Physical Computing Inputs and Outputs ¬

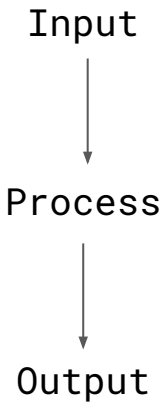
CCI / OCT 2021

Andy Sheen

What is Physical Computing?

In physical computing we build interactive systems that can sense the world and respond with actuators.

An intersection of electrical engineering, mechatronics, robotics and computer science.



Types of input

Buttons

Environmental sensors

Cameras

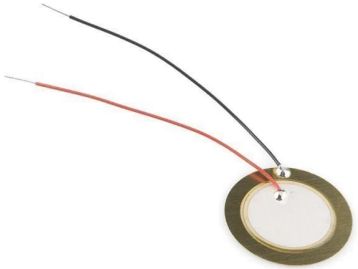
RF sensors

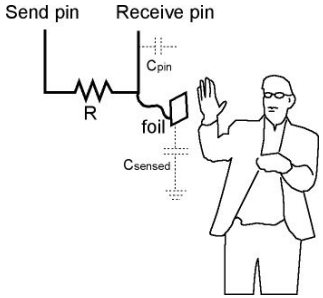
Resistive sensors

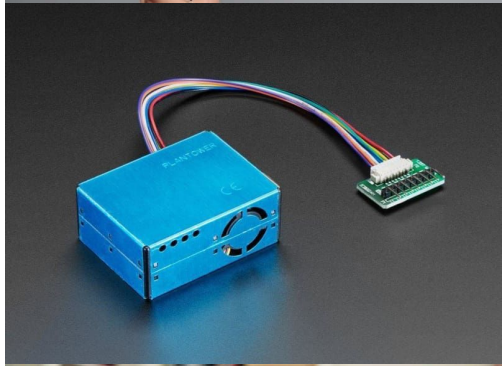
Capacitive sensors

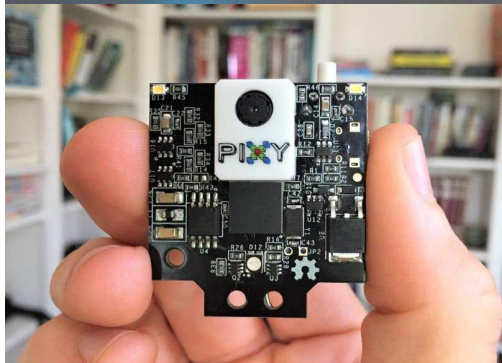
Motion sensors

...









Types of output

LEDs

Motors

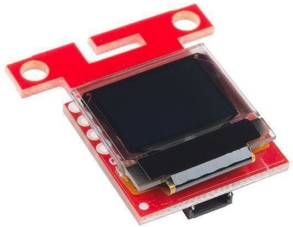
Displays

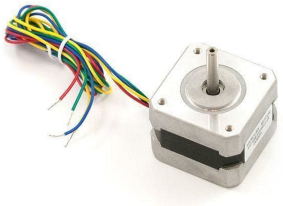
Actuators

Electromagnets

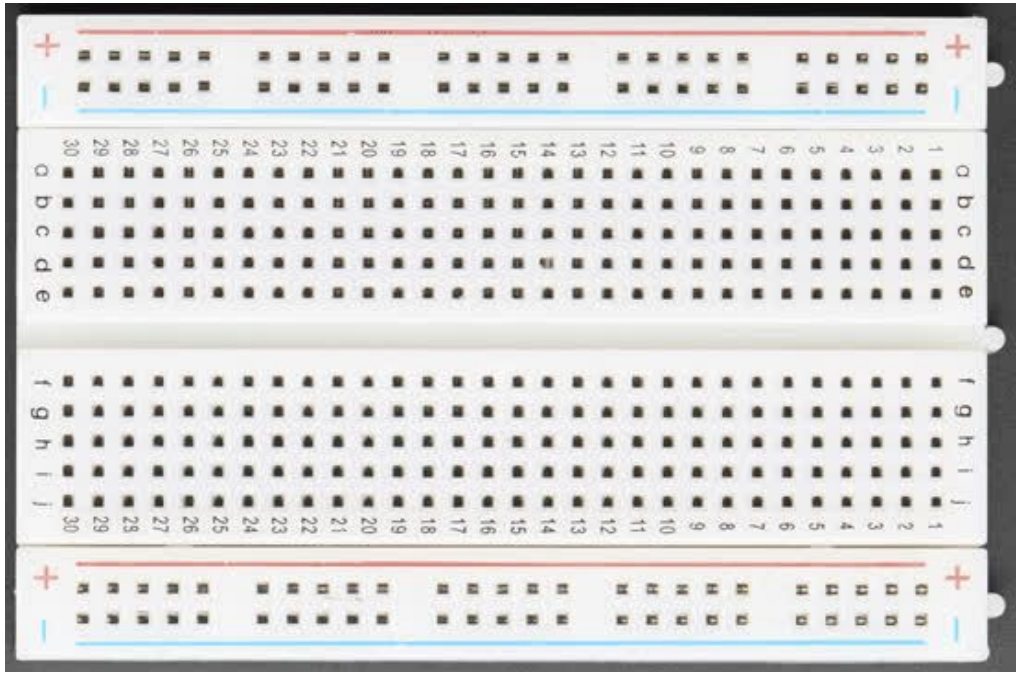
Speakers

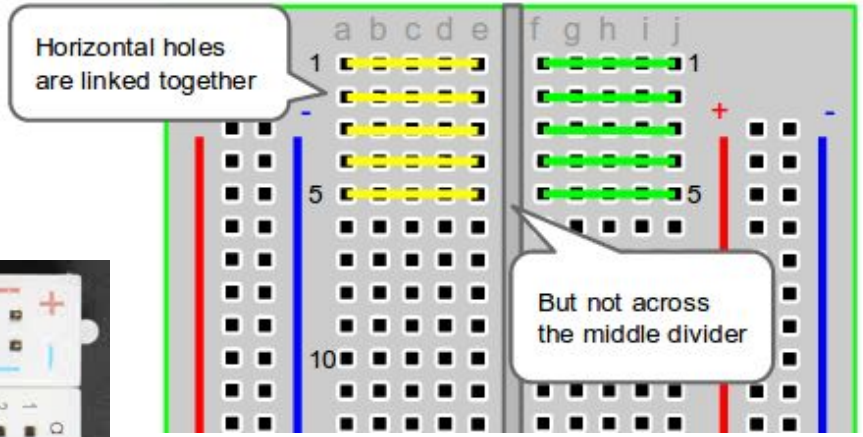
...

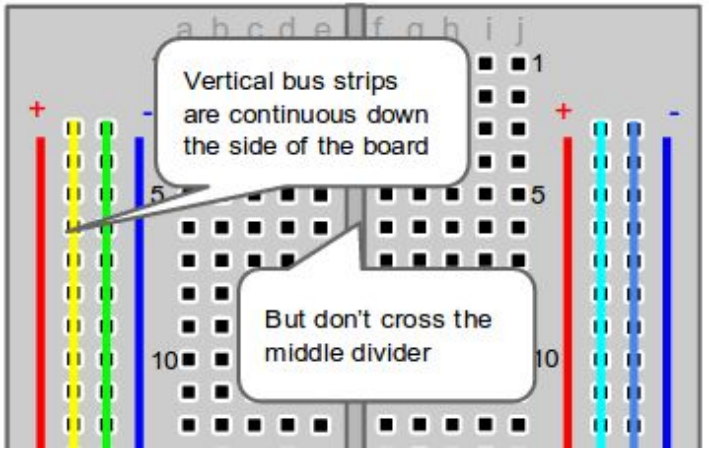




Breadboard




Horizontal holes
are linked together
But not across
the middle divider

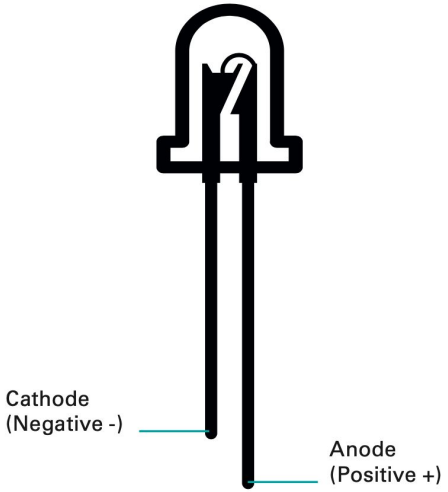

Vertical bus strips
are continuous down
the side of the board
But don't cross the
middle divider

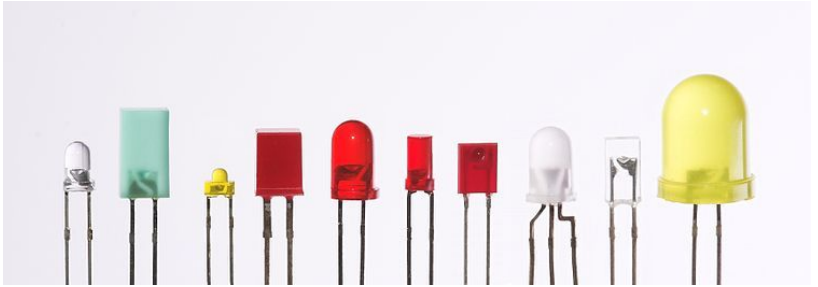
LED

A Light Emitting Diode (LED) is a light that only lets electricity through it one way. If you look at the LED, you will notice that there is a difference in length of the legs. This is how we know which way to use the LED in the circuit.

The longer leg is to signify the anode. This is the side we connect to the positive of a power source. The shorter leg is the cathode. This side needs to be connected to the negative of a power source.

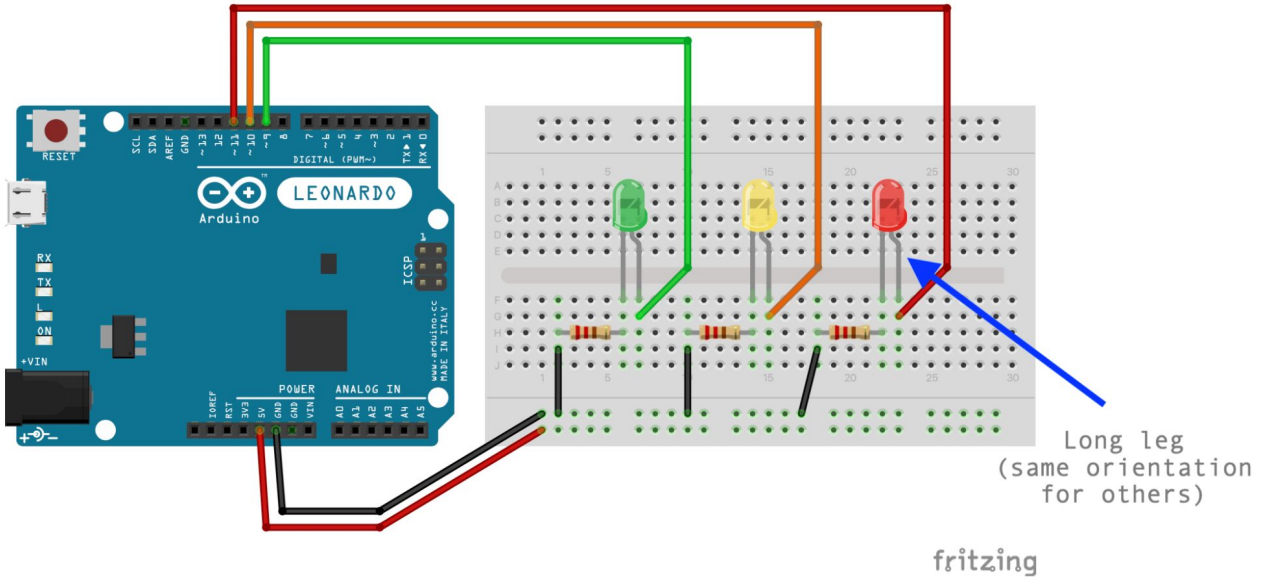
Once in a circuit, you can also easily identify the cathode side by the flat edge on the side of the case.





Traffic Light

3x LED (Red Green Yellow)
3x 220 Ohm Resistors
(Red Red Brown Gold)
8x Jumper Wires



TrafficLight | Arduino 1.8.13

TrafficLight

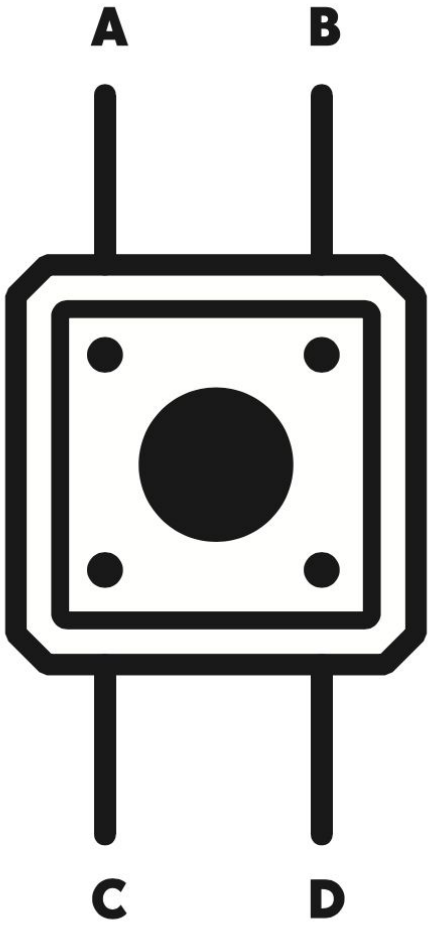
```
// these are the pin numbers for the leds
int redLed = 11;
int yellowLed = 10;
int greenLed = 9;

// a variable to store the length of time the leds will be on and off (in milliseconds)
int ledOnTime = 1000;
int ledOffTime = 500;


void setup() {
  // put your setup code here, to run once:
  // sets the led pins to be an Outputs
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //a continuous loop cycling through the leds

  //turns the red led on for 1000 milliseconds, then turns it off
  digitalWrite(redLed, HIGH);
  delay(ledOnTime);
  digitalWrite(redLed, LOW);

  //waits for 500 milliseconds
  delay(ledOffTime);

  //the same as above, but for the yellow and green led
  digitalWrite(yellowLed, HIGH);
  delay(ledOnTime);
  digitalWrite(yellowLed, LOW);

  delay(ledOffTime);

  digitalWrite(greenLed, HIGH);
  delay(ledOnTime);
  digitalWrite(greenLed, LOW);

  delay(ledOffTime);

}
```

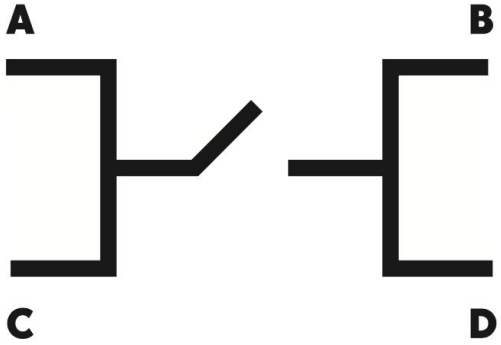
Done Saving.

48 Arduino Leonardo on /dev/cu.SLAB_USBtoUART

Push Button

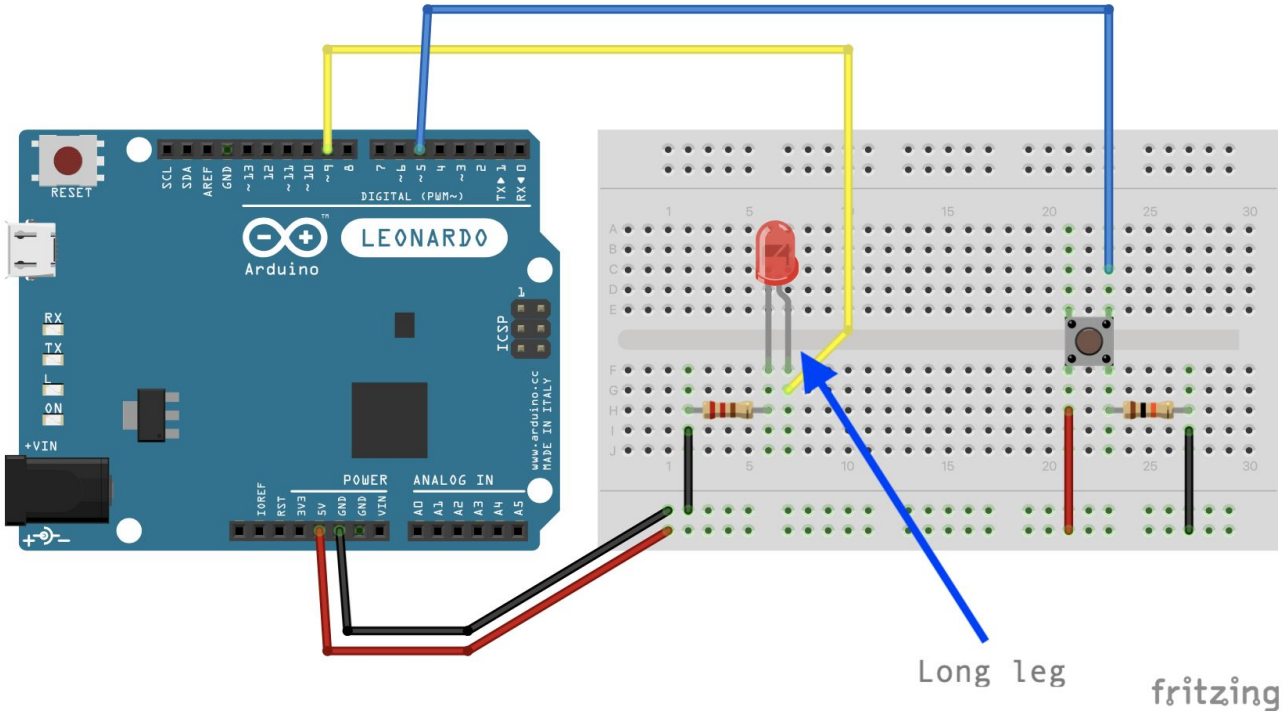
Pushbuttons (also known as tactile switches), are what we use to open and close circuits. Buttons and switches are used in electronics to allow us to interact with our circuits.

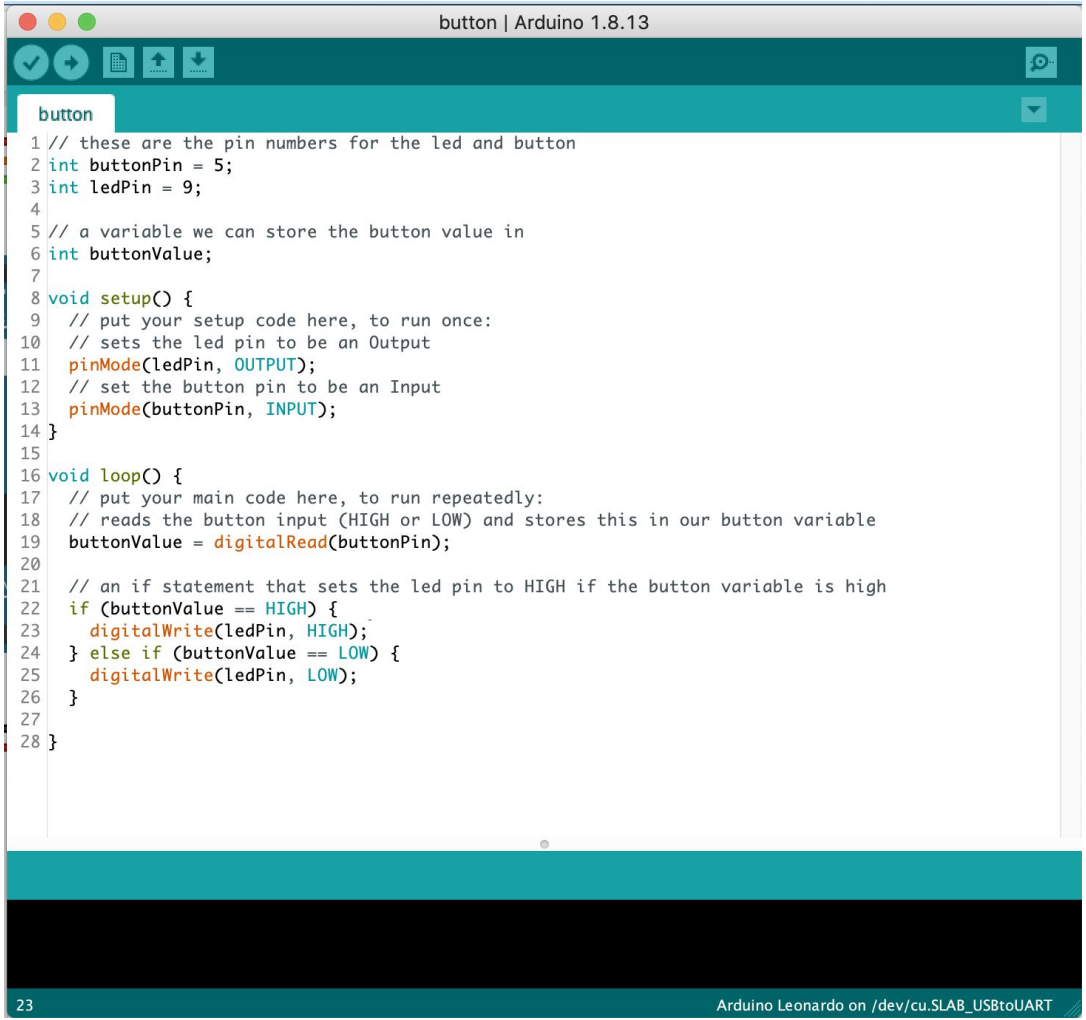




Button

1x LED
1x Push Button
1x 220 Ohm Resistor
(Red Red Brown)
1x 10k Ohm Resistor
(Brown Black Orange)
8x Jumper Wires




button | Arduino 1.8.13
button
// these are the pin numbers for the led and button
int buttonPin = 5;
int ledPin = 9;
// a variable we can store the button value in
int buttonValue;
void setup() {
// put your setup code here, to run once:
// sets the led pin to be an Output
pinMode(ledPin, OUTPUT);
// set the button pin to be an Input
pinMode(buttonPin, INPUT);
}
void loop() {
// put your main code here, to run repeatedly:
// reads the button input (HIGH or LOW) and stores this in our button variable
buttonValue = digitalRead(buttonPin);
// an if statement that sets the led pin to HIGH if the button variable is high
if (buttonValue == HIGH) {
digitalWrite(ledPin, HIGH);
} else if (buttonValue == LOW) {
digitalWrite(ledPin, LOW);
}
}
23
Arduino Leonardo on /dev/cu.SLAB_USBtoUART

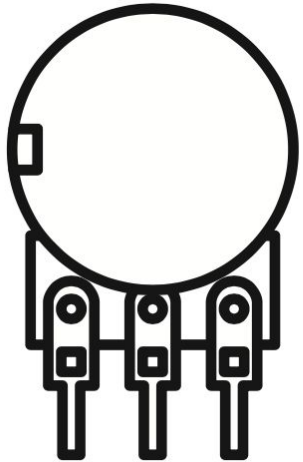
Circuits and Sketches

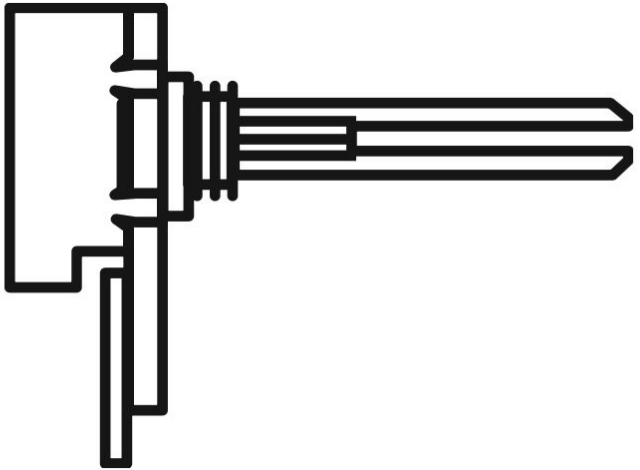
bit.ly/3uKyI

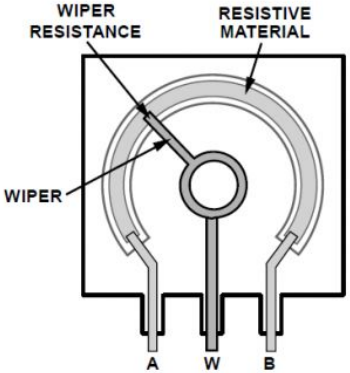
Potentiometer

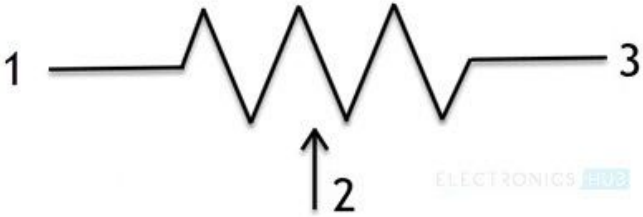
A potentiometer is a resistor that allows the user to change the potential of the resistance by turning the knob. Potentiometers have three legs, the three legs are: input, output and ground. In this circuit, we are using the output of the potentiometer as a sensor value to be sampled by the analog input.

To make sure the output value is in range of the analog input, we will supply the potentiometer with 5v. This is the maximum value the analog inputs can sample.



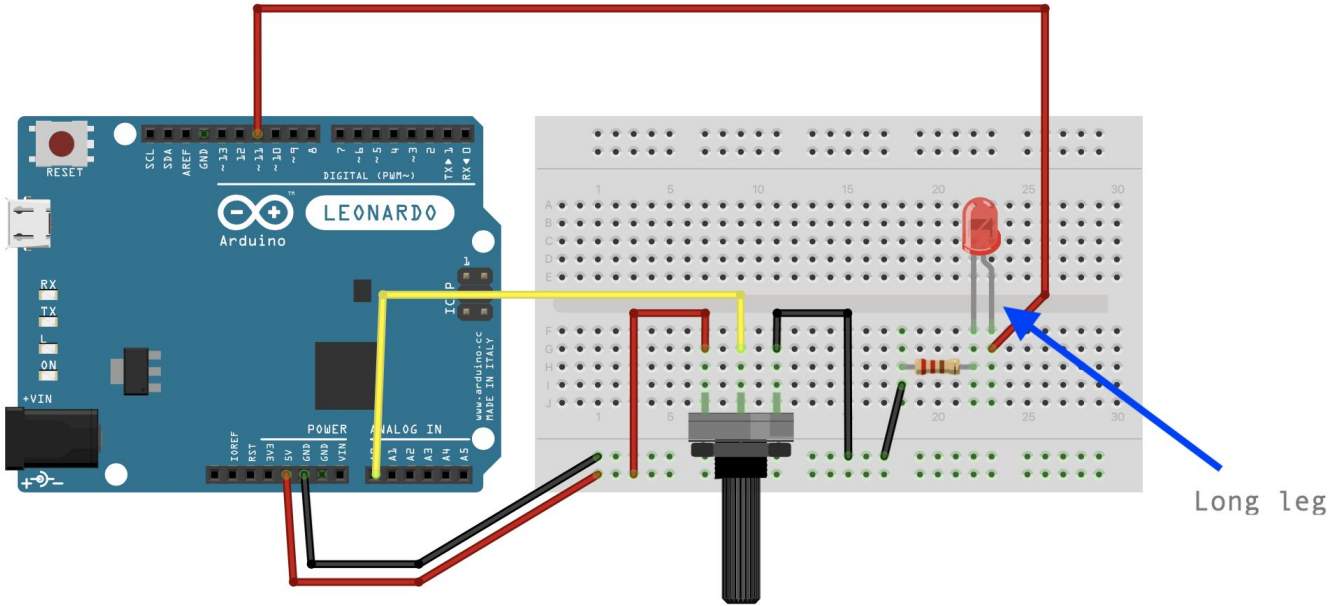






Fading

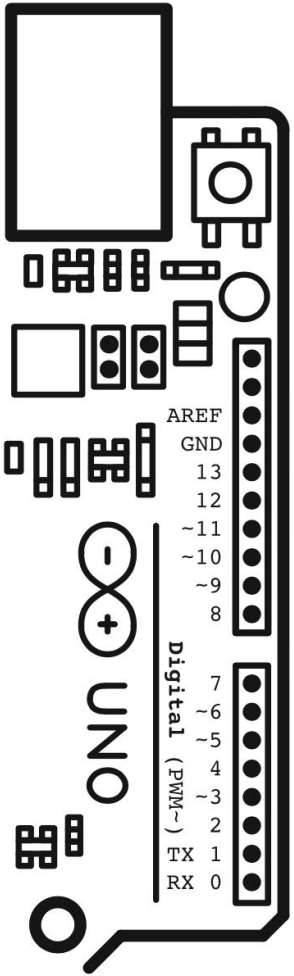
1x LED
1x 220 Ohm Resistor
(Red Red Brown)
1x 10k Potentiometer
7x Jumper Wires

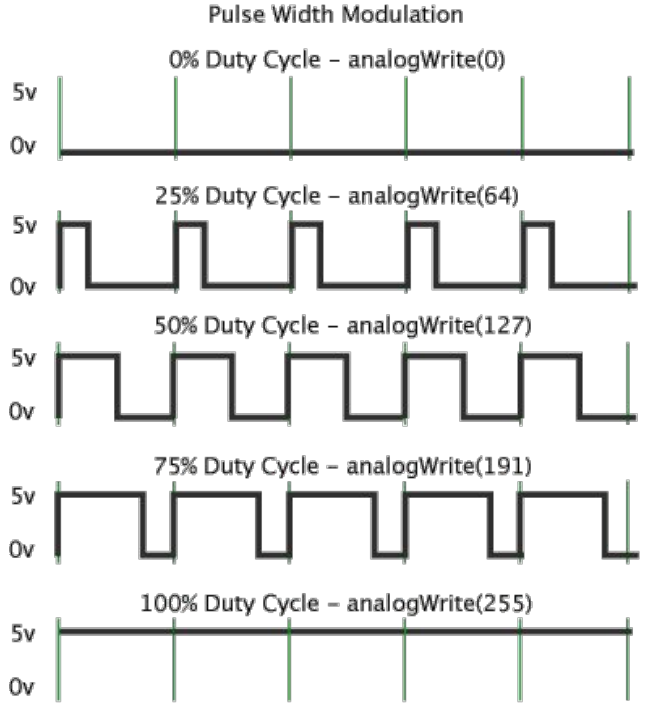


PWM (Pulse Width Modulation)

Look at the Arduino Digital pins closely, you will notice that 6 of the pins have tilda signs (~) next to the pin number.

These are to tell us that this pin can perform Pulse Width Modulation (PWM), a technique in digital computing used to create the perception of changeable voltage. Digital pins can only output 0v or 5v (On or Off). With PWM it creates the illusion that we are outputting different voltages to this. we can do this is to use the Arduino's **analogWrite()** function.





25% duty cycle

25% of 5v = 1.25v

analogWrite(64)

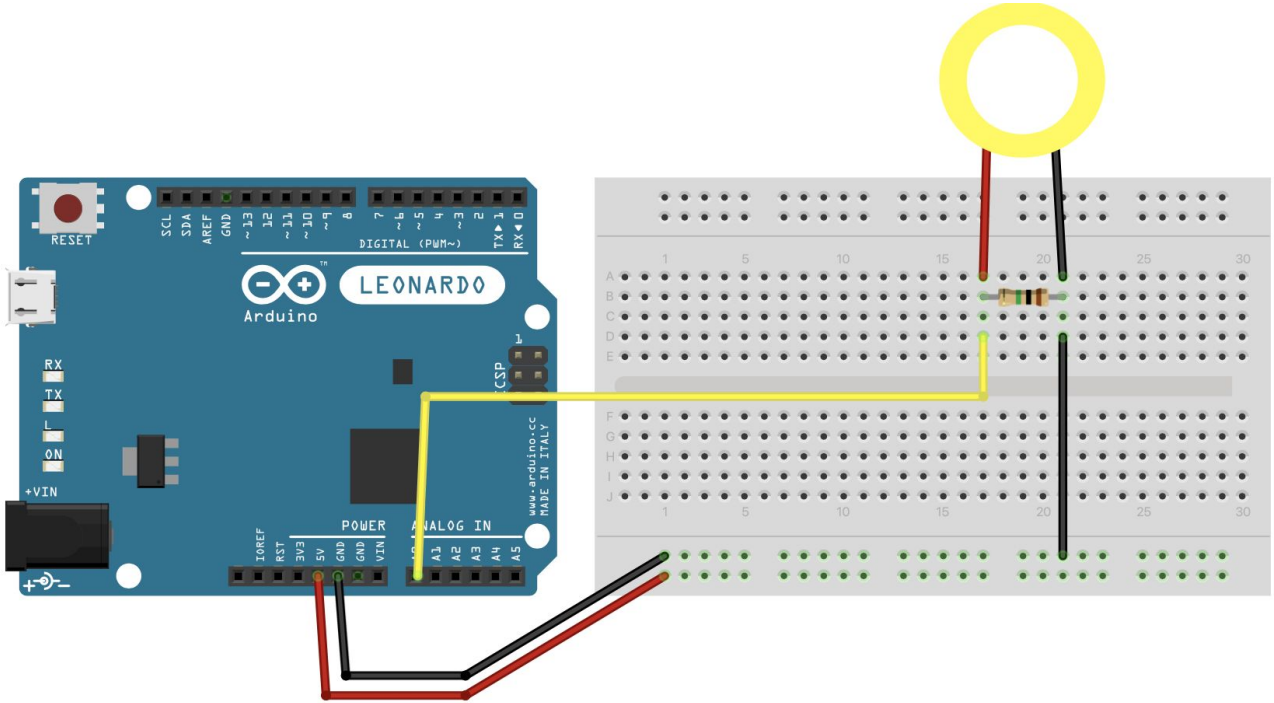
75% duty cycle

75% of 5v = 3.75v

analogWrite(191)

Knock Sensor

1x Piezo Sensor
1x 1 MegaOhm Resistor
(Brown Black Green)
4x Jumper Wires



knockSensor | Arduino 1.8.13

knockSensor §

```
int piezoPin = A0;
int piezoValue;

int piezoThreshold;

void setup() {
  // put your setup code here, to run once:
  //Set up Serial communication to communicate with the Serial Monitor at 9600 Baud rate
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  // Sample the analog input the piezo is connected to
  piezoValue = analogRead(A0);

  // Send this sampled value to the Serial monitor
  Serial.println(piezoValue);

  // delay so the Serial monitor isn't flooded by incoming values
  delay(20);

}
//Look at the incoming data on your serial monitor.
//Add an Led to your circuit and create blinking knock sensor using piezoThreshold
//and an if statement (see Button example)
```

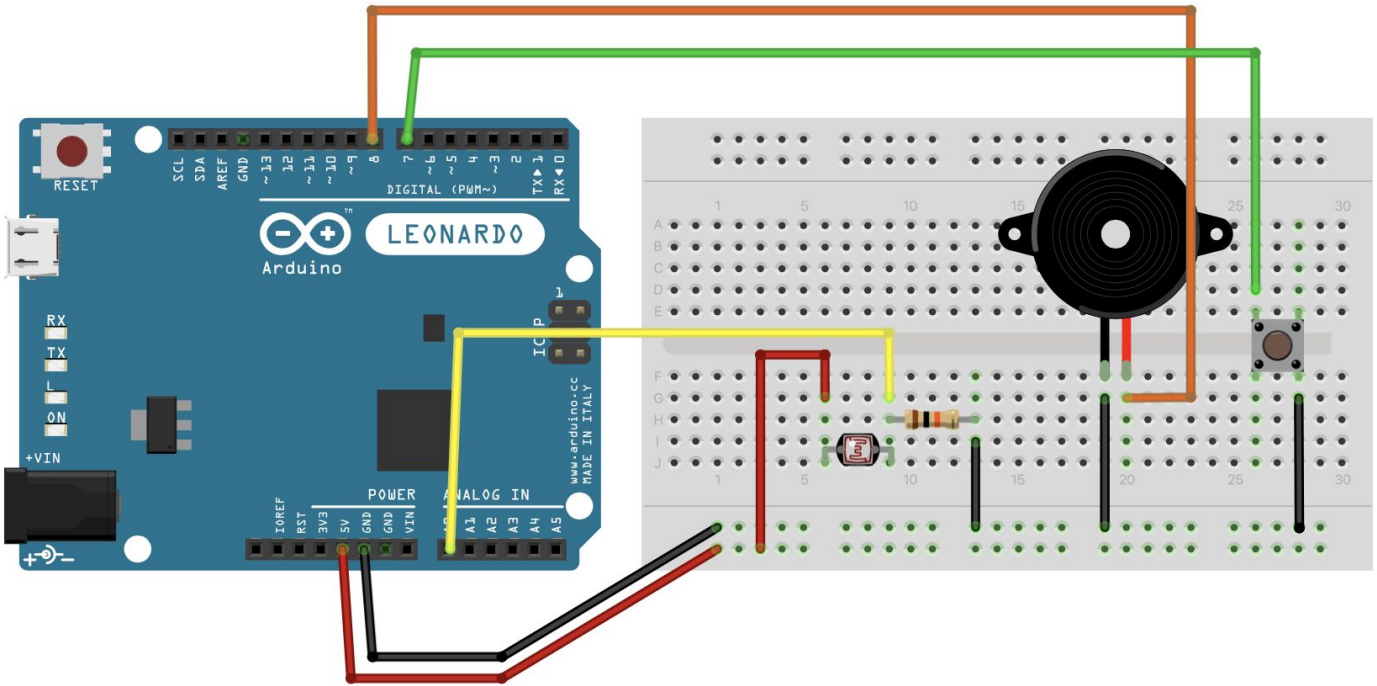
Done Saving.

30

Arduino Leonardo on /dev/cu.SLAB_USBtoUART

Theremin

1x Light Dependent Resistor
1x Piezo Buzzer
1x Push Button
1x 10k Resistor (Brown Black Orange)
9x Jumper Wires



theremin | Arduino 1.8.13

theremin

```
//program variables
int ldrMin = 0;
int ldrMax = 1023;
int freqMin = 200;
int freqMax = 1200;
int freqOutput;


void setup() {
  // set up inputs and outputs
  pinMode(piezoPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); //no need for resistor!

}

void loop() {

  //sample the light dependent resistor
  ldrValue = analogRead(ldrPin);

  //map the ldr input value (0-1023) to frequency value
  //using map() function. map(input,input_minimum,input_maximum,output_minimum,output_maximum)
  freqOutput = map(ldrValue, ldrMin, ldrMax, freqMin, freqMax);

  //check if button is pressed to make the sound. logic is inverted because we're using the
  //internal pullup.
  if (digitalRead(buttonPin) == LOW) {
    tone(piezoPin, freqOutput);
  } else {
    noTone(piezoPin);
  }

  delay(5);
}
```

Done Saving.

```
Sketch uses 5662 bytes (19%) of program storage space. Maximum is 28672 bytes.
Global variables use 178 bytes (6%) of dynamic memory, leaving 2382 bytes for local variables. Maximum
```

42 Arduino Leonardo on /dev/cu.SLAB_USBtoUART

Tilt Sensor

The tilt switch works by using a metal ball-bearing inside the switch (shake it and you'll hear it rattle!). This means that the ball-bearing is conductive, allowing electricity to flow through and close the circuit if it is touching the legs of the switch.

As this is a physical open and close of the switch we can sense a digital On or Off (0 or 1).

We can wire a tilt switch up the same as push button.