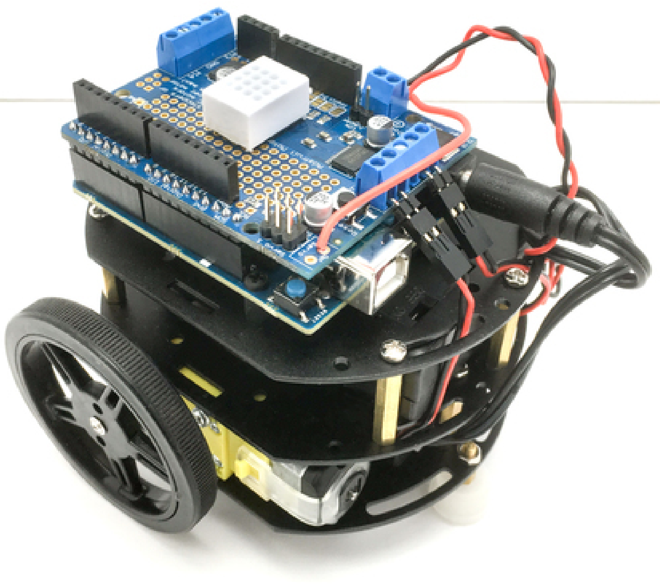




Congratulations, you've built your robot!







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

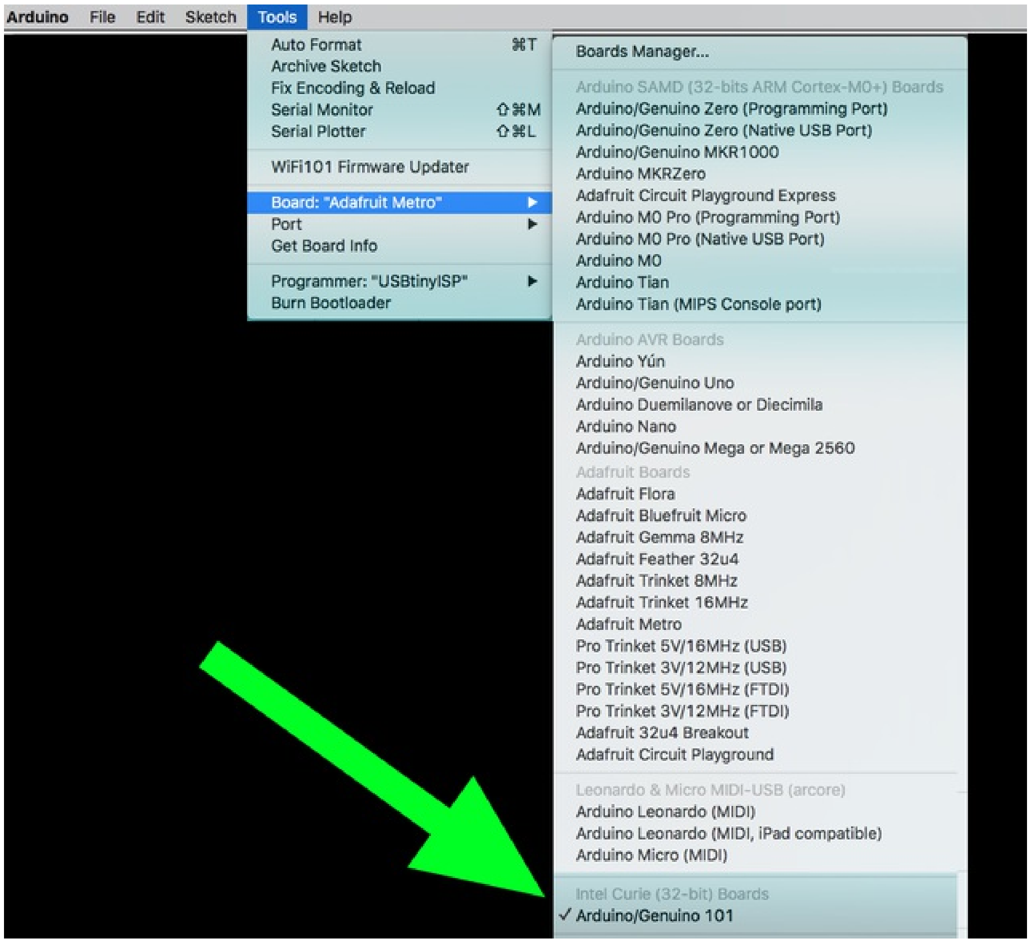
Before we dig into the more complicated code, let's take a minute to break down the simple motor controller code, and how it works to control your robot's motors.

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 (https://adafru.it/pcg), and then come back to this guide to continue.

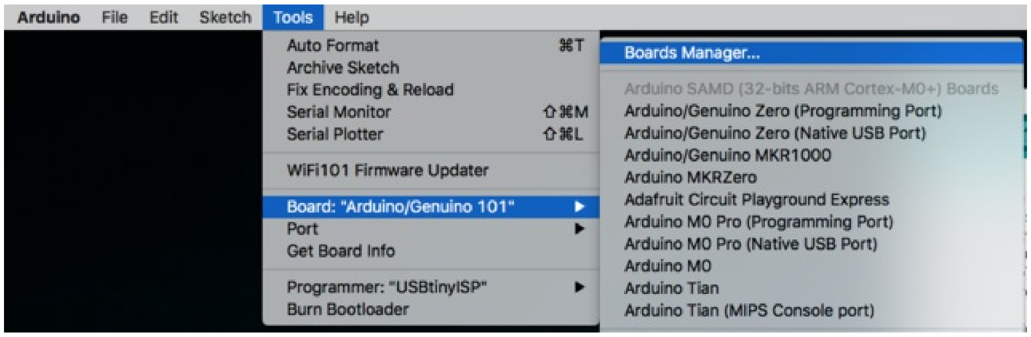
Before going any further, make sure you have a basic understanding of how to program and use an Arduino

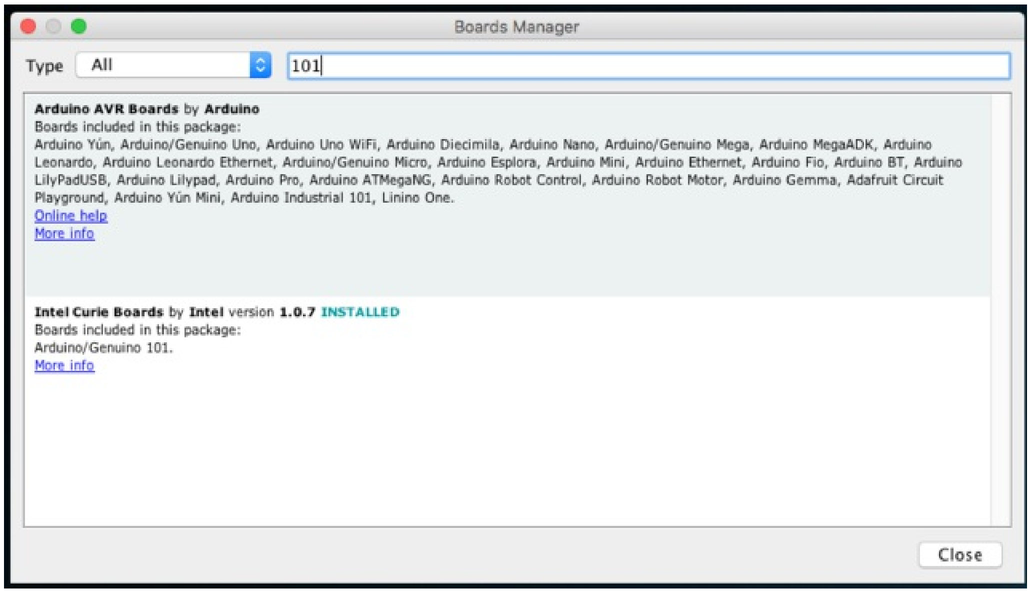
Board Manager

To use the Arduino 101 board, check the Arduino IDE Tools > Board list and select Arduino 101.


Arduino File Edit Sketch Tools Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 Firmware Updater
Board: "Adafruit Metro"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Boards Manager...
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Arduino/Genuino Zero (Programming Port)
Arduino/Genuino Zero (Native USB Port)
Arduino/Genuino MKR1000
Arduino MKRZero
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Arduino Tian (MIPS Console port)
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Adafruit Boards
Adafruit Flora
Adafruit Bluefruit Micro
Adafruit Gemma 8MHz
Adafruit Feather 32u4
Adafruit Trinket 8MHz
Adafruit Trinket 16MHz
Adafruit Metro
Pro Trinket 5V/16MHz (USB)
Pro Trinket 3V/12MHz (USB)
Pro Trinket 5V/16MHz (FTDI)
Pro Trinket 3V/12MHz (FTDI)
Adafruit 32u4 Breakout
Adafruit Circuit Playground
Leonardo & Micro MIDI-USB (arcore)
Arduino Leonardo (MIDI)
Arduino Leonardo (MIDI, iPad compatible)
Arduino Micro (MIDI)
Intel Curie (32-bit) Boards
✓ Arduino/Genuino 101

In case you don't see that board choice, you can click on Arduino IDE Tools > Board > Boards Manager... and then sort the list in the Boards Manager search bar with the word "101" and then update the AVR boards definition and install the 101 board as seen here.


Arduino File Edit Sketch Tools Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 Firmware Updater
Board: "Arduino/Genuino 101"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Boards Manager...
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Arduino/Genuino Zero (Programming Port)
Arduino/Genuino Zero (Native USB Port)
Arduino/Genuino MKR1000
Arduino MKRZero
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Arduino Tian (MIPS Console port)

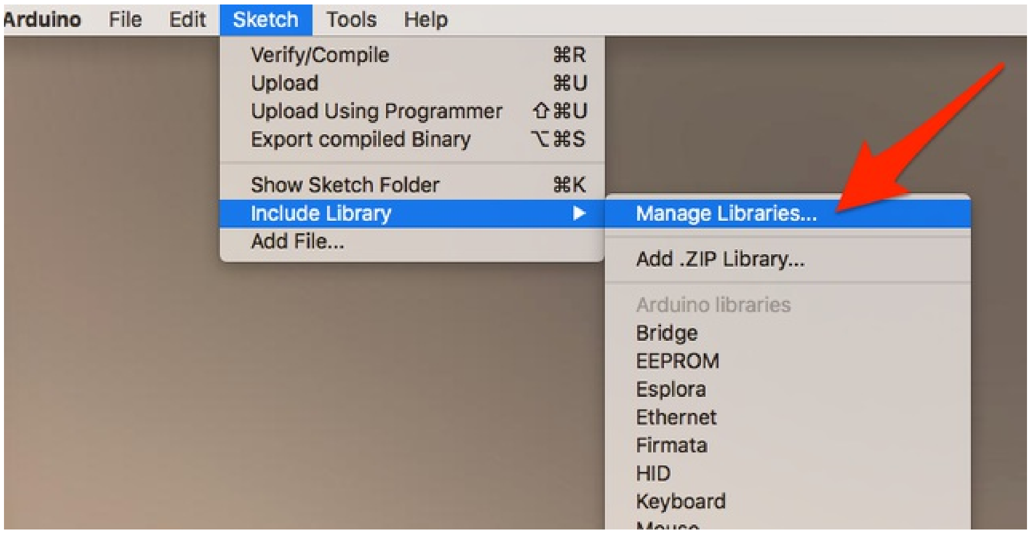


Arduino AVR Boards by **Arduino**
Boards included in this package:
Arduino Yún, Arduino/Genuino Uno, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino/Genuino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino/Genuino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
Online help
More info

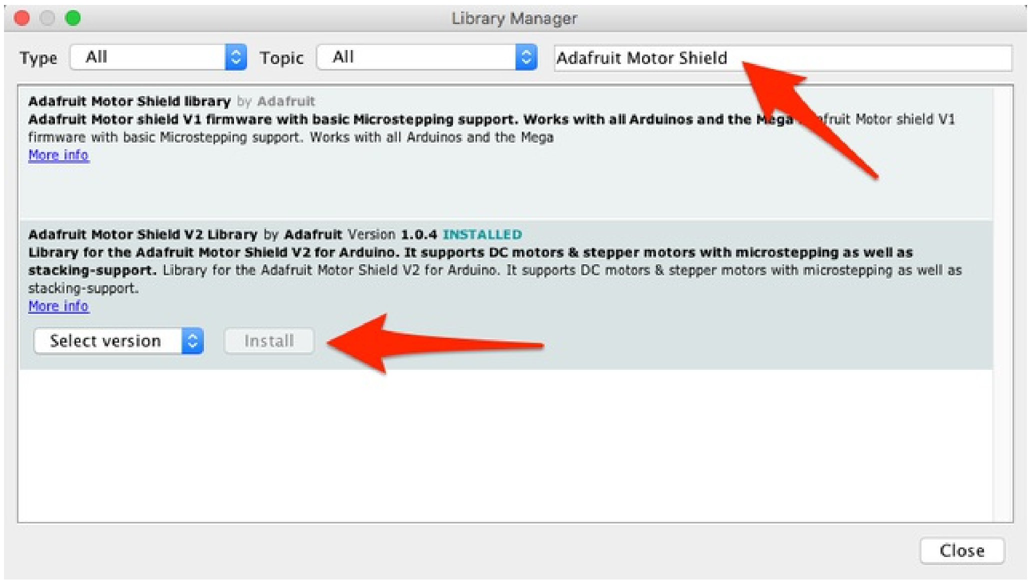
Intel Curie Boards by **Intel** version **1.0.7** **INSTALLED**
Boards included in this package:
Arduino/Genuino 101.
More info

Code Library

For your robot to work correctly, you will need to install the MotorShield library. To install 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.

DC Motor Test

Now that you have the library installed, let's open up the example sketch to try out the DC motors on your robot. You can find the example sketch **DC Motor Test** in the **Arduino File>Examples>Adafruit Motor Shield V2** menu. Open it and upload it to your CurieBot.

Note this snippet of code below. It's instructing only Motor 1 to rotate.

```
// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);
// 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.

- Turn on AA battery holder switch to **ON**
- Select 'Arduino/Genuino 101' as the board under the Arduino Tools menu, and...
- Upload this code 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, M2
Adafruit_DCMotor *myMotor = AFMS.getMotor(2);
// You can also make another motor on port M2
//Adafruit_DCMotor *myOtherMotor = AFMS.getMotor(2);
```

With the motor port changed to 2, 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, reset the first motor to port 1 and keep the second motor on port 2 like so:

```
// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);
// You can also make another motor on port M2
Adafruit_DCMotor *myOtherMotor = AFMS.getMotor(2);
```

In this case, we have `myMotor` set to `M1`, which is our right side motor, and `myOtherMotor` is set to `M2`, 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 M1
Adafruit_DCMotor *leftMotor = AFMS.getMotor(1);
// Set up the right motor on port M2
Adafruit_DCMotor *rightMotor = AFMS.getMotor(2);
```

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.**

```
/*
CurieBot MotorShield test 01
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 M1
Adafruit_DCMotor *leftMotor = AFMS.getMotor(1);
// Set up the right motor on port M2
Adafruit_DCMotor *rightMotor = AFMS.getMotor(2);

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:

```
/*
CurieBot MotorShield Test 02
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 M1
Adafruit_DCMotor *leftMotor = AFMS.getMotor(1);
// Set up the right motor on port M2
Adafruit_DCMotor *rightMotor = AFMS.getMotor(2);

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 (https://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 later.

MotorShield Library

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

We will need the Adafruit MotorShield library. Learn more about this library and download it here. (https://adafru.it/t8A)

You probably already have this installed if you followed the previous page but triple check now!

The Code

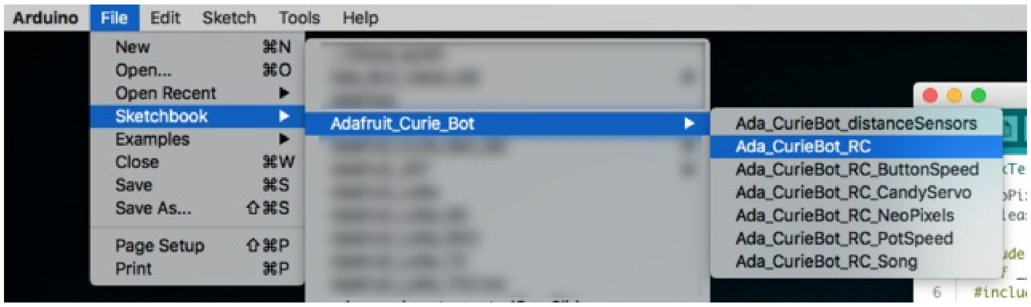
Once you have the libraries installed, you will want to download the latest rover code. Click the button below to download.

https://adafru.it/usA

https://adafru.it/usA

Once the file has downloaded, unzip it and rename the folder from **Adafruit_CurieBot-master** to **Adafruit_CurieBot**

Then, place this folder in your Arduino sketches directory and open the sketch Ada_CurieBot_RC.



There are a few other sketches included in the download which you'll use later on in this guide.

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 use a for loop to slowly ramp up the speed until it hits max speed. Learn more about for loops here (https://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.

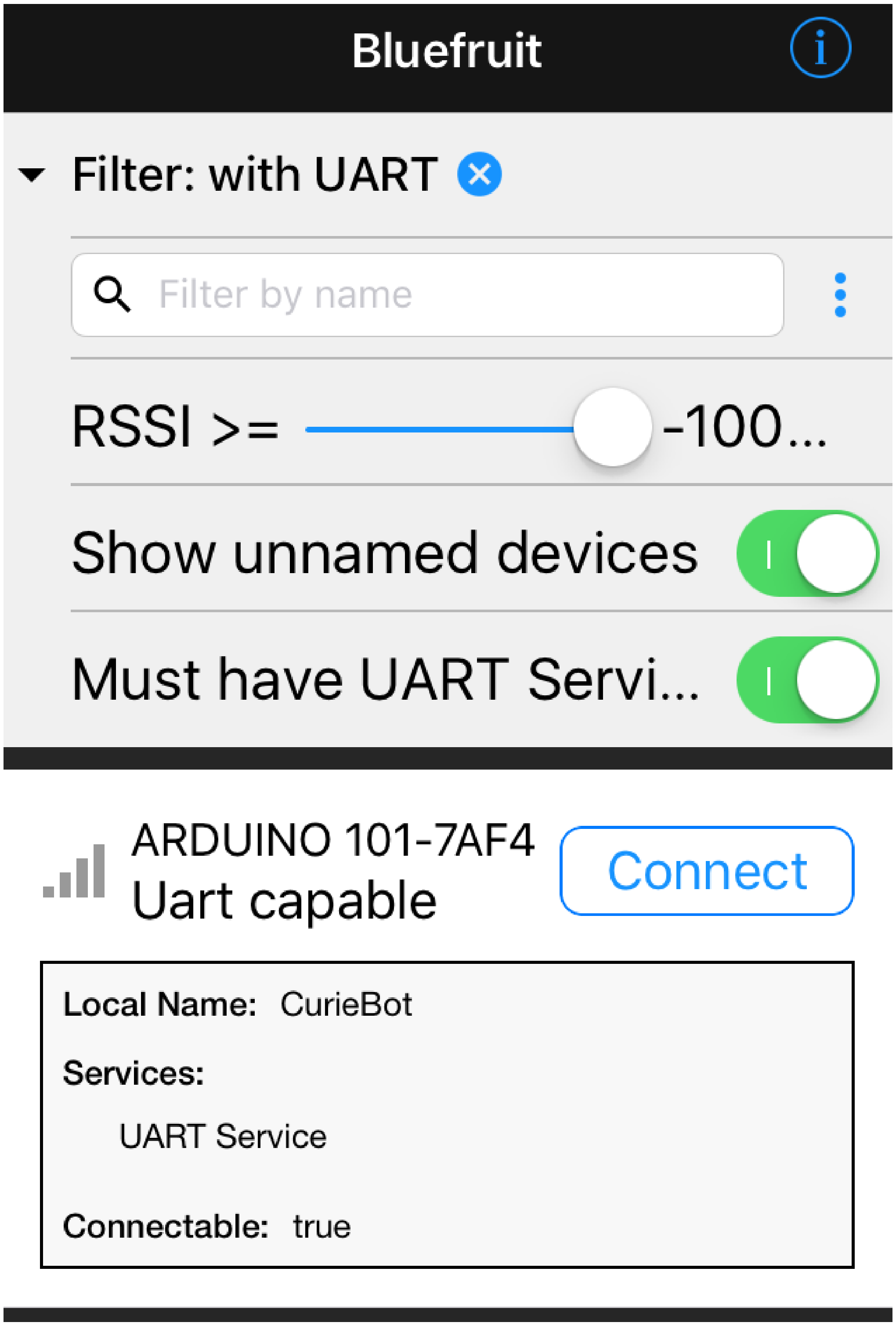
https://adafru.it/ddu

https://adafru.it/ddu

https://adafru.it/f4G

https://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 Arduino 101 (or possibly named CurieBot, depending on your mobile device), and tap the Connect button.


Bluefruit

Filter: with UART

Filter by name

RSSI >= -100...

Show unnamed devices

Must have UART Servi...

ARDUINO 101-7AF4
Uart capable

Connect

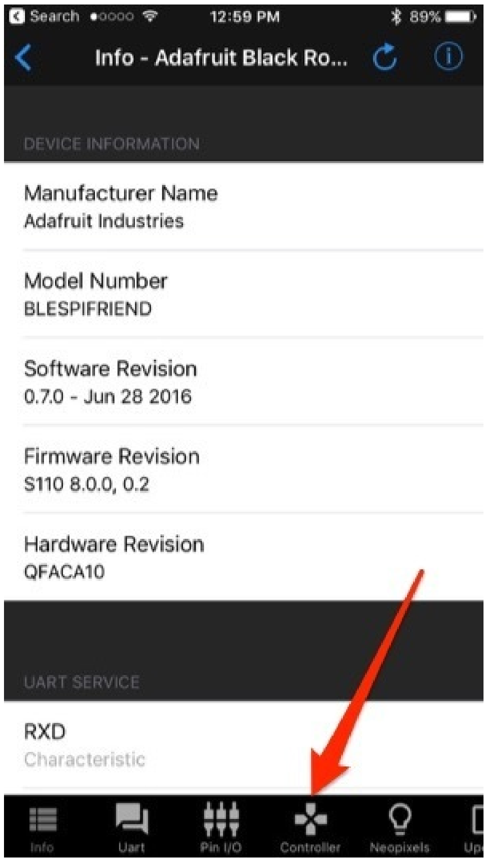
Local Name: CurieBot

Services:

UART Service

Connectable: true

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


Search ●○○○○ 12:59 PM 89%

Info - Adafruit Black Ro...

DEVICE INFORMATION

Manufacturer Name
Adafruit Industries

Model Number
BLESPIFRIEND

Software Revision
0.7.0 - Jun 28 2016

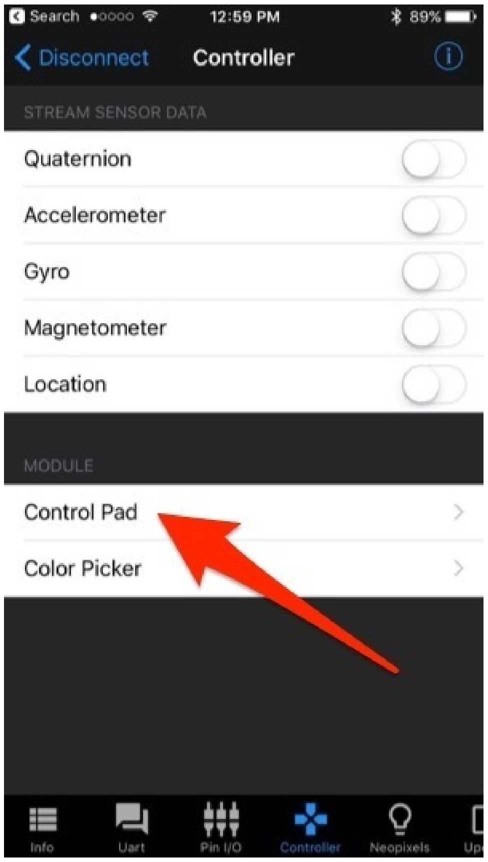
Firmware Revision
S110 8.0.0, 0.2

Hardware Revision
QFACA10

UART SERVICE

RXD
Characteristic

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


Search ●○○○○ 12:59 PM 89%

Disconnect Controller

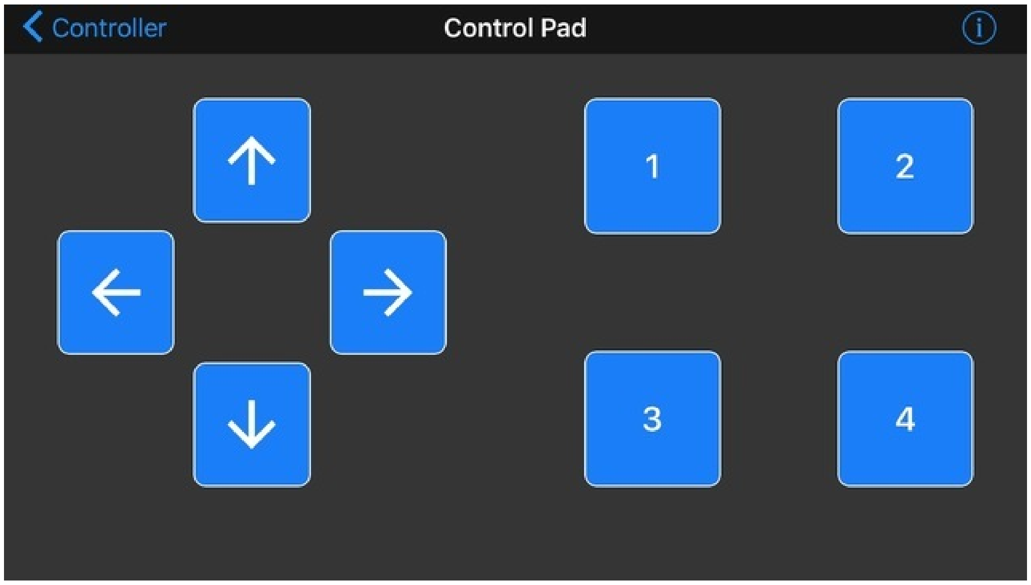
STREAM SENSOR DATA

Quaternion
Accelerometer
Gyro
Magnetometer
Location

MODULE

Control Pad
Color Picker

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 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.

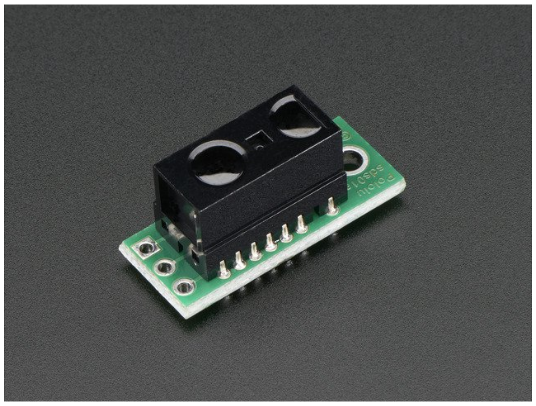
The simplest way to do this is to add a couple robot whiskers to sense when it physically runs into a wall or object. You simply write some code to listen for when the switch has been triggered, stop, turn around, and go forward again. These Micro Switches with a wire are perfect for this.

Micro Switch w/Wire - Three Terminals

$2.95
OUT OF STOCK

OUT OF STOCK

While this is a great way to navigate around obstacles, 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 (https://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

$6.95
IN STOCK

ADD TO CART

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.

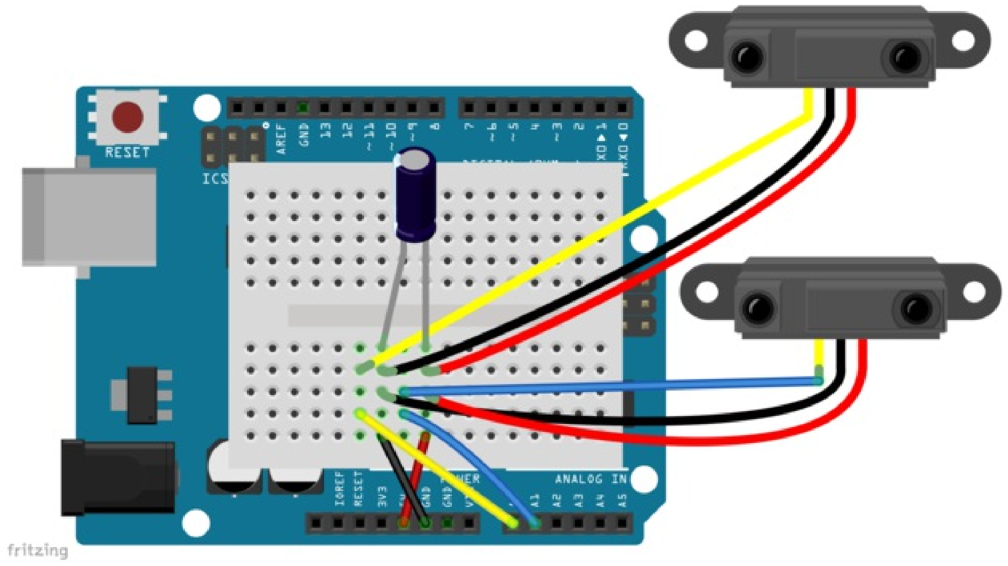
Mounting the Sensors

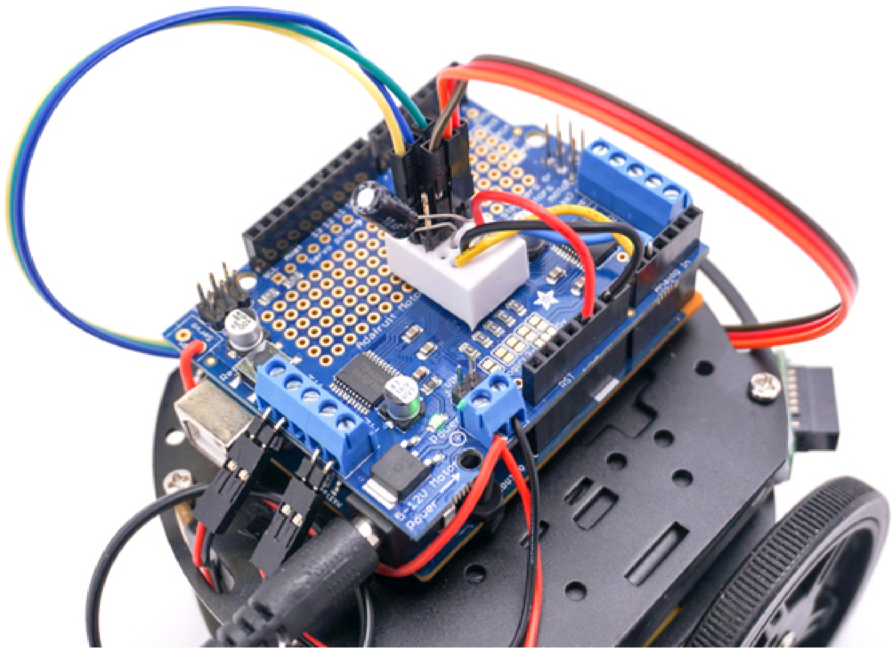
There are a ton of ways you can mount this sensor to your robot. The easiest way would be to just a bit of double sided foam tape to stick it in place, but I decided to make a super simple 3D printed mount.

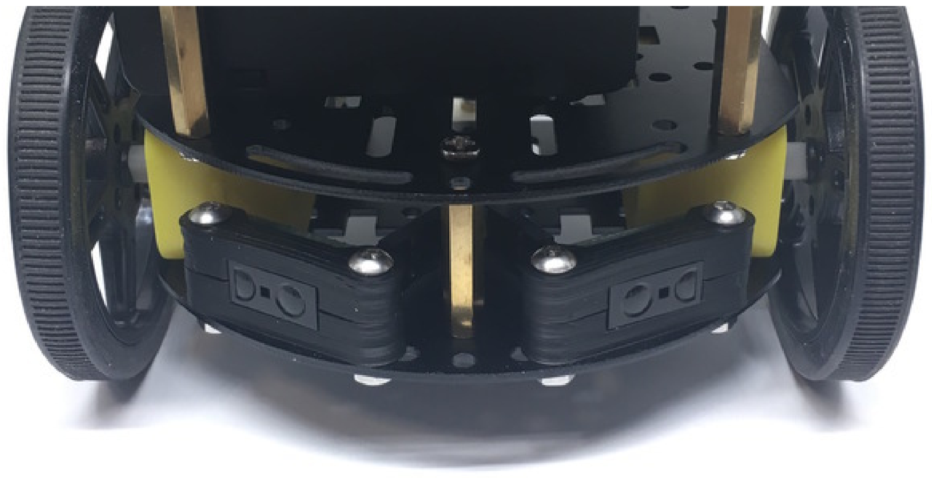
https://adafru.it/t9b

https://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 A0, and the right sensor to pin A1.







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.

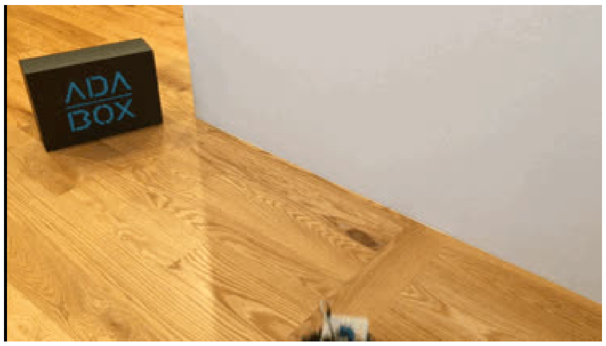
https://adafru.it/t9c

https://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 M1 & M2 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(1);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(2);

// And connect the Sharp distance sensors
int leftSensor = A0;
int rightSensor = A1;

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);
  }
}
```

Open the **Ada_CurieBot_distanceSensors** sketch in the Arduino IDE and upload it to your robot.

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.

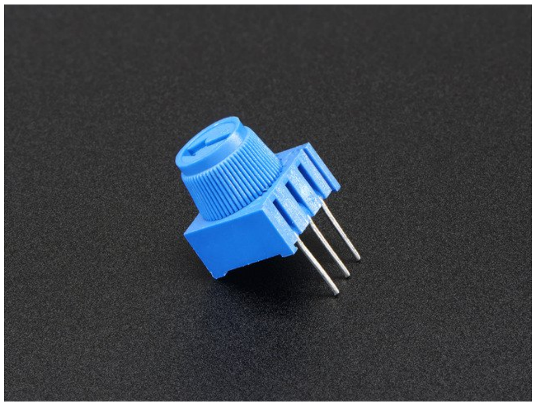


This is just the beginning of what you can do with distance sensing. The next steps are up to you. You can integrate this code into the bluetooth controller code to turn on and off auto mode with a button press. What other ideas can you think of?

Control Your Robot's Speed

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

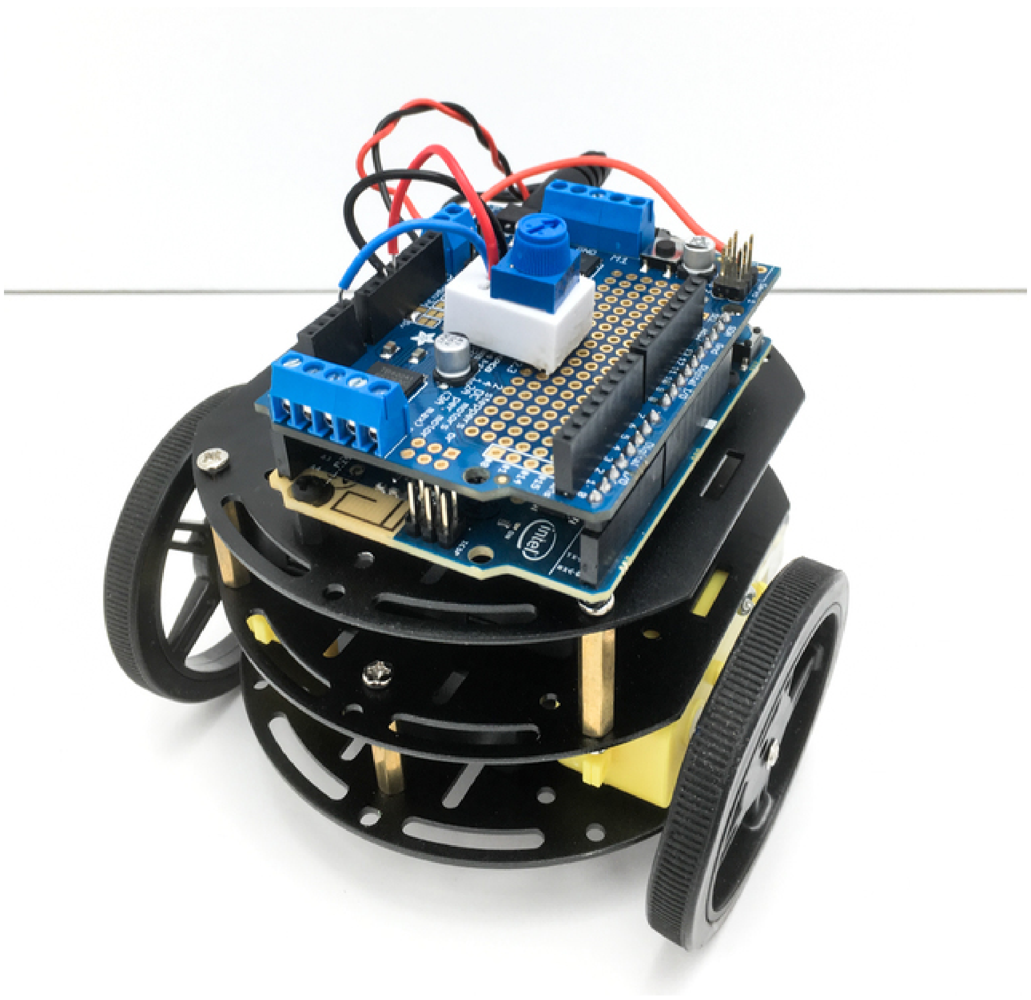
The first way we are going to adjust the speed is with a simple breadboard trim potentiometer. They are super cheap! Pick one up on the Adafruit Shop:



Breadboard trim potentiometer

$1.25
IN STOCK
ADD TO CART

Potentiometers, or Pots for short, are variable resistors that allow us to send different voltages to the analog pin. Wiring it up is super simple. Just connect one of the outside pins to the 3.3V pin, and the other outside pin to ground. Then, connect the middle pin to the A0 pin on the Arduino 101.



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 in the Ada_CurieBot_RC code.

```
//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);

  if (len == 0) return;

  // Read from Accelerometer input
  if( accelMode() ) {
    lastAccelPacket = millis();
    modeToggle = true;
    return;
  }

}
```

What we are doing here is reading that analog pin, and the Arduino 101 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 (https://adafru.it/tbc).

Open the **Ada_CurieBot_RC_potSpeed** sketch in the Arduino IDE and upload it to your robot.

Then, turn the pot, then press forward on the controller in the Bluefruit LE app to see how it works. Try turning knob of the potentiometer different amounts in both directions 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);
```

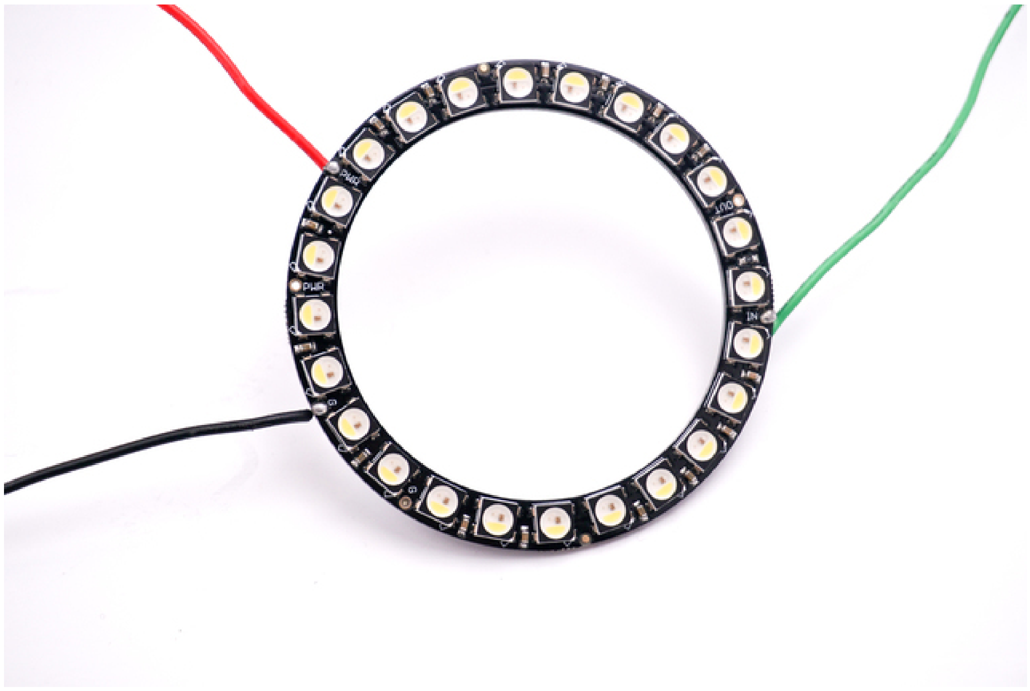
Then, all we need to do is increment the speed up every time we press the 1 button, and down every time we press the 3 button.

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

if(buttnum == 2){
}

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

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

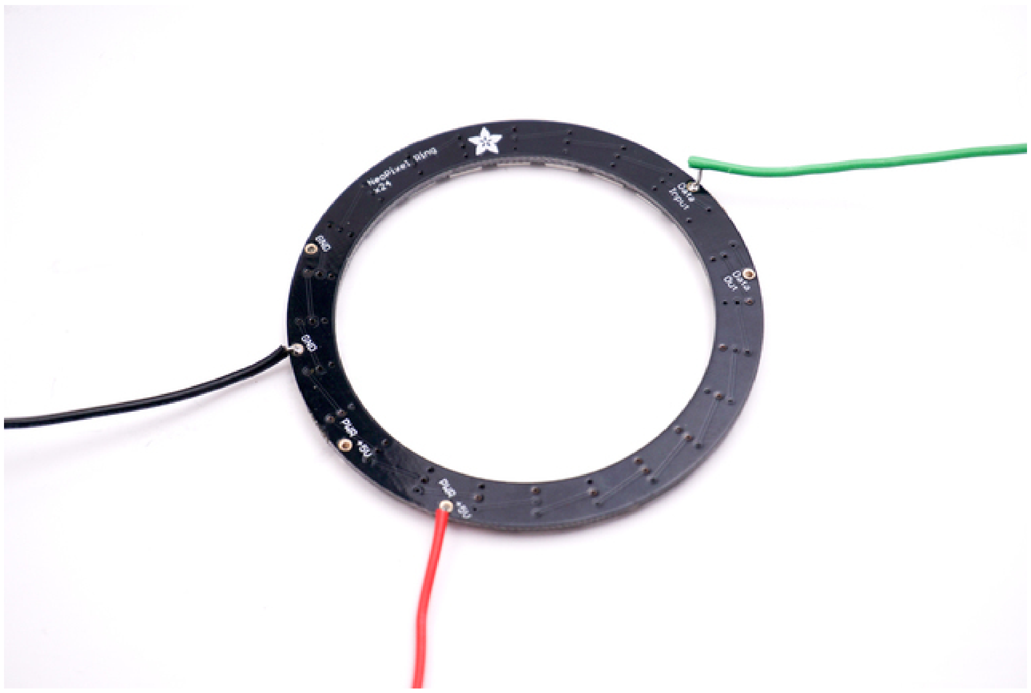
Open the **Ada_CurieBot_RC_buttonSpeed** sketch in the Arduino IDE and upload it to your robot.

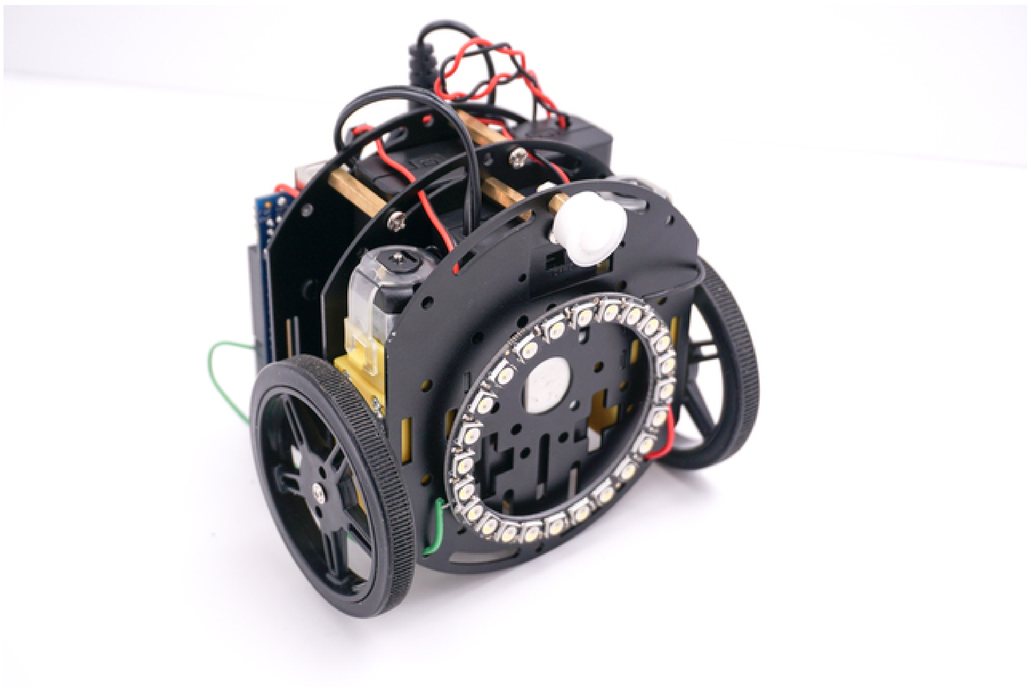
Bling Out Your Bot

You can add some fresh ground effects to your bot using a NeoPixel 24 ring!



Solder leads to the **DIN**, **+5V**, and **GND** pins on the NeoPixel ring. Solid core hookup wire is a nice choice, as it can serve double duty as the structural connection to hold the ring in place underneath.





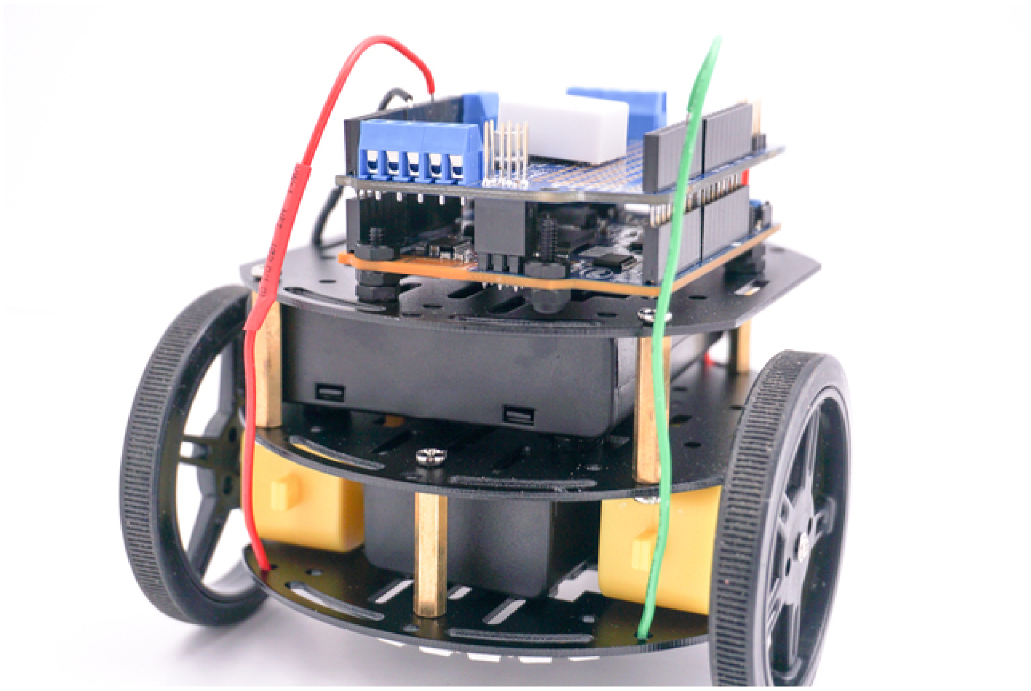
Then, make the following connections from the NeoPixel ring to the Arduino 101:

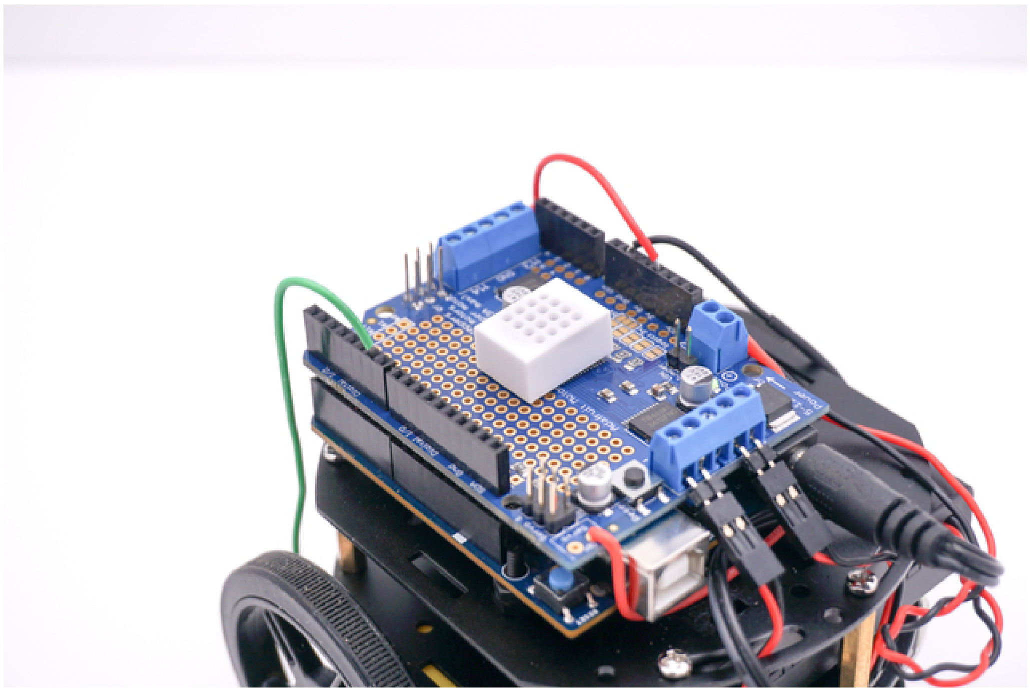
DIN to **pin 6**

+5V to **3.3V**

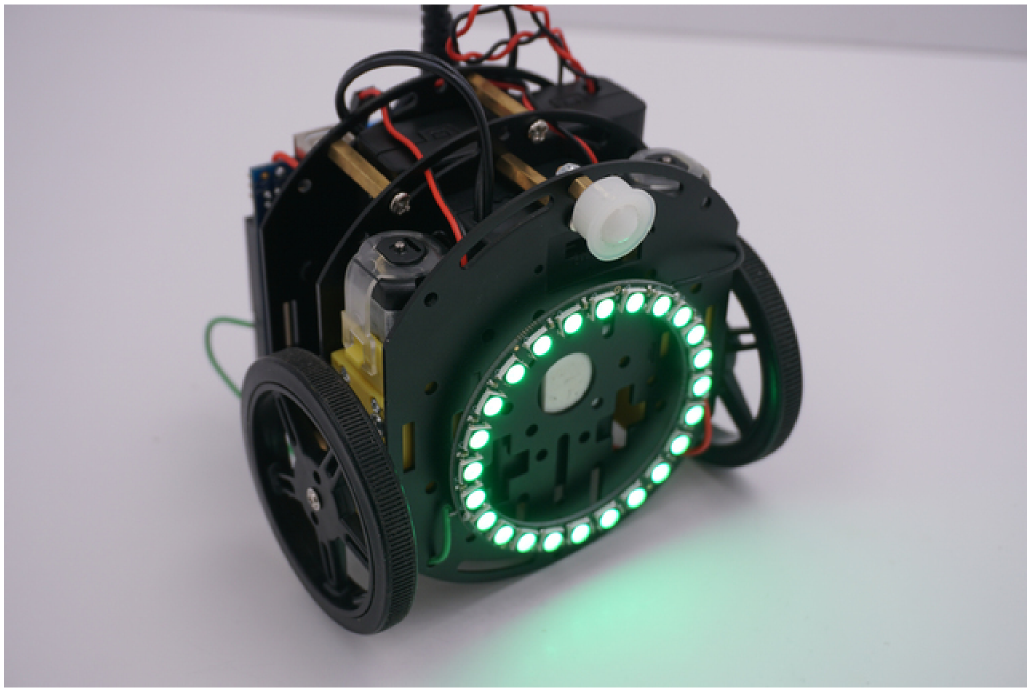
GND to **GND**

Since the Arduino 101 operates on 3.3V logic level, the NeoPixel should be powered by 3.3V instead of the usual 5V to avoid problems.

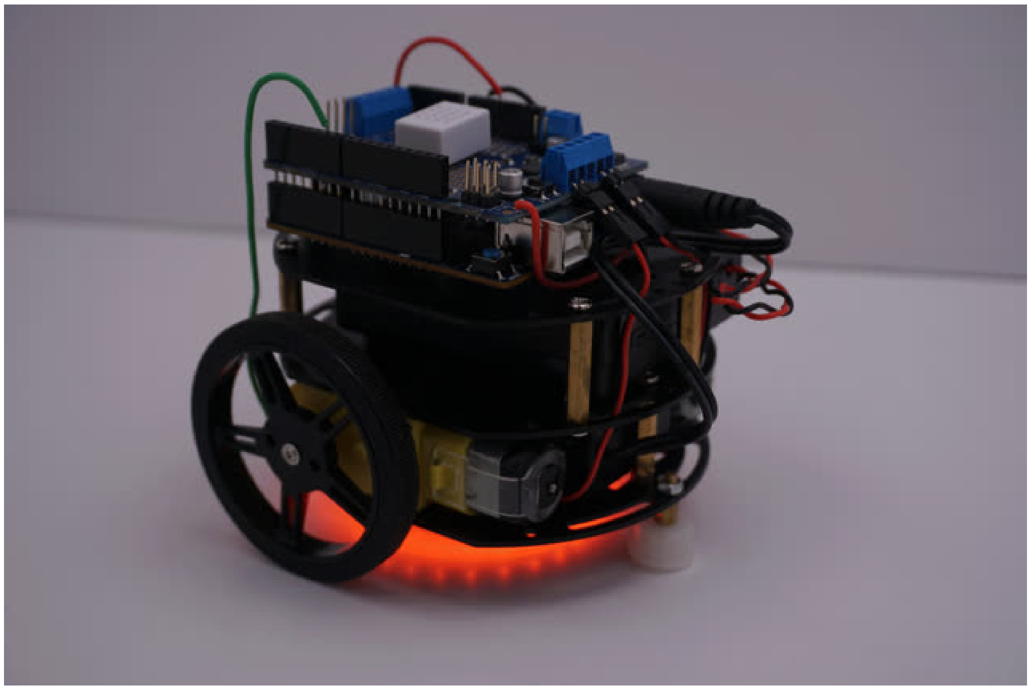




Open the **Ada_CurieBot_RC_NeoPixels** sketch in the Arduino IDE and upload it to your robot.



You can now use the Bluefruit LE app's color picker control to set your CurieBot's underglow right before you go cruising!



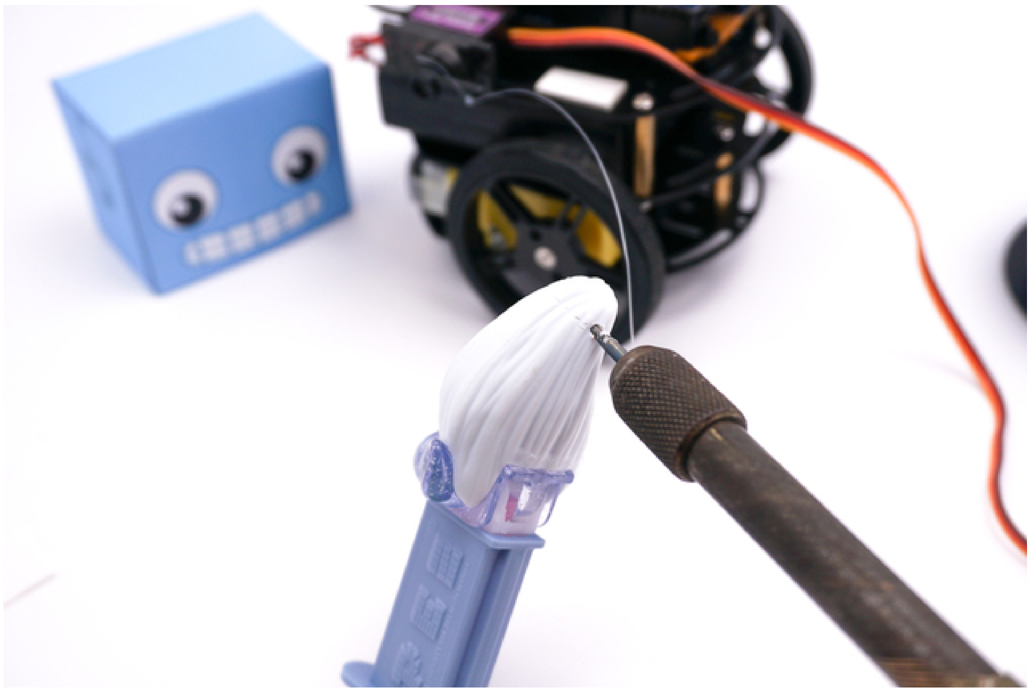
Servo Up Some Sweets

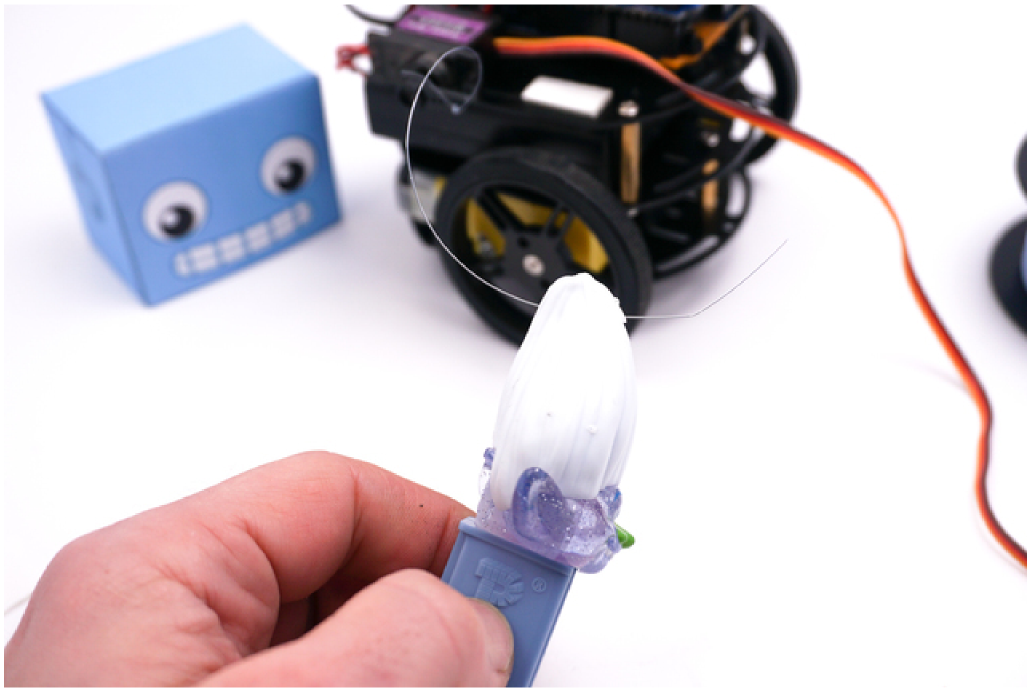
CurieBot is a robot friend, so why not modify your bot to serve candy from a dispenser? That would be totally sweet!

The MotorShield on your CurieBot can be used to control not just DC motors, but also steppers and servos. We can run a small hobby servo from the shield to open an close the head of a candy dispenser.

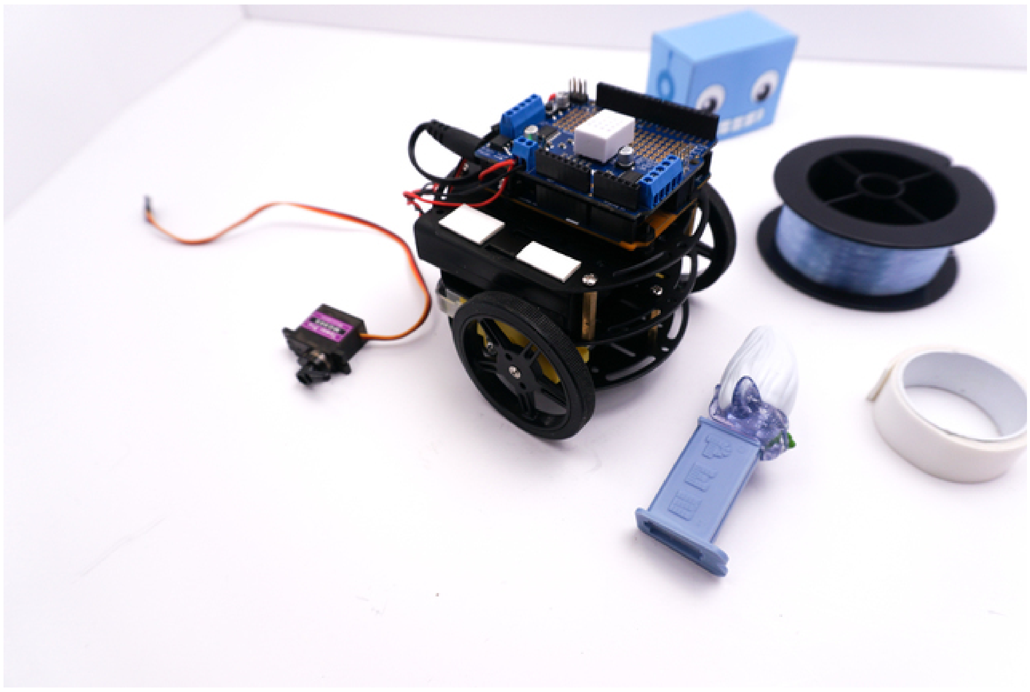
It can be difficult to mount actuators (in this case, the servo) right at the joint, so we often use rigid or flexible linkages to actuate them remotely. In our case, we'll mount our servo at the base of the candy dispenser, and use some fishing line to transmit the rotation to the joint.

Prepare the candy dispenser by drilling a small hole in the top of its head so you can tie the filament on for the servo to tug.

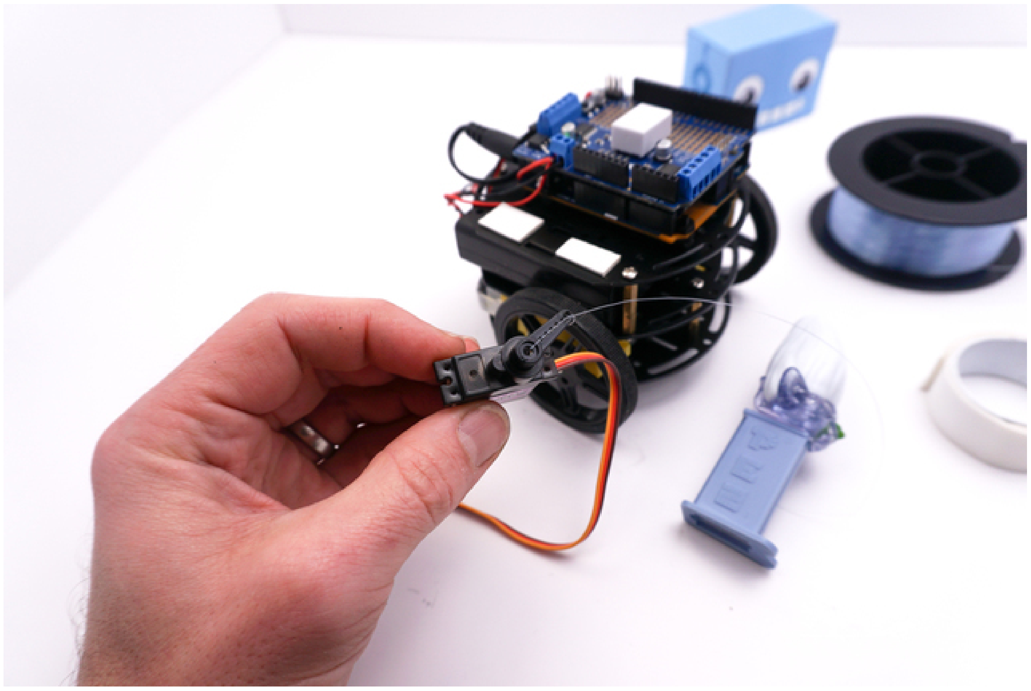


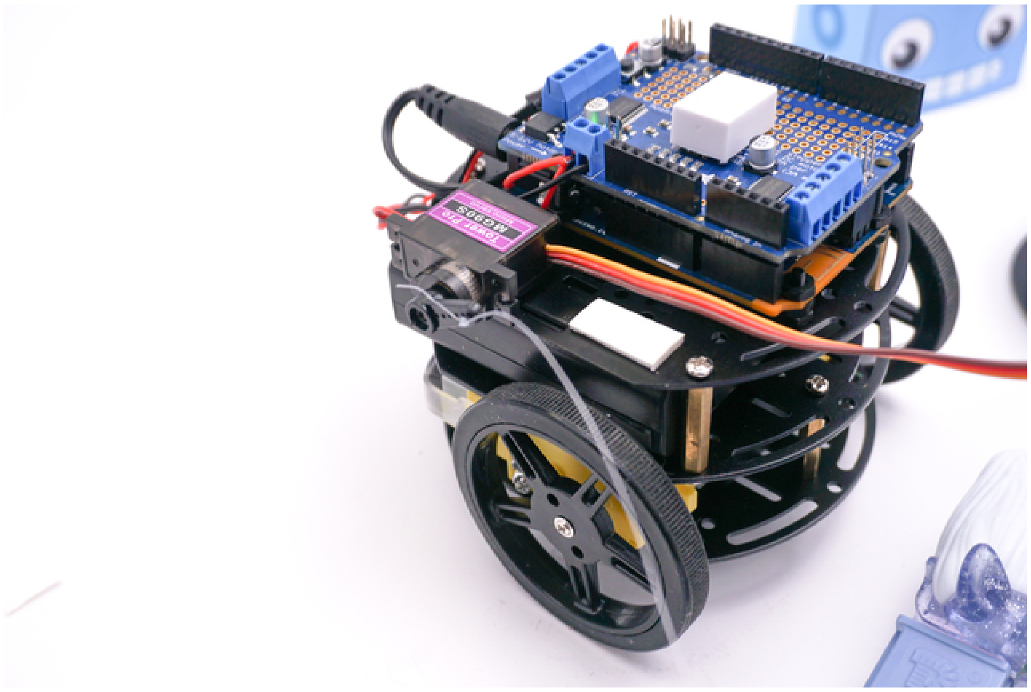


Use a couple of pieces of double stick foam tape to mount the micro servo and candy dispenser to the top plate of CurieBot. You can also drill a small hole into the foot of the candy dispenser base and screw it in place on one of the robot's brass standoffs.

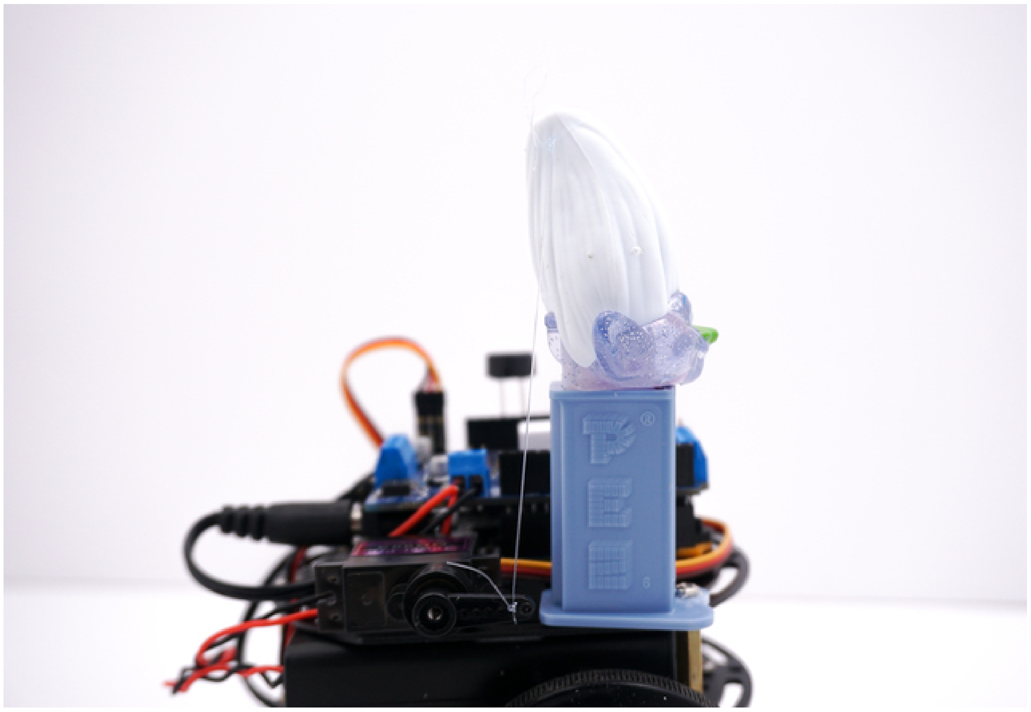


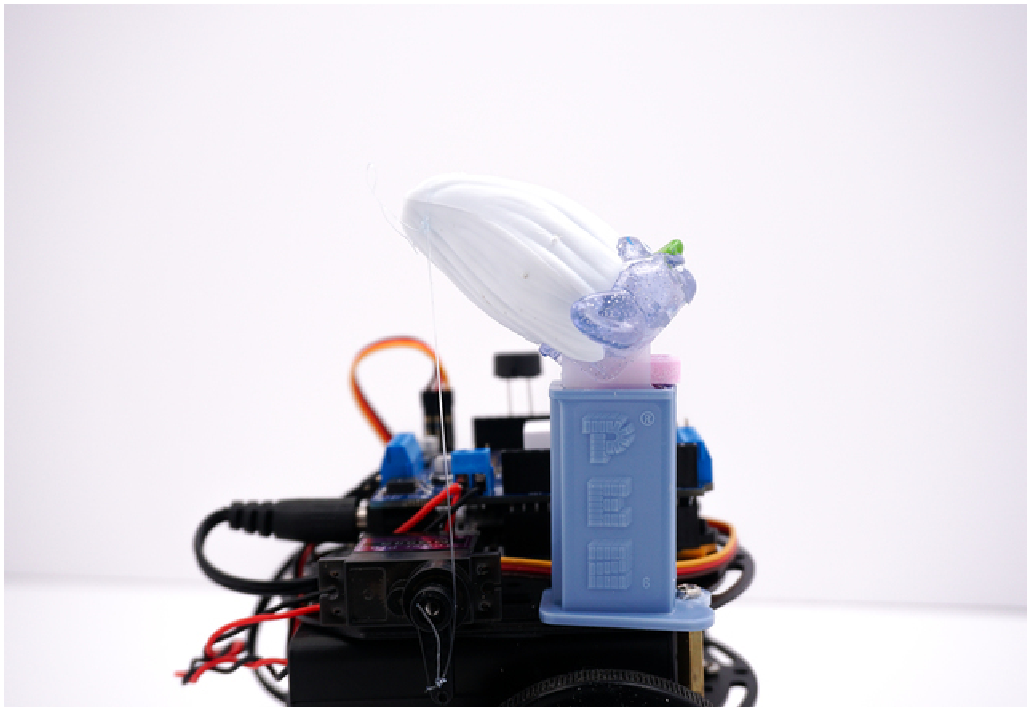
Mount the servo, tie the filament to one of the servo horn's holes.





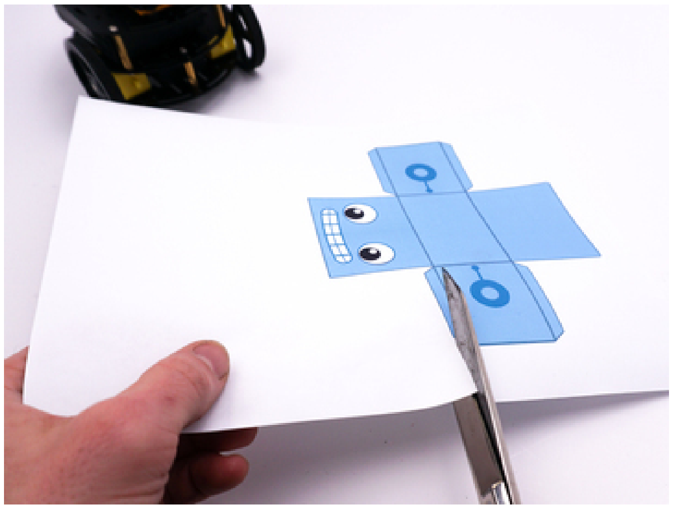
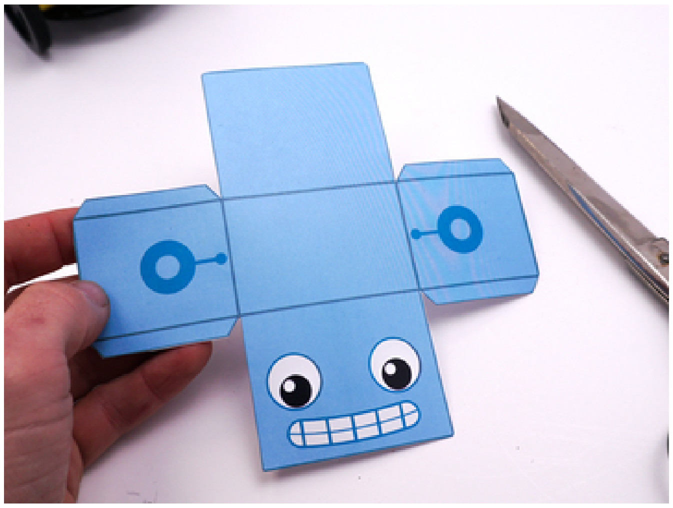
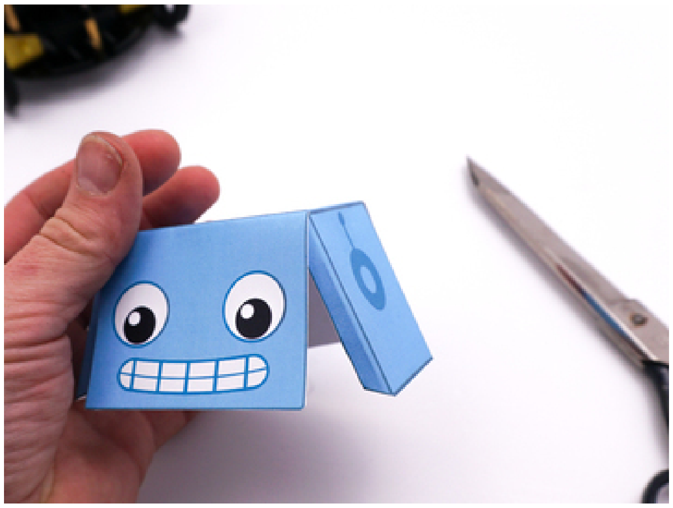
Tie the filament to the dispenser's head so the line is taut when the servo is at its resting position and when rotated downward, as seen in the images.

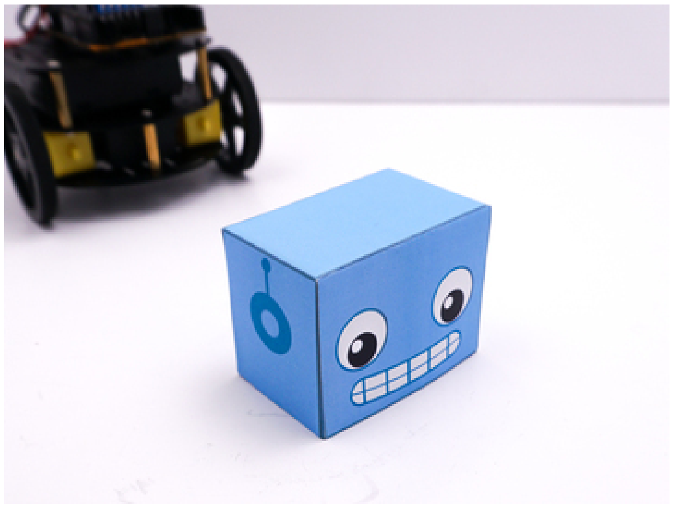




Mini- or full-sized candy dispensers are a great base for this delivery mechanism, but the theme may leave something to be desired. Let's dress it up in papercraft Adabot styling!

Print out a copy of the Adabot papercraft head .pdf file linked below, and then cut out, fold, and glue or tape it into a nice, boxy robot head.

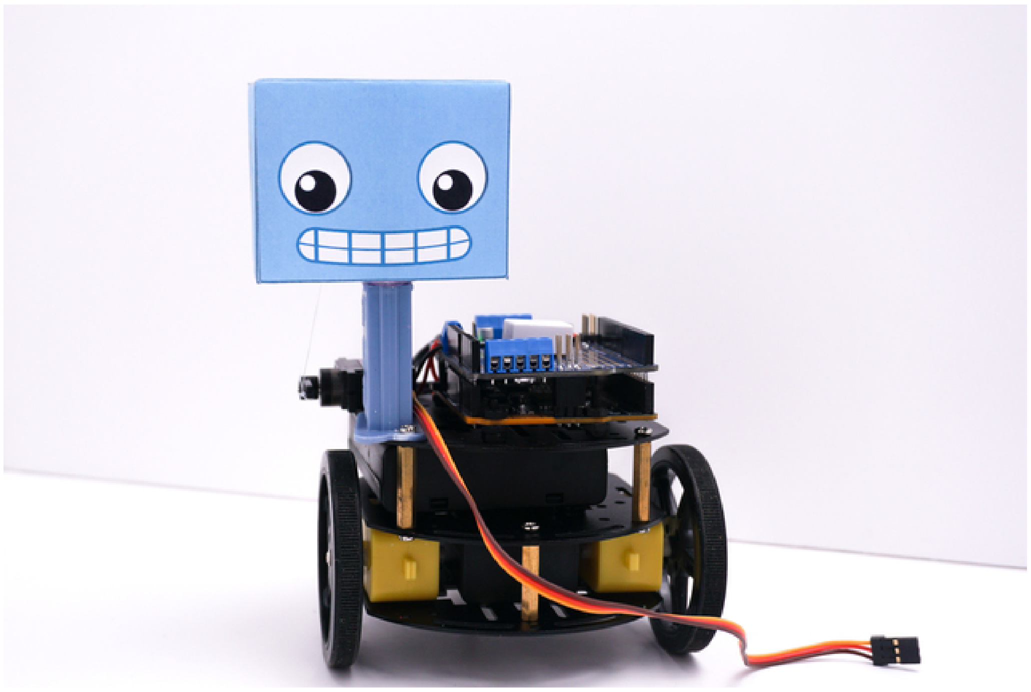






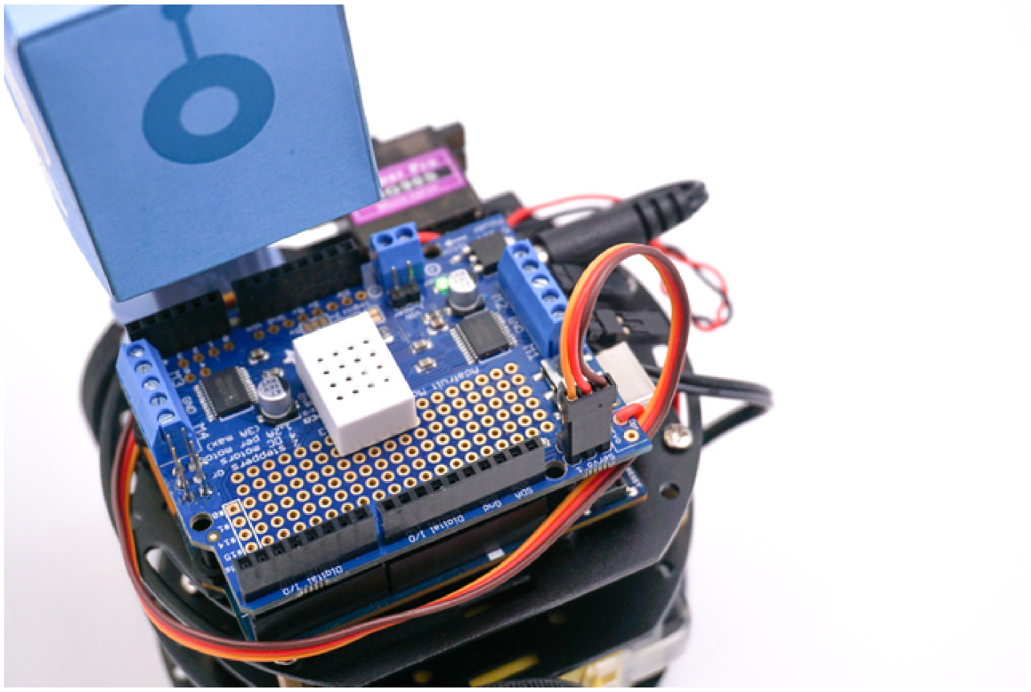
https://adafru.it/usE

https://adafru.it/usE

Affix the Adabot head to the candy dispenser with some blue tack or foam tape.

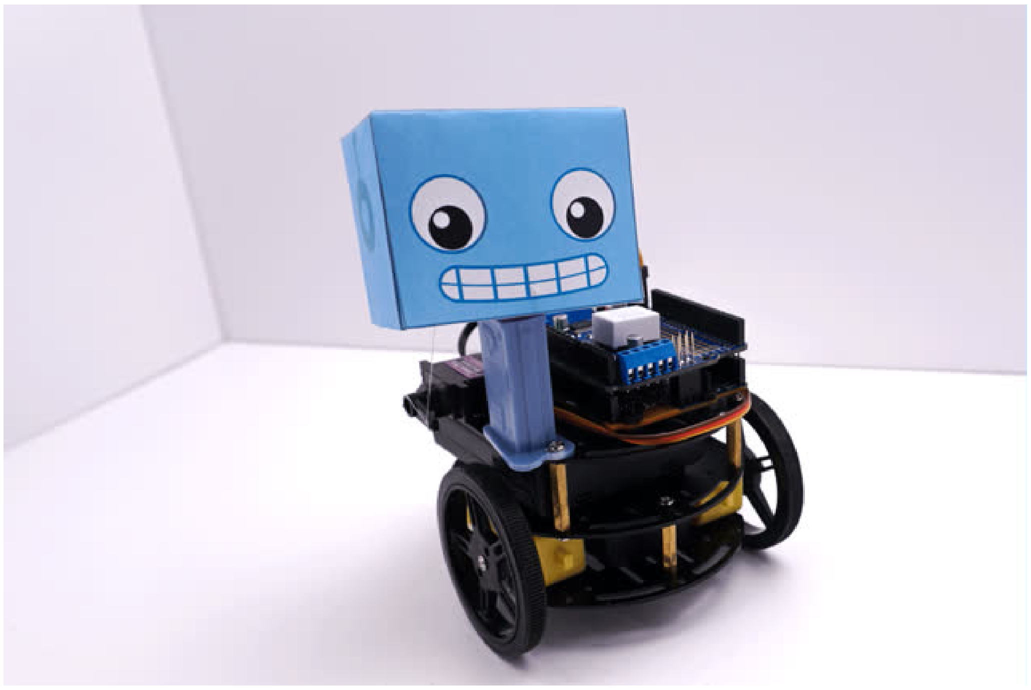


Plug the servo cable into the **Servo 1** port on the MotorShield.



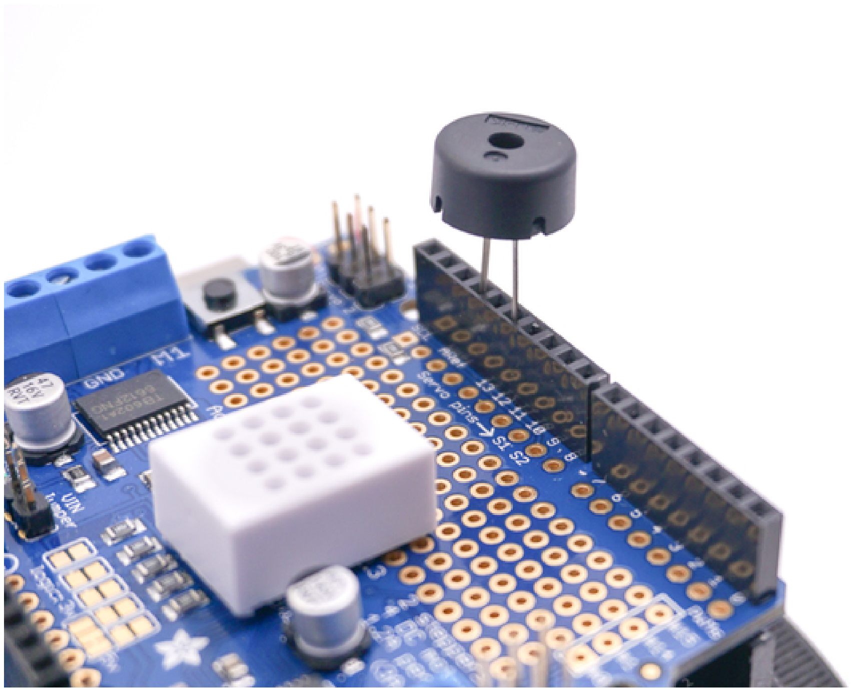
Open the **Ada_CurieBot_RC_CandyServo** sketch in the Arduino IDE, and then upload it to CurieBot.

Now, when you launch the control pad in Bluefruit LE you can press the "1" button to rotate the servo and serve candy!



Sing a Song

CurieBot loves to sing! You can have her drive up and serenade someone by adding a piezo buzzer and some software.



Open the **Ada_CurieBot_RC_Song** sketch in the Arduino IDE and upload it to your robot.

In this code snippet you can see how you'll use the `tone()` command to make music. The `melody[]` array and `noteDurations[]` array are used to describe the pitch and time value of each note.

CurieBot will play "Shave and a Haircut" by default, but you can change the song to anything you like, or even add other tunes, each one controlled by a different number on the Bluefruit LE app.

```
      //music
      if (buttnum == 2){
        int melody[] = {262, 196, 196, 220, 196, 0, 245, 262 };
        int noteDurations[] = { 4,8,8,4,4,4,4,4 };
        for (int thisNote=0; thisNote < 8; thisNote++){
          int noteDuration = 1000 / noteDurations[thisNote];
          tone(12, melody[thisNote], noteDuration);
          int pauseBetweenNotes = noteDuration * 1.30;
          delay(pauseBetweenNotes);
          noTone(12);
        }
      }
```

Now, use the Bluefruit LE app to drive CurieBot, and then whenever you press the "2" button, CurieBot will play a song!

Now that you've built your own robot friend, what kinds of modifications to hardware and software will you make next?