MAKE THIS! ELECTRONICS PROTOTYPING using ARDUINO

INTRODUCTION

Welcome to the CHI course *Make This! Electronics Prototyping using Arduino*.

The course comprises two 80-minute sessions. In the first session, you'll learn about electronics prototyping by lighting LEDs and controlling them using a few digital and analog sensors.

In the second, you'll build a small paper robot to see how a microcontroller, software, sensors and actuators work together on a single project.

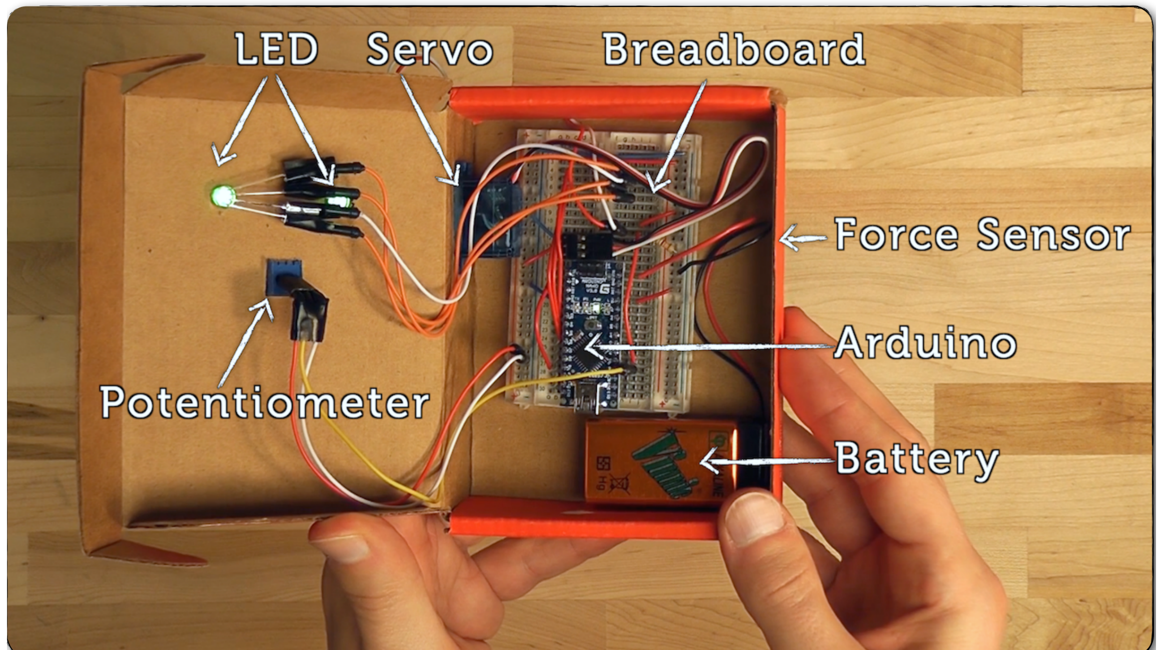
We'll cover Parts I & II of this guide today. Try Part III if you're experienced.

I: Electronics Prototyping
II: Paper Robot
III: The Advanced Section
IV: Make (More Than) This!

David Sirkin[1], Nik Martelaro[2], Wendy Ju[3]
[1]Stanford, [2]Accenture Labs, [3]Cornell Tech

PAPER ROBOT

Here's the paper robot project that you'll be building in the second half of the session.



MAKE THIS! ELECTRONICS PROTOTYPING using ARDUINO

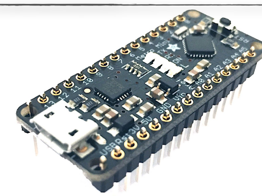
PART I – ELECTRONICS PROTOTYPING

1. Your Kit
2. Arduino
3. Breadboard
4. LEDs
5. Sensors
6. Voltage Dividers

ELECTRONICS PROTOTYPING KIT

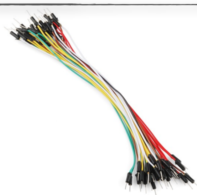
To get started, take out the first 4 items listed below: breadboard, Arduino, USB cable and jumper wires.

Breadboard: Wires plug into its sockets.

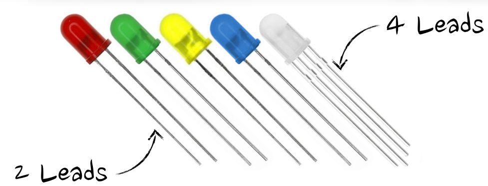


Arduino Metro Mini: Microcontroller.

USB Cable: A-type to micro-B connector.



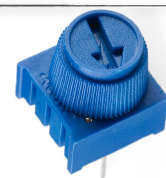
Jumper Wires: To connect components.



LED: Red, Green, Yellow, Blue and RGB.

RC Servo: Motor holds at a set position.

Button: Normally-open momentary type.



Potentiometer: Resistance of 10K Ohm.

FSR: 250-30K Ohm force-sense resistor.

Photocell: Changes resistance with light.

Battery & Clip: To power the Arduino.

Box: We use it for the body of the robot.

INSTALL THE ARDUINO SOFTWARE

If you haven't already done so, download the Arduino software now.

1 Install Arduino

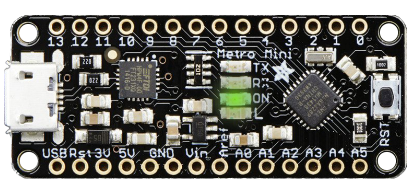
Visit the Arduino Software page and download Arduino IDE version 1.8.5 for your OS.

http://arduino.cc/en/main/software

Unzip the download, making sure to preserve the structure of any folders. Copy the application to a location on your system that you prefer.

You can try the Web Editor if you like, or have trouble installing the drivers.

Connect the Arduino to your laptop using the USB cable. The green LED on the top of the Arduino should light.



2 Install the Drivers

You need to install drivers to program the Arduino. Drivers convert data sent through the USB port into serial signals that the Arduino can understand.

There are 2 drivers to install, one for the USB control chip (made by FTDI), and one to make sure the port shows up correctly.

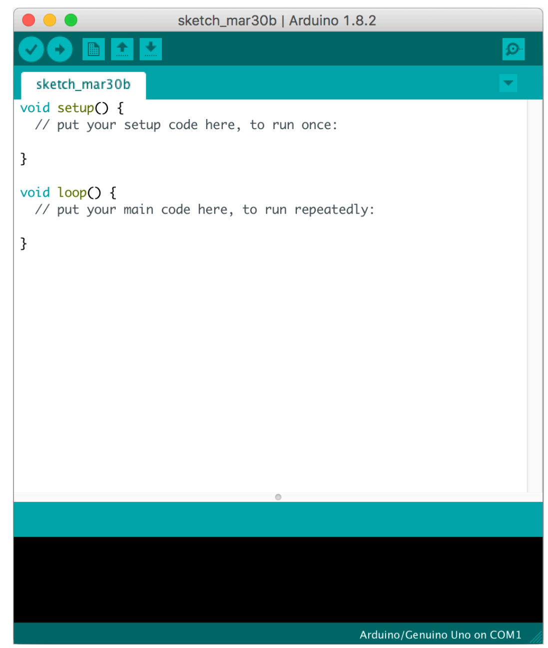
1: FTDI VCP Driver
http://www.ftdichip.com/Drivers/VCP.htm

Scroll down the page to find the driver for your OS. Ignore the Virtual COM Port Drivers at the top of the page.

2: SiLabs Driver
http://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

3 Launch Arduino

Launch the Arduino application. It will open to the template of a new, empty sketch. Here's what it looks like.



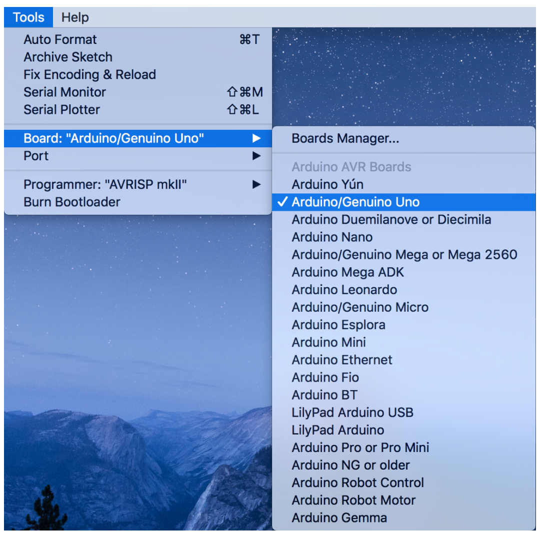
SETUP THE ARDUINO SOFTWARE

Now we'll set up the software to work with the Arduino Micro board.

1 Set the Board

The programmer needs to know what hardware you're using, so select the Arduino Micro using the menu bar.

Tools > Board > Arduino/Genuino Uno



2 Set the Port

Next, select a port for the serial device to communicate through.

Tools > Serial Port

On Windows, the port to choose will typically be COM3 or higher.

On macOS, the port to choose will resemble /dev/cu.SLAB_USBtoUART.

To check that you've chosen the right port, disconnect the Arduino and reopen the Serial Port menu.

The entry that has disappeared is the correct port. Reconnect the board and select that serial port.

3 Interface

Do you see the 5 buttons at the top left of the interface?

Hover your mouse over each of these buttons to highlight its function.

In particular, Verify (the check mark) compiles sketches. Upload (the arrow) first compiles sketches, and then uploads them to the Arduino.

Since Upload does both jobs, you can use it (alone) to save compiling twice.

Comments and Notes are in Callout Boxes

ARDUINO'S "HELLO WORLD!"

Open, compile, upload, then edit the Blink sketch to start learning how to program Arduino.

1 Flash "Blink"

Open the Blink sketch example by navigating through the top menu bar.

File > Examples > 01.Basics > Blink

Compile and upload the sketch, using the arrow button. After a few seconds, the yellow RX and TX LEDs on the Arduino should flash quickly.

If the upload is successful, the message "Done uploading" will appear in the status bar at the bottom.

You should see the red LED on the top of the Arduino repeatedly blink on for 1 second, then blink off for 1 second.

2 Update "Blink"

Try to modify the Blink sketch so that the LED flashes at twice its original rate.

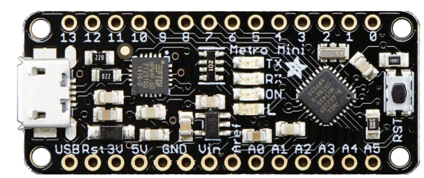
If you're not sure how to change the flashing rate, find the line below, and edit it so the delay is shorter. Then re-upload your modified sketch.

```
delay(1000); // wait for a second
```

When you can readily change the way that the LED flashes, you're ready to move on with the tutorial.

Keep the Blink sketch open for now.

The brain of the Arduino is its micro-controller. It has a similar role as the CPU in your laptop, only it's designed to be used with sensors and actuators. The Metro Mini micro-controller has a 16 MHz ATmega328, made by Atmel.



Arduino programs are called sketches. Sketches are just C++ programs. Every sketch includes 2 special functions: `setup()` and `loop()`.

`setup()` runs once at the start of the program, and `loop()` runs forever after that.

BREADBOARD WIRING OVERVIEW

Your breadboard has a few regions: power and ground rails on the sides, and circuit-building areas in the center.

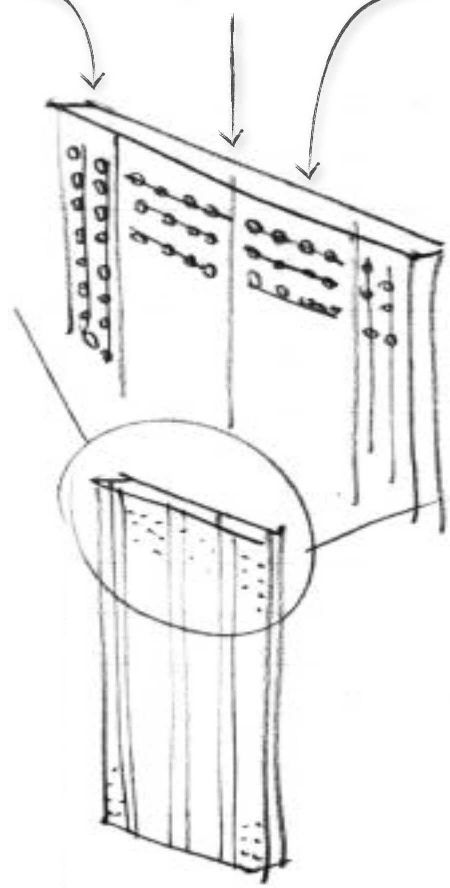
Power & Ground Rails

The holes along any **column** (either red or blue) are connected to every other hole in that column (only!).

The rails marked by **red** stripes are used for power, and the rails marked by **blue** are used for ground.

The Arduino receives power from your laptop through the USB cable. It then makes that power available to circuits on your breadboard through its 5V, 3.3V and GND pins.

Separator



Circuit Building Area

The holes along any **row** are connected to every other hole in that row (only!).

The separation that runs down the middle of the board isolates the rows on the left side from those on the right.

When changing any circuit on the breadboard, it's a good idea to unplug the USB cable from the Arduino, so you don't power something by mistake, so **unplug it for now**.

SETUP THE BREADBOARD

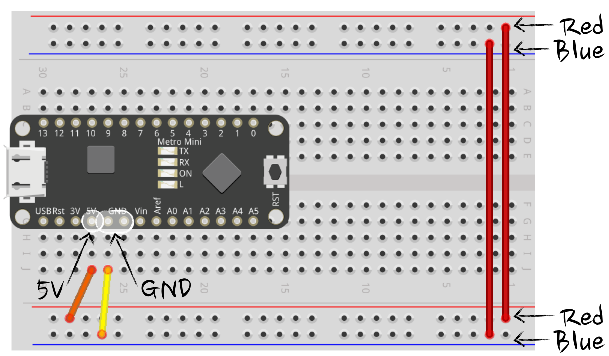
Now we'll configure the board for convenient access to power and ground through the rails.

1 Socket the Arduino

Unplug the USB cable from the Arduino, then socket it into your board as shown, with the USB port facing the outside.

Make sure it straddles the separation that runs down the middle of the board.

If it doesn't, then the pins on one side of the Arduino would connect to those on the other side, mixing their signals.



2 Wire the Board

A: Using 2 long jumper wires, connect the power and ground rails across opposite sides of the board, as shown on the right side of the image. Red connects to red, blue to blue.

Note: The wires may be red or some other color. Pay attention to the rails.

B: Using 2 short jumper wires, connect the Arduino's 5V pin to the red power rail, and either of the Arduino's 2 GND pins to the blue ground rail.

Together, these connections create a convenient power supply along both rails for your circuits.

Double-check your wiring against the image on the left to make sure that you got everything right.

3 Power the Board

Now plug the USB cable back into the Arduino. You should see the green power LED on the Arduino turn on.

If you don't, you have a short circuit, and should disconnect the USB cable and fix the circuit right away!

If everything looks good, you're up and running, and ready to light some LEDs.

A short circuit is when the GND and 5V pins of the Arduino connect together, and it can destroy the Arduino. :-(

We recommend you unplug the USB cable every time you change a circuit.

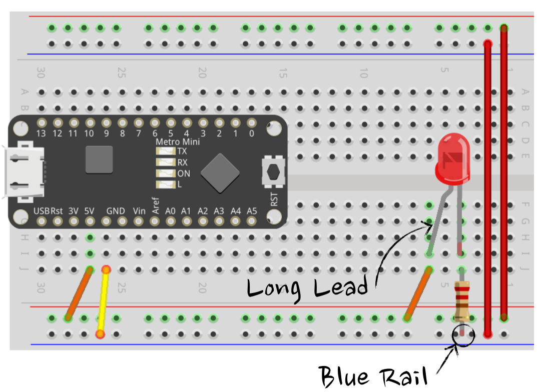
BASIC LED CIRCUITS 1: LED Always On

Let's build 3 LED circuits: **1) LED always on**, 2) LED control by button, and 3) LED control by program.

1 Add a Resistor

First, unplug the USB cable from the Arduino, then build the circuit shown below. Use a 220 Ohm resistor.

Resistors have 4 color bands to indicate their resistance. For 220 Ohm resistors, the bands are: **red**, **red**, **brown** and (farther away) **gold**.

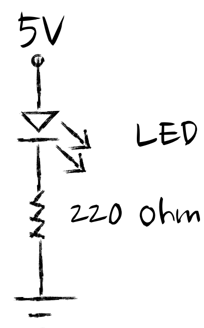


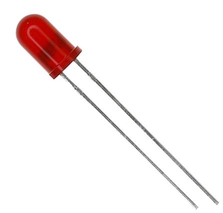
2 Add an LED

LEDs are diodes, and diodes only work (as intended) when oriented one way.

Connect the longer lead (the anode) toward power, and the shorter lead (the cathode) toward ground. In the diagram at left, the longer lead has a bent knee.

Note: **The long arm reaches for power!**





Diffuse Red LED

3 Power the Arduino

Now plug the USB cable back into the Arduino. Your new LED should light up.

Notice that the onboard LED is still flashing. That's because the Arduino is still running the Blink program from earlier.

It's important to always place a resistor in series with any LED, to limit the current in the LED to a safe value.

We include a circuit diagram alongside each drawing. Circuit diagrams are readable and portable, so it's worth being (somewhat) familiar with them.

BASIC LED CIRCUITS 2: LED Control by Button

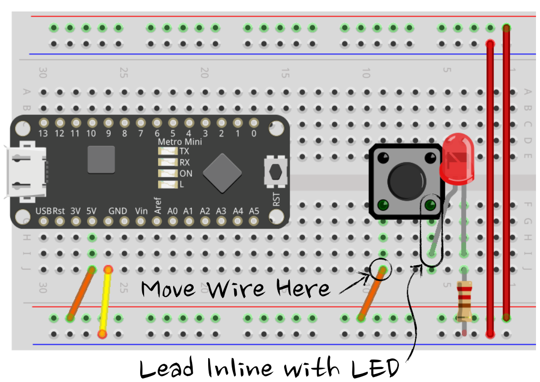
Let's build 3 LED circuits: 1) LED always on, **2) LED control by button**, and 3) LED control by program.

1 Add a Button

Unplug the USB cable from the Arduino, then insert a pushbutton into the circuit, as shown below. Press the button's leads fully into the board.

Make sure that the button crosses the center separator, as shown on the right.

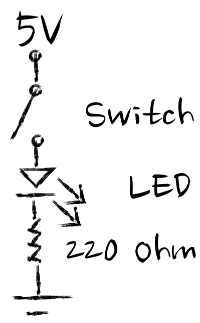
Note that you have to move the short wire that connected the LED to power.



2 Power the Arduino

Now plug the USB cable back into the Arduino, and check that the button works: the LED should light only when you press the button.

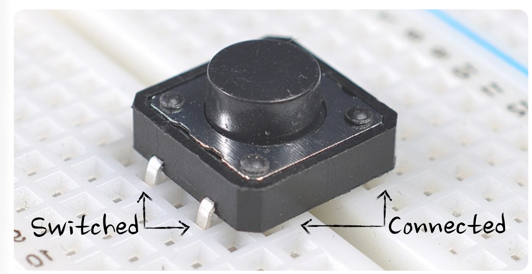
Congratulations, you've made a light switch.



12mm Pushbutton

If your button doesn't work, check that it's oriented as shown below. You might need to rotate it by 90 degrees.

A typical pushbutton, when pressed, connects the 2 pins on one side to the 2 pins on the other side.



BASIC LED CIRCUITS 3: LED Control by Program

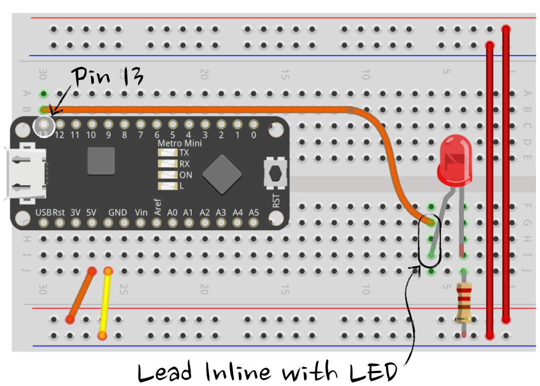
Let's build 3 LED circuits: 1) LED always on, 2) LED control by button, and **3) LED control by program**.

1 Rewire the Board

Unplug the USB cable from the Arduino, then rewire the board shown below.

First remove the button and the short wire that powered the circuit before. Then connect a long jumper wire to Arduino pin 13. This pin will now power the LED.

Reconnect the USB cable. The external LED should flash with the onboard LED.



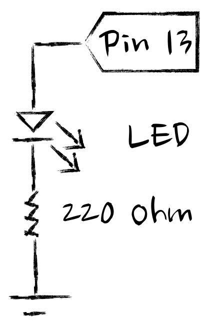
2 Revisit "Blink"

Return to the Blink sketch that's open on your laptop. The following line sets pin 13 to output, or "source," voltage.

```
pinMode(13, OUTPUT);
```

And the following lines tell the Arduino to send 5V, and later 0V, out that pin.

```
digitalWrite(13, HIGH);
digitalWrite(13, LOW);
```



The Metro Mini's red onboard LED is internally connected to pin 13. You can also connect an external LED to the same pin: they'll behave the same.

Why is Blink still running? The most recently loaded sketch is stored in flash memory, so it's remembered even after you power down the Arduino.

FADING THE LED

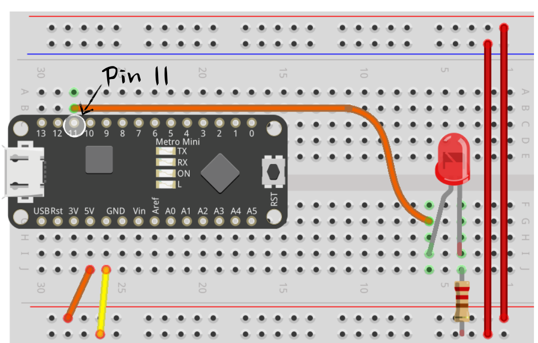
The last way to control an LED is to vary its brightness, using pulse width modulation, or PWM.

1 LED Brightness

What about those "breathing" LEDs during sleep mode on many laptops?

The fading light is done using pulse width modulation, or PWM. The LED is toggled on and off very quickly: say, 1,000 times per second. Much faster than your eye can follow.

The percentage of time that the LED is on (called the duty cycle) controls its apparent brightness.



2 Rewire the Board

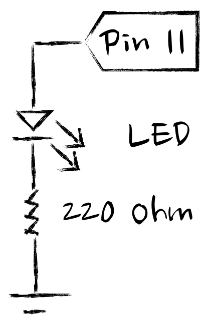
Update your circuit so the jumper wire connects to pin 11, as shown on the left. Then open the Fade sketch.

File > Examples > 01.Basics > Fade

The first line (after the comment) sets the variable named `led` to pin 9.

```
int led = 9;
```

But your LED is connected to pin 11. Edit the line above to set `led` to pin 11.



3 Flash "Fade"

Now upload your sketch and check that the LED fades on and off.

Try changing a few parameter values to better understand how things work. Start with the last line, for example.

```
delay(30);
```

To control an LED using PWM, you have to connect it to one of the pins that supports PWM (not all do!).

On the Metro Mini, these are pins 3, 5, 6, 9, 10 and 11.

You also have to use `analogWrite()` (which enables PWM) in your sketch, rather than `digitalWrite()`.

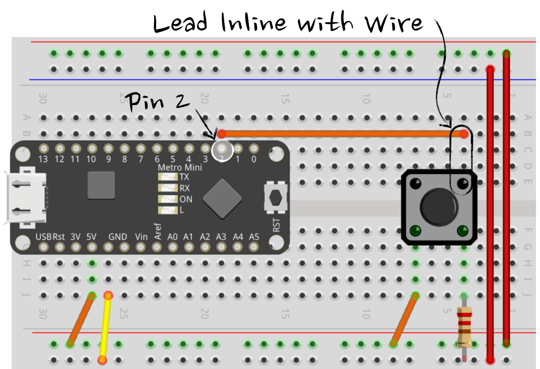
SENSING A BUTTON PRESS

The Arduino can detect different voltages, as well as provide them. Here's how.

1 Replace the Button

It's just as easy for your Arduino to sense input as it is to control output. Wire your board with the button circuit shown below.

Check that the wire that connects from the button to the Arduino is on pin 2.

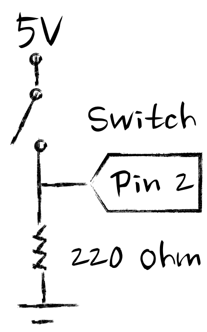


2 Flash "Button"

Open the Button sketch, look over how it works for a moment, and then run it.

File > Examples > 02.Digital > Button

Make sure that the Arduino's onboard LED lights when you press the button.



It's okay to connect the jumper wire to the "opposite" side of the button: recall that the 2 leads on the same side are connected to each other.

Pins as Input or Output

In the earlier circuit, the LED turned on because the button directly switched a supply of 5V to it.

Here, pushing the button sends a 5V signal to pin 2. The sketch reads this, and sends 5V to pin 13, which is attached to the onboard LED.

Any Arduino pin can be made an input (or an output) by setting its mode in the `setup()` function, like the following.

```
pinMode(buttonPin, INPUT);
```

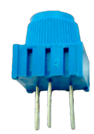
The resistor is required for the button circuit to work. Without it, closing the switch would create a short circuit (!).

ANALOG SENSORS: Potentiometer

Buttons have only 2 states: open and closed. A potentiometer provides a continuous range of values.

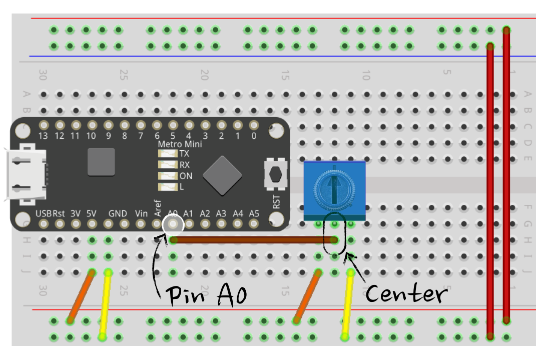
1 Add a Potentiometer

A potentiometer (or “pot”) is really just a variable resistor. It includes 3 pins.



Find the potentiometer in your kit and rewire your board as shown below.

Using a long jumper wire, connect the potentiometer’s center pin to the Arduino’s analog input pin labeled AO. Connect 1 outer pin to the power rail, and the other to the ground rail.

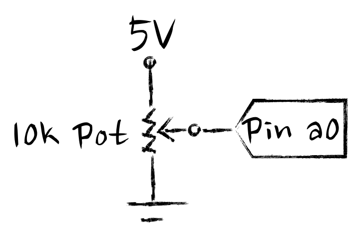


2 Flash “AnalogInput”

Open the AnalogInput sketch, look over how it works for a moment, and then upload it.

File > Examples > O3.Analog > AnalogInput

At first, the onboard LED should flash. As you rotate the potentiometer’s cap, the LED should flash at different rates.



For this potentiometer, the resistance between the 2 outer pins is a constant 1OK Ohms.

However, the resistance between either of the 2 outer pins and the center pin varies as you turn the cap.

For more about what input the Arduino can sense (only voltage), and how a potentiometer (which varies resistance) works with it, read the page on Voltage Dividers at the end of Part I.

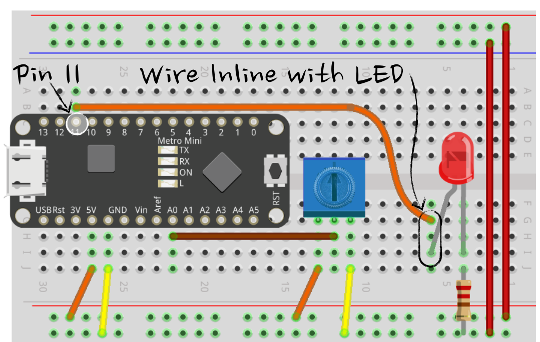
ANALOG SENSORS: Potentiometer & LED

Now that you can read a signal from a potentiometer, lets use it to control an external LED.

1 Add an LED

Without removing the potentiometer circuit, connect a long jumper wire from an external LED to Arduino pin 11, as shown below.

Notice that we're just combining the circuits from a) Fading the LED and b) Analog Sensors: Potentiometer.

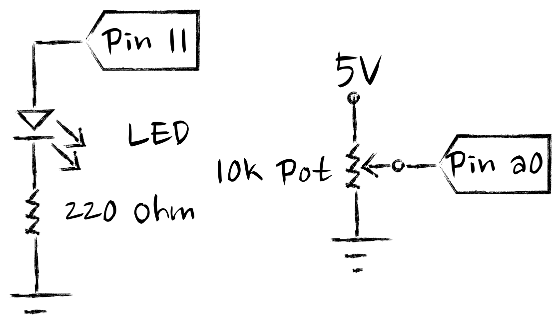


2 Update "AnalogInput"

Update the AnalogInput sketch to flash the external LED that you just wired to pin 11, rather than the onboard LED, which is connected to pin 13, then upload the modified sketch.

If you need a hand, here's the line that you should edit.

```
int ledPin = 13; // select the pin
```



Analog sensors are among the most useful in electronics. Other common types include photocells, proximity sensors, force-sensing resistors (FSRs), accelerometers, gryoscopes and ultra-sonic rangefinders.

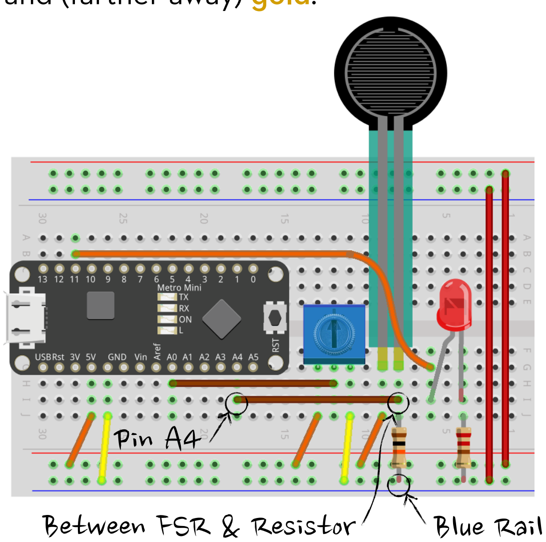
ANALOG SENSORS: FSR & Photocell

Let's explore 2 other analog sensors, both of which are used on real-world robots.

1 Try an FSR

Now is a good time to try the force-sensing resistor (FSR). It looks like this.

Leave the potentiometer and LED in place, and add a 10K Ohm resistor and jumper wires as below. The 10K Ohm color bands are: **brown**, **black**, **orange** and (farther away) **gold**.



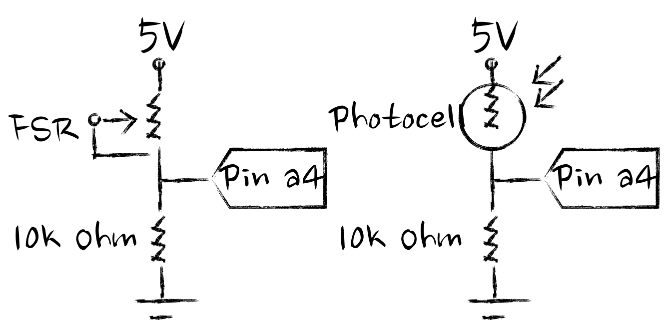
2 Update "AnalogInput"

Update the AnalogInput input sketch to read the value from the FSR, which you just connected to pin A4.

If you need a hand, here's the line that you should edit.

```
int sensorPin = A0; // select the pin
```

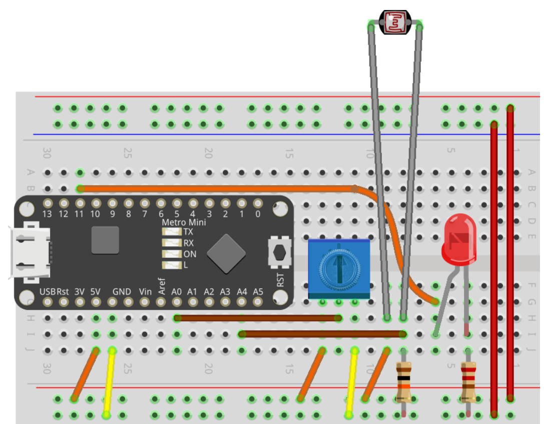
Like a potentiometer, the FSR and photo cell change resistance, just when the pressure or ambient light changes.



3 Try a Photocell

Find the photocell (also called a photo resistor) in your kit. It looks like this.

Just replace the FSR with the photocell. You can continue to use the same 10K Ohm resistor and jumper wires.



VOLTAGE DIVIDERS

One of the most fundamental and useful circuits in electronics!

Why You Need Them

The Arduino's pins can only input or output voltages.

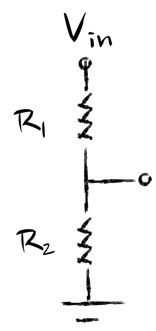
So how did we use sensors, such as a potentiometer, FSR and photocell, which only vary resistance?

Because it's easy to transform varying resistance into varying voltage, using a voltage divider circuit. Here's how...

How Do They Work?

A basic voltage divider has 2 resistors, connected in series, between power and ground, as shown below.

One of these (which is your sensor) has varying resistance, while the other has a fixed resistance.



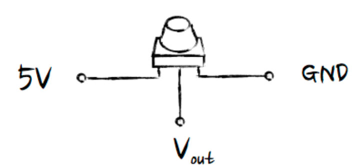
$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

At the point between the 2 resistors (V_{out} on the left), you can measure how much the voltage dropped as it passed through the first resistor (R_1).

The voltage at V_{out} changes as the ratio between the 2 resistances changes (by the equation shown at left).

You can then connect V_{out} to one of your Arduino's analog input pins. On the Metro Mini, these are pins A0–A5.

A potentiometer includes **both** R_1 and R_2, so the connection will look like this.



MAKE THIS! ELECTRONICS PROTOTYPING using ARDUINO

PART II – PAPER ROBOT

1. Servos
2. Robot Face
3. Batteries
4. Arms & Body

SERVOS 1: Making Things Move

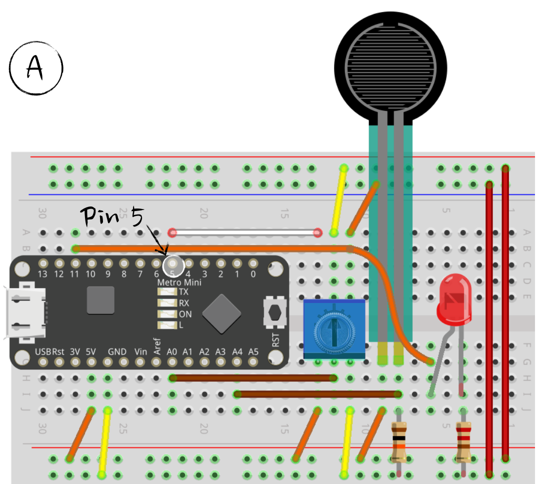
When you're ready to move on from flashing lights to moving things.

1 What's a Servo?

A servo is a DC motor, geartrain, potentiometer and feedback circuit, all in a single housing.

By sending a PWM signal from your Arduino to the servo, you're telling it what angular position you'd like it go to.

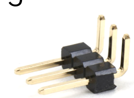
The potentiometer tells the feedback circuit the servo's current position, and the circuit drives the motor to match the desired position.



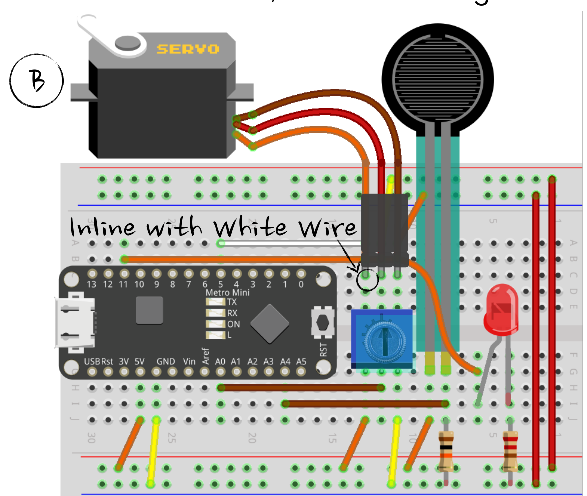
2 Add a Servo

A: Connect 3 jumper wires as shown on the left, with the white wire to pin 5: this wire carries the signal.

B: Connect the servo to your bread board, by snapping off a 3-pin segment from a right-angle breakaway header.



Socket the header as follows. You want white on the left, black on the right.



3 Flash "Sweep"

Now open the Sweep sketch.

File › Examples › Servo › Sweep

Edit the following line to attach the servo to pin 5, instead of pin 9.

```
myservo.attach(9); // attach the servo
```

When you upload the sketch, your servo should be sweeping back and forth, by about 180 degrees.

Change some parameters in the sketch to make the servo sweep slower, or over a smaller angle.

For example, editing the following line will alter the sweep speed.

```
delay(15); // waits 15ms
```

SERVOS 2: Controlling Movement

We really want to control servos using sensors. Let's start with a potentiometer.

1 Update "Knob"

Open the Knob sketch.

File > Examples > Servo > Knob

We'll be changing 2 lines here, because we used different pins for our circuit.

A. Edit the following line to set the `potPin` variable to A0, rather than 0.

```
int potpin = 0; // analog pin
```

B. Edit the following line to attach the servo to pin 5, rather than pin 9.

```
myservo.attach(9); // attach the servo
```

2 Flash "Knob"

Now upload and run the Knob sketch, then turn your potentiometer's cap. If all goes well, the servo should turn in sync with the potentiometer's cap.

You can turn the servo the opposite way by updating the following line.

```
val = map(val, 0, 1023, 0, 180);
```

This line maps possible analog input values (which range from 0-1,023) to possible servo angles (which range from 0-180). Try the following instead.

```
val = map(val, 0, 1023, 179, 0);
```

Why aren't we using the default pins in the example sketches?

Arduino's current servo library disables PWM on pins 9 and 10. However, these are the pins that the examples use.

As a result, we've changed our circuit to use other pins instead.

There are 2 common types of servo: standard and continuous. A continuous servo doesn't go to a desired position. Instead, the signal tells it what speed and direction to (continually) turn.

THE ROBOT'S FACE 1: Using LEDs as Eyes

We recommend using 2 LEDs for the robot's eyes, and the potentiometer for its nose.

1 Add a 2nd LED

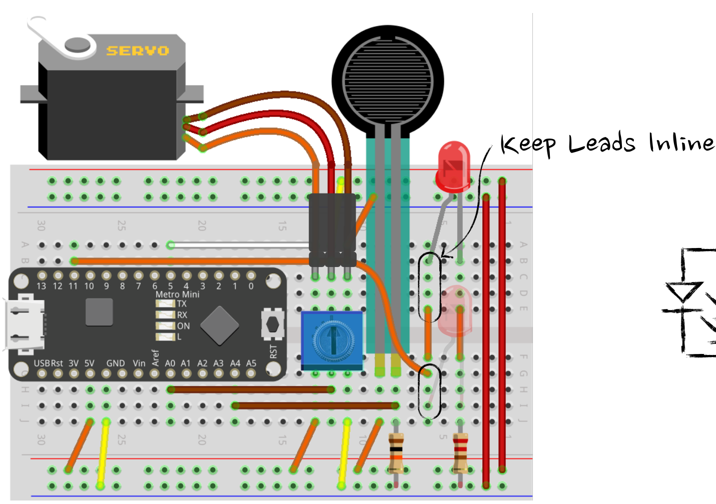
Let's add a second LED in parallel with the first.

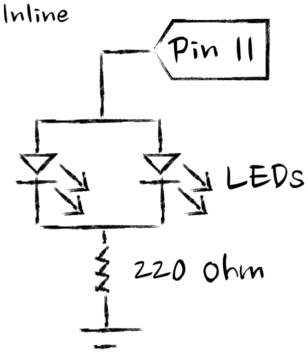
First, add 2 short jumper wires to cross the separator down the center of the breadboard, as shown below.

Second, orient the second LED's leads the exact same way as the first LED's, and socket it into the same rows of the breadboard (just on the other side).

Since you've connected 2 rows across the separator, the LEDs will receive the same signal, and both should light up.

With the breadboard set at the bottom of your box, you should now be able to position the LEDs to poke out the top of the box, as the robot's eyes.





LEDs in Series & Parallel

LEDs can be connected in series or in parallel. Wired in series, the supply voltage gets divided among the LEDs, so each LED will appear dimmer.

Wired in parallel, each LED draws the full supply voltage, so they appear the same brightness (but the power supply will drain faster as a result).

THE ROBOT'S FACE 2: Moving Sensors Off-Board

Sensors are often more useful located away from the main circuit board, such as a robot's face or arm.

1 Breadboard Layout

As you build projects with many lights, sensors and actuators, planning the layout of your breadboard beforehand can be a great help.

But you may eventually need to move LEDs, potentiometers, FSRs and photocells off the breadboard, so you can attach them to your project wherever they're needed.

2 Long Jumper Wires

The shorter, right-angle jumper wires have a core of solid wire. They hold their set shape well, but aren't flexible.

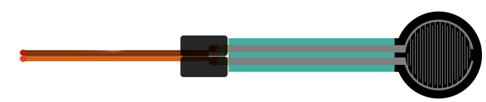
The longer jumper wires have a core of flexible, stranded wire, making them great for connecting to components off the breadboard.

Find the male-female jumper wires in your kit. They look like this.

3 Move the Sensors

Split off a 2-wire pair and a 3-wire triple. Try to keep each of them joined over its length, to keep things tidy.

Remove the FSR from your bread board. Attach one end to the 2-wire pair, and socket the other end back into the breadboard (in the same place as before). Do the same for the potentiometer and the 3 wires.



You may want to use some Blu-Tack adhesive, or electrical tape, to hold the sensors in place against the wires.

POWER: USB & Batteries

The electronics are almost done. Let's untether the electronics from your laptop.

About the USB Cable

Up to this point, your Arduino has been connected to your laptop through a USB cable.

The cable does 2 things: 1) it allows the laptop and Arduino to communicate, and 2) it powers the Arduino.

Many do-it-yourself (DIY) projects are meant to run standalone, without being connected to a laptop. For that, we use batteries.

Arduino sketches reside in flash memory, so once a sketch is uploaded, it can run without communicating with your laptop..

1 Add a Battery

We've had many students injure their Arduino when connecting a battery (!). Please follow these 3 steps carefully.

A: Find the 9V battery in your kit and snap on the cap with red & black leads.

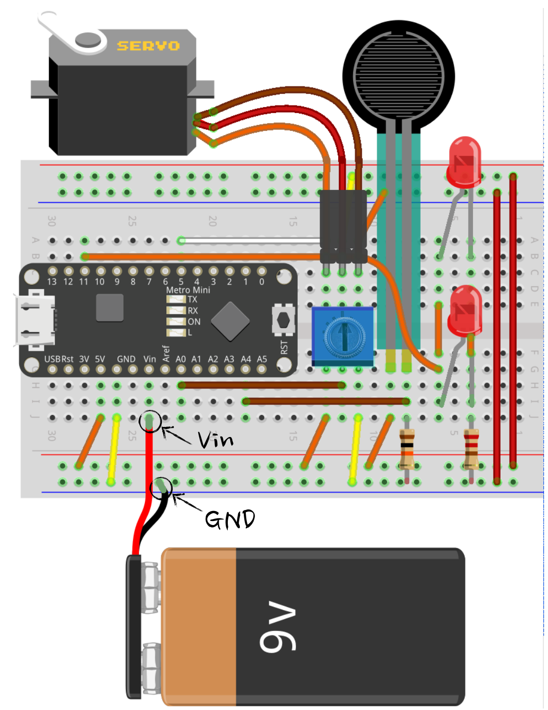
B: Connect the black (ground) lead to either the ground rail or one of the 2 GND pins on the Arduino.

C: Connect the red (power) lead to the Vin (voltage in) pin. Only this pin has a regulator, which safely translates 9V down to 5V.

Your project should start running again, with no tether to the laptop.

Final Breadboard Layout

Here's the final layout of your robot's electronics.



THE ROBOT'S BODY

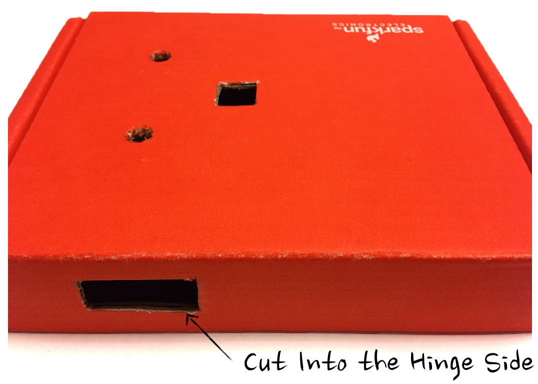
It will house the breadboard circuit, off-board components and battery.

1 Place Openings

Trace out the shape of the 2 LEDs, potentiometer and servo on the box's top and side, and cut them out with a utility knife.

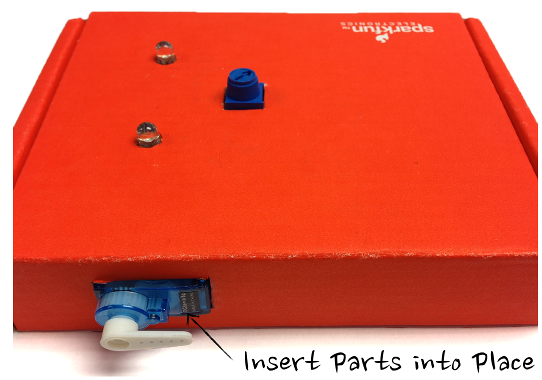
2 Cut Openings

Cut the servo into the hinge side of the box! Otherwise you have to cut through 2 layers of cardboard (including the flap) instead of 1.



3 Attach Parts

Position and tape or glue components in place. Attach them from inside of the box to keep the outside clean.



THE ROBOT'S ARMS

Both arms work together: one supports the FSR, the other attaches to the servo.

1 Cut Out 2 Arms

The robot's arms can be cut from construction paper, cardboard or foam sheets. Here's a sample pattern you might follow.

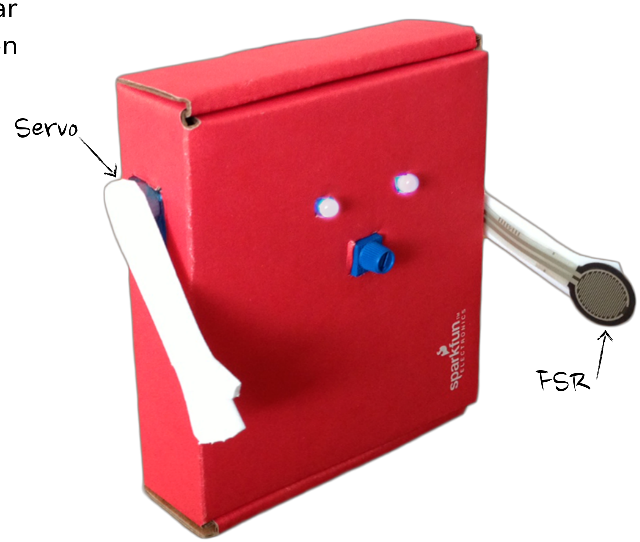
To attach the cutout arm to the servo, snap a white plastic arm (from the clear plastic bag) onto the servo's hub, then tape or glue the cutout arm to that.

2a Make an Arm Move

There are 2 approaches to making an arm move. The quickest approach is to reuse the Sweep or Knob sketches from the start of this section.

Keep in mind that Sweep constantly moves the arm (unless you edit it), and Knob requires the potentiometer.

The fancier approach is to interactively shake hands, described next...



2b Shake Hands

If you want the robot to shake hands, follow these 3 easy steps.

A: Bring the FSR to the outside of the box, either through the top of the lid, or an opening you cut along one side.

B: Affix (tape or glue) the FSR to the arm that **is not** attached to the servo, positioning the round pad at the hand.

You can affix the FSR to the arm that **is** attached to the servo, but wiring can be a challenge, since the arm moves.

C: Upload the robot control sketch we provide in person, or from our GitHub repository:

https://github.com/CDR-IxD/makeThis-CHI

This sketch reads the FSR value, and moves the servo. Voilà.

MAKE THIS! ELECTRONICS PROTOTYPING using ARDUINO

PART III – THE ADVANCED SECTION

1. Fading LEDs
2. RGB LEDs
3. Control Sketch

ANALOG SENSORS: Potentiometer, LED & Fading

We can use the potentiometer to fade an LED's brightness, rather than to change its flash rate.

1 Edit "loop()"

In the AnalogInput sketch, turning the potentiometer changes the flash rate. What if we want to change the LED's brightness instead?

We need to edit the sketch so that the value read from the `sensorPin` is used to PWM the `ledPin`.

Try to figure this out on your own, referring back to the Fade sketch.

Or, it'll be quicker to look over how the code here works, then paste it instead of the original `loop()` function.

```
void loop() {
  // read the analog value from the sensor
  sensorValue = analogRead(sensorPin) / 4;

  // write the value out to the led
  analogWrite(ledPin, sensorValue);
}
```

2 Divide by 4

We divide `sensorValue` by 4 because the Arduino's analog input has 4 times the resolution of its PWM output.

Without dividing, we'd only use 1/4 of potentiometer's rotation range to fully light the LED.

Remove the division and try it!

THE ROBOT'S FACE: About RGB LEDs

To use RGB LEDs for the robot's eyes, you need to add 2 more wires and update the sketch.

1 About RGB LEDs

The LEDs that you use can be uni-color (easier) or RGB (fancier). You already know how to connect a uni-color LED.

RGB LEDs have 4 (!) leads instead of 2. That's because the dome housing contains 3 lights (red, green and blue).

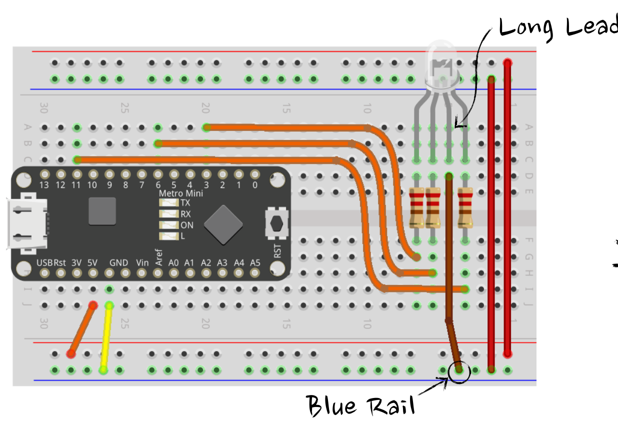
The long lead is a (common) cathode. Each of the others is the anode for one of the 3 colors.

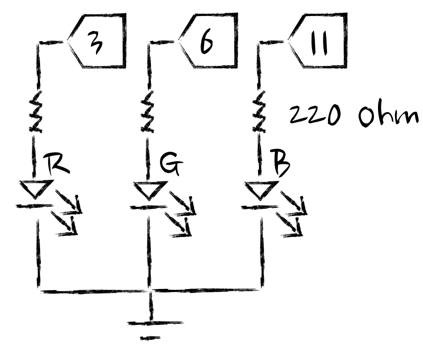
2 Add an RGB LED

Below is a typical way to connect an RGB LED, using a 220 Ohm resistor on each anode.

Connect wires from the 3 anodes to pins 3, 6 and 11, as shown. We chose these pins to fit with the other parts of the robot circuit, but any of the PWM pins will work: 3, 5, 6, 9, 10, or 11.

You can create almost any color by sending a different PWM signal to each light, and changing its brightness relative to the others.





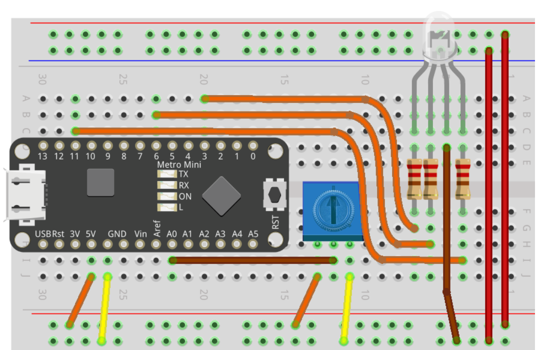
THE ROBOT'S FACE 2: Controlling RGB LEDs

Connect the potentiometer just as you did earlier.

1 Add a Pot

To control the RGB LED, connect the center pin of a potentiometer to Arduino analog pin AO, just as we did earlier.

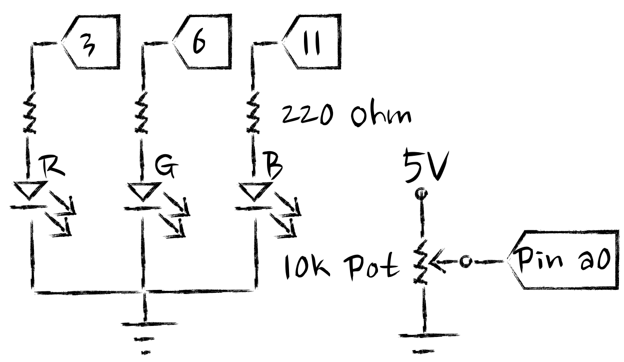
Remember to connect its 2 outer leads to the power and ground rails.



2 How to Fade Color

To fade an RGB (rather than uni-color) LED, you need to add 2 more output pins to your sketch, since there are now 3 colors.

Create a new sketch in the Arduino IDE, and paste in this sample code.



The idea is to vary the relationship between 3 parameters (red, green and blue) using only 1 degree of freedom (the sensorValue variable).

3 Flash this Sketch

If we change all 3 pins the same way, the LED will only change brightness. By changing 1 pin **relative to** the other 2, the LED will change color.

```
int potPin = A0;
int potValue = 0;

int redPin = 3;
int greenPin = 6;
int bluePin = 11;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  // read analog value from the sensor
  potValue = analogRead(potPin) / 4;

  // write analog value to the 3 LEDs
  analogWrite(redPin, 255 - potValue);
  analogWrite(greenPin, potValue);
  analogWrite(bluePin, potValue);
}
```

SAMPLE CONTROL SKETCH

This program: 1) fades LED brightness using the potentiometer, and 2) moves the servo using the FSR value.

1 Initialize Variables

The sketch has 2 analog input pins: AO for the potentiometer and A5 for the FSR, and 2 output pins: 9 for the LED and 5 for the servo.

```
// a sample paper robot control program
// the pot changes the led's brightness

// a servo shakes the robot's arm after
// an fsr is pressed and fully released

#include <Servo.h>

int potPin = A0;
int potValue = 0;

int ledPin = 9;

int fsrPin = A5;
int fsrValue = 0;

int pressValue = 32;
int releaseValue = 16;

boolean pressed;

Servo myservo;
int servoPin = 5;
```

2 Setup Pins & Servo

The setup function initializes the LED and servo pins as outputs (all Arduino pins default to input), and attaches the servo to PWM pin 5.

```
// Arduino pins default to INPUT mode

void setup() {
  pinMode(ledPin, OUTPUT);

  pinMode(servoPin, OUTPUT);
  myservo.attach(servoPin);

  // move the servo to its 0 position
  myservo.write(0);
  delay(5);
}
```

We've provided this sketch for you, in the RobotControl folder, so you don't have to copy-paste it from here.

3 Control LED & Servo

The loop function detects an FSR "press" then checks for a corresponding "release." When that arrives, it moves the arm forward and back.

```
void loop() {
  potValue = analogRead(potPin) / 4;
  analogWrite(ledPin, potValue);

  fsrValue = analogRead(fsrPin) / 4;
  if (fsrValue > pressValue) {
    pressed = true;
  }

  if (pressed) {
    if (fsrValue < releaseValue) {
      for (int pos = 0; pos <= 90; pos++) {
        myservo.write(pos);
        delay(5);
      }
      for (int pos = 90; pos >= 0; pos--) {
        myservo.write(pos);
        delay(5);
      }
      pressed = false;
    }
  }
}
```

MAKE THIS! ELECTRONICS PROTOTYPING using ARDUINO

PART IV – MAKE (MORE THAN) THIS!

1. Your Own Lab
2. Resources

SETUP A MAKE THIS! LAB

How you can continue electronics prototyping after this course.

Getting Started

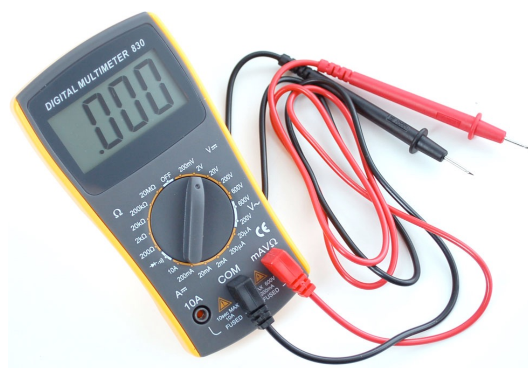
To start your own Make This! lab, you only need a computer and a prototyping board, like Arduino.

At some time soon, you'll probably also want a diagnostic instrument, external power supply and a few hand tools.

Multimeters (DMMs)

Regarding the diagnostic instrument, a digital multimeter (DMM) should probably be your first (and maybe only) purchase. It's a great tool to diagnose circuits, confirm resistor values, check a battery's power level, and more.

DMMs like those available during the course are inexpensive, starting at $10.



Better quality DMMs are $25–$50, and provide more precision, functionality, robust internal circuits and external housing and probes.

Professional quality DMMs are $100 and up, and usually sense capacitance, frequency and temperature.

OSCILLOSCOPES

Useful when you're working with analog signals, and a DMM becomes difficult to use.

Voltage Over Time

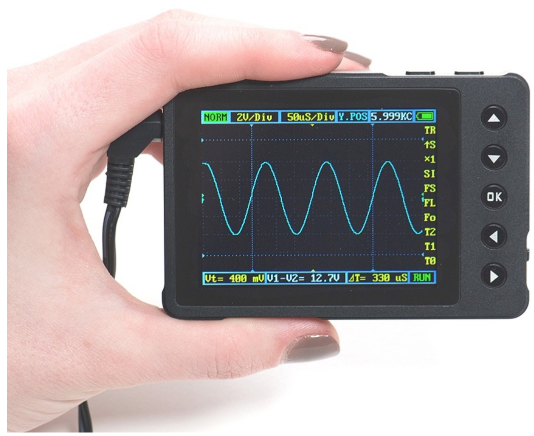
A DMM is all you'll need for a while. When will you need something else?

DMMs sense voltage, but not very well if that voltage varies over time: it's just a fluctuating number. To identify frequencies or waveforms (like PWM), you need an oscilloscope!

An oscilloscope's strength is in sensing periodic (sine or square) signals, making it well suited to analog input.

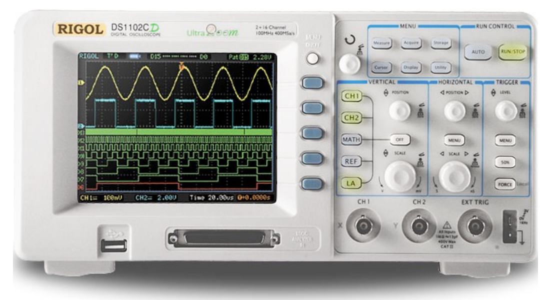
Inexpensive Scopes

A hand-held scope, such as the Nano from DSO, is an inexpensive place to start, for about $90. A limited feature set trades off low price and small size.



Also consider a PC-based scope. This includes a USB probe and software that runs on your computer, starting at $200.

Desktop scopes are useful when you work with analog signals a lot. Look for 20+ MHz bandwidth. Prices start at $400 and go up (way up!) from there.



POWER SUPPLIES

The 3 most useful types include batteries, AC wall adapters and tabletop supplies.

Batteries

Batteries let you run things on their own, without being tethered through a USB cable to your computer.

The Arduino Micro can run safely when powered from 5V–20V. That's a single 9V battery, a clip of 3–4 AA or AAA batteries, or even 2 camera or lithium ion batteries (connected in series).

Coin cell batteries are designed to source current slowly, at low levels. They'll work for several LEDs, but greater power demands will exhaust them quickly.

AC Wall Adapters

AC wall adapters are common in the 5V–12V range, which cost $5–10. They allow you run your project without a computer, and without batteries that run down.

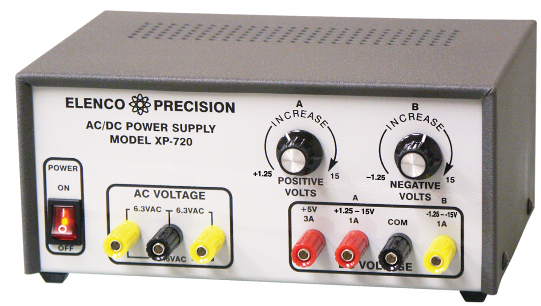
The tradeoff is that your project is now tethered to a power socket, which may be okay if it won't be moved frequently.

Tabletop Supplies

Tabletop supplies have 2-3 separate voltage outputs, and let you vary the amount of voltage sourced by each.

This can be useful when you're testing several circuits, especially those using high-power LEDs or DC motors.

A basic 3-output variable supply, like the Elenco XP-720, costs $65.



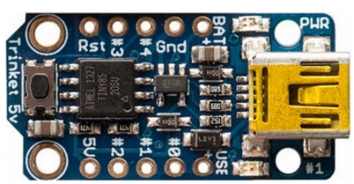
PROTOTYPING PLATFORMS

Arduino isn't the only microcontroller-based prototyping platform. Here are a few others.

Arduino Compatible

Arduino hardware and software are open-source, leading to a variety of available compatible boards.

There are too many to list here, but 2 of our favorites are the Teensy and Trinket. Teensy even makes boards with 32 bit ARM microcontrollers.



New boards are being developed all the time, so explore the alternatives. Start with Arduino's online store.

Other Platforms

Some prototyping boards use the Arduino's AVR microcontroller, but without the Arduino hardware or software. AVR Freaks has examples.

Another choice is the Microchip PIC. Microchip provides free (!) samples at your request, has starter kits with 8-32 bit architectures, and their own free programming environment, MPLab. The Microstick prototyping board starts at $25 and should look pretty familiar to you.

The BeagleBone Black by BeagleBoard, costs $45, which is more than Arduino, but it has a 1 GHz ARM processor, onboard microSD card, and runs Android or Ubuntu. Several add-on boards enhance its graphics, motor control and networking abilities.

If you feel limited by the Arduino software, you can compile sketches using avr-gcc without the IDE, or use an Arduino plugin for Eclipse.

COMPONENTS, SENSORS & TOOLS

Useful things to keep on-hand for when they're needed.

Basic Components

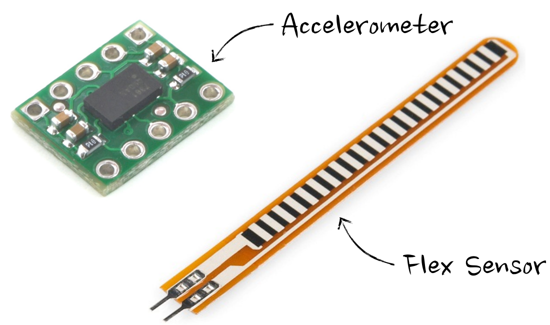
To make your lab more useful, keep a stock of components on hand.

The basics include resistors, capacitors, jumper wires and alligator clips. Look for the larger assortment packs.

Other frequently-used components are transistors (to switch large currents), voltage regulators (to connect to different power supplies), op-amps, 555 timer chips, and so many more.

Other Sensors

You've only used a small sample of sensors. Others that are common for prototyping include flex sensors, encoders (similar to a potentiometer), RFID sensors, accelerometers and gyroscopes, infrared or ultrasonic rangefinders, and even small cameras.



Wiring & Connecting

A few hand tools are also worthwhile. Some, such as a utility knife, are quite common. Consider these too.

Hot glue gun: Connect parts, fill gaps, and sets in seconds. Can be removed.

Wire clippers and strippers: To trim jumper wires and remove the sheathing.

Soldering iron: For stronger mechanical and electrical connections than crimps.

Perfboard: A base for soldering circuits together. Similar in use to a breadboard.

SOURCES OF COMPONENTS & IDEAS

Where to purchase components, sensors and hand tools. Where to find ideas about what to make.

Component Vendors

The websites of hobbyist vendors such as Sparkfun and Adafruit provide a sense of the components and custom breakout boards available for your projects, including product selection guides, discussion threads and video tutorials.

There are many other online vendors in countries around the world. Just start searching.

A breakout board is a custom-designed board that provides easy (breadboard) access to a small, special-function chip.

Examples include bluetooth, wifi, sensors, LCDs, even MP3 decoders.

Ongoing Education

Several online resources, including Instructables, Make and LifeHacker, feature tutorials about do-it-yourself projects that use the tools and materials covered in this course.

To leverage the graphics capability of your PC, and create interfaces to your electronics, download Processing and its Arduino library, and try a few sample programs. Highly recommended!

Maker Resources

To meet up with fellow makers in person, you can visit the Maker Faire, which hosts events around the world.

You can upload your projects and ideas for other people to browse, improve upon or purchase (!) at sites such as Ponoko and Shapeways. You can even have your own designs laser cut or 3D printed, then mailed back to you.