to assign a value to that variable, simply write the following:

```
type nameOfVariable = value;
```

Here are some more examples of how a variable is used:

Copy to clipboard

```
int x = 5;

char theLetterC = 'c';

String hello = "Hello World";
```

Notice how char's value has ' ' while string's value has " ". Again, this is just syntax that the code follows.

Conditionals

Conditionals are extremely useful in order to structure your code better. Below are some of the most basic conditionals found in almost all programming languages.

If / Else

Whether it's deciding whether to wear a sweater vs a raincoat or deciding to spend the weekend at the house or with friends, if-else conditions are relevant in every decision you make. Depending on the type of program you make, you can use if-else conditions to make your code run only if it hits a certain condition. Generally, if-else conditions are written as follows:

Copy to clipboard

```
if (statement) {
  // Code goes here
} else {
  // Code goes here
}
```

Let's take a look at an example. Let's say we want to make a slot machine game. A player pulls the handle and receives a three digit number. If the three digits are the same, then the player will see the message, "You won!". If the player receives three digits that are different, then the player will see the message, "Sorry, try again!"

To code this, we have to recognize that if the player gets three of the same number, then all those variables will have the same value. For example, let's say the player has three variables, x, y, and z. If the player has x = 5, and the player wins, variables y and z must be assumed to be the same value of x such that x = y = z = 5. If the player has lost, then y and z can be any other number in such a way that $x \neq y \neq z$. Let's put this thought into code!

Copy to clipboard

```
if (numberOne == numberTwo && numberTwo == numberThree) {
  Serial.print("You win!");
} else {
  Serial.print("Sorry, try again!");
}
```

The `&&` sign means “and”. So, if x == y and y == z, then all three values are the same. If the condition is not fulfilled, then the statement will print “Sorry, try again!”. A programmer can put multiple if statements if desired. To do that you'd use a structure like this.

Say we wanted to make the slot machine game reward the play if they receive two out of three same numbers. We'd use an "or" which looks like this `||`. Check this out.

Copy to clipboard

```
if (numberOne == numberTwo && numberTwo == numberThree) {
  Serial.print("You win!");
} else if (numberOne == numberTwo || numberTwo == numberThree || numberOne == numberThre
  Serial.print("You half-win!");
} else {
  Serial.print("Sorry, try again!");
}
```

Now if two but not three numbers are the same they player wins something. Now we have three separate conditionals at work. We used an else if statement to add that third condition. You can use else if statements to make as many additional conditionals as you need. The code below uses 4.

Copy to clipboard

```
if (condition) {
  // Code goes here
} else if (condition) {
  // Code goes here
} else if (condition) {
  // Code goes here
} else {
  // Code goes here
}
```

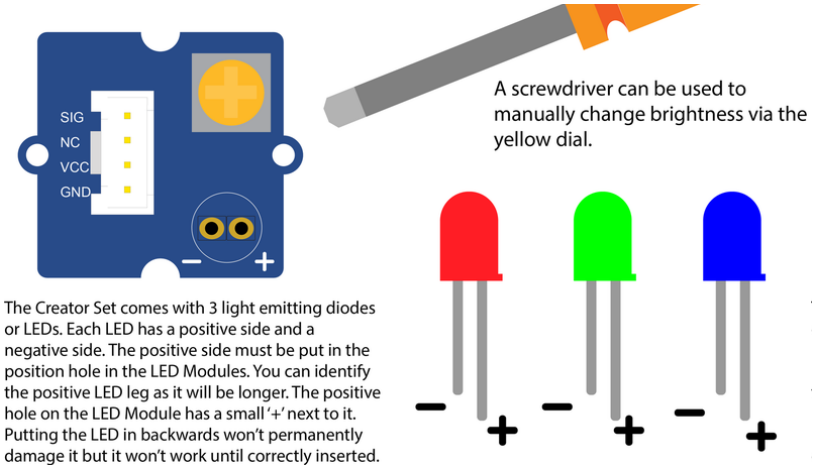
Nightlight

Start

This project uses the light sensor to create a nightlight that only turns on when it's dark.

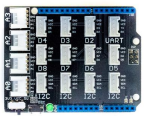
Modules

Gather the following parts to complete this project. If your LED module doesn't have an LED on it, then one must be inserted. The Creator Set comes with 3 Light Emitting Diodes, or LEDs. Each LED has a positive side and a negative side. The positive side must be put in the positive hole in the LED Module. The positive LED "leg" is the longer one. The positive hole on the LED Module has a small ‘+’ next to it. Putting the LED in backwards won’t permanently damage it, but it won’t work until correctly inserted.



LED Module

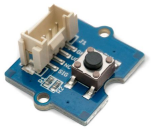
Parts



All Parts x Qty

LED x 1

Light x 1

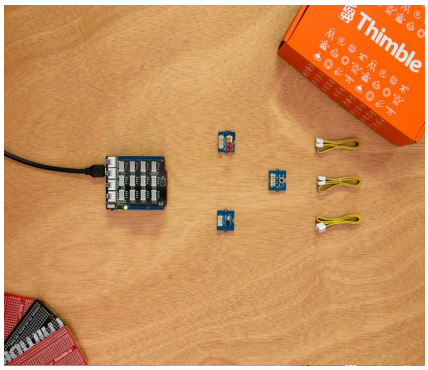


Button x 1

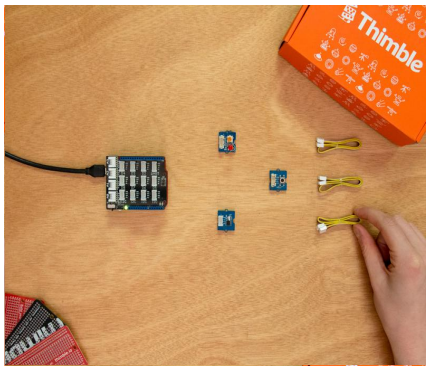
Cable x 3

Starting With the Light Part of Nightlight

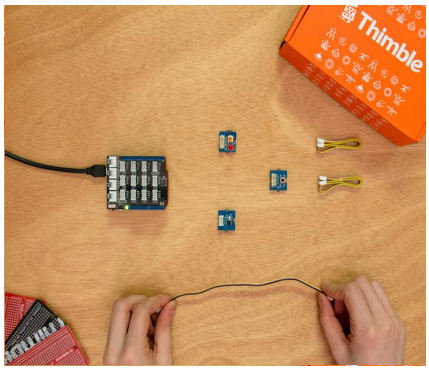
Take a cable and unwrap it. Plug one side into the light sensor and the other into any **Analog** socket.



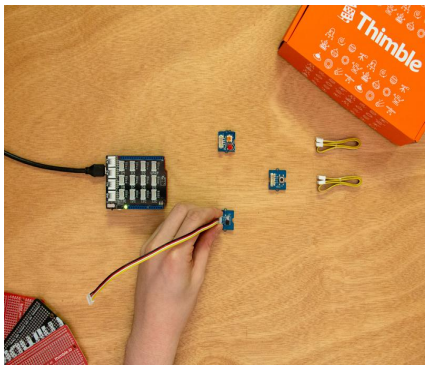
All the parts needed for this project



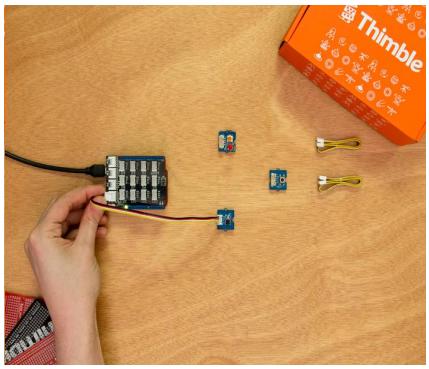
Take a cable...



... and unwrap it



Plug one side into the light sensor...



... and the other into Analog socket A0

Upload

Upload the code below. This tutorial uses Analog socket A0. If you are using a different socket, update the code after copying it.

Copy to clipboard

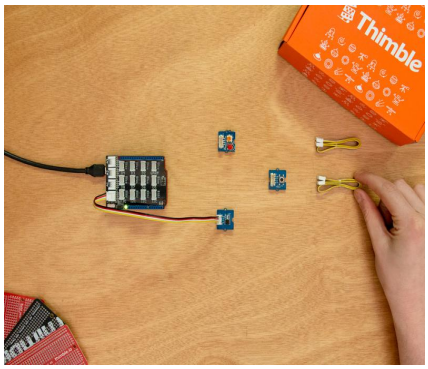
```
//Change here if you're using a different socket
#define sensorSocket A0

int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(sensorSocket);
  Serial.println(val);
}
```

Observe

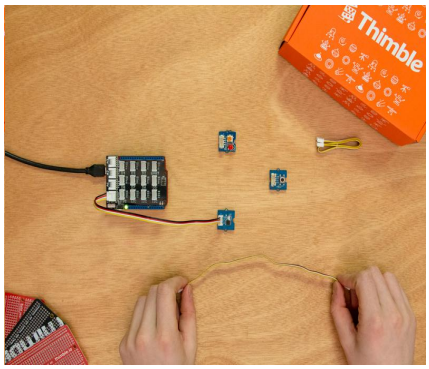
Open up the Serial Plotter and move your hand over the light sensor. Check out what values you can get. Look at the light value you get when the sensor is uncovered and covered. We'll need those two later.

LED Modules

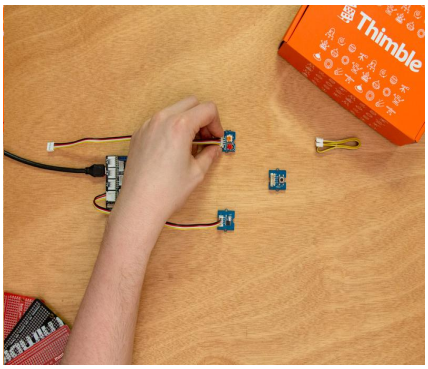
Take a cable and unwrap it. Plug one side into the led socket and the other into D6.



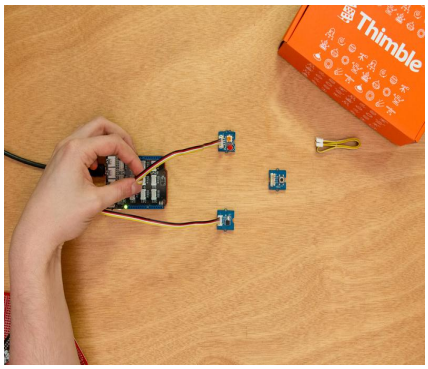
Take a cable...



... and unwrap it



Plug one side into the LED socket



The other into Digital socket D6

Upload

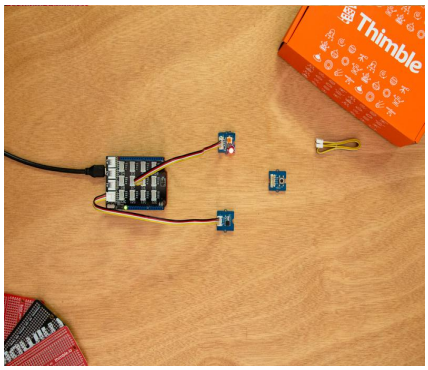
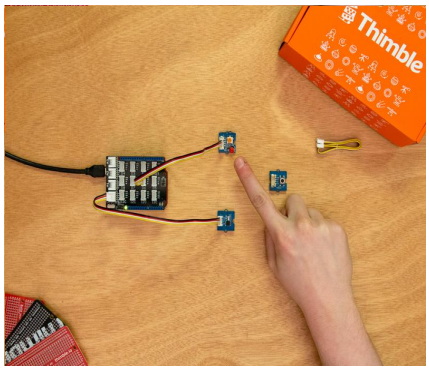
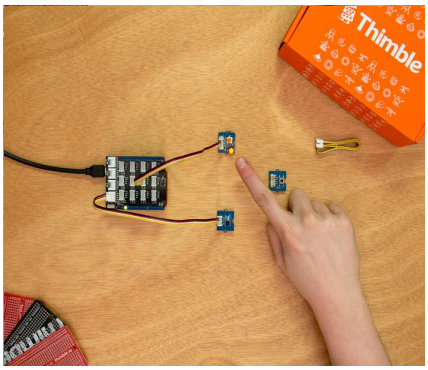
Upload the code below. This tutorial uses **Digital** socket 6. If you are using a different socket, update the code after copying it.

Copy to clipboard

```
#define ledSocket 6

int i;
int fadeTime = 50;

void setup()
{

}

void loop()
{
  for (i = 0; i <= 255; i++) {
    analogWrite(ledSocket, i);
    delay(fadeTime);
  }
  for (i = 255; i >= 0; i--) {
    analogWrite(ledSocket, i);
    delay(fadeTime);
  }
}
```

Observe

The LED should be fading in and out.

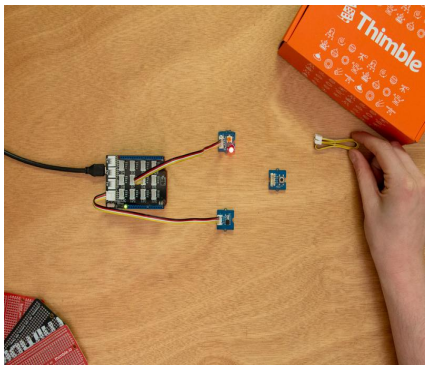




Modify

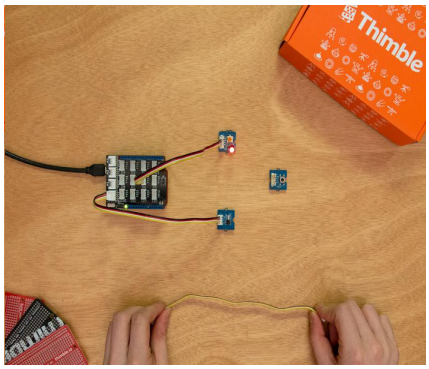
Change the `fadeTime` delay value to change the speed of the fading.

Button

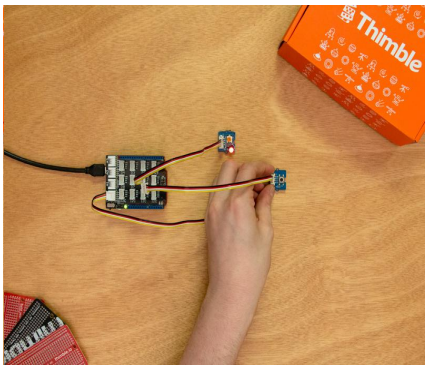
Take a cable and unwrap it. Plug one side into the button socket and the other to **Digital** socket D5.



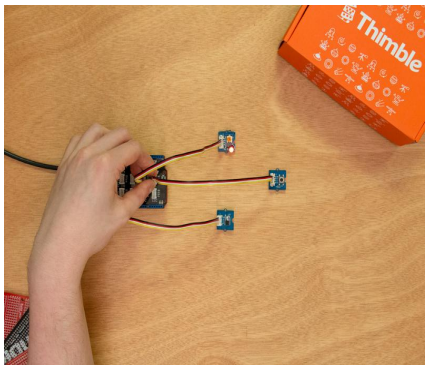
Take a cable...



... and unwrap it



Plug one side into the button socket



The other into Digital socket D5

Upload

Upload the code shown below. This tutorial uses **Digital** socket 5. If you are using a different socket, update the code after copying it.

Copy to clipboard

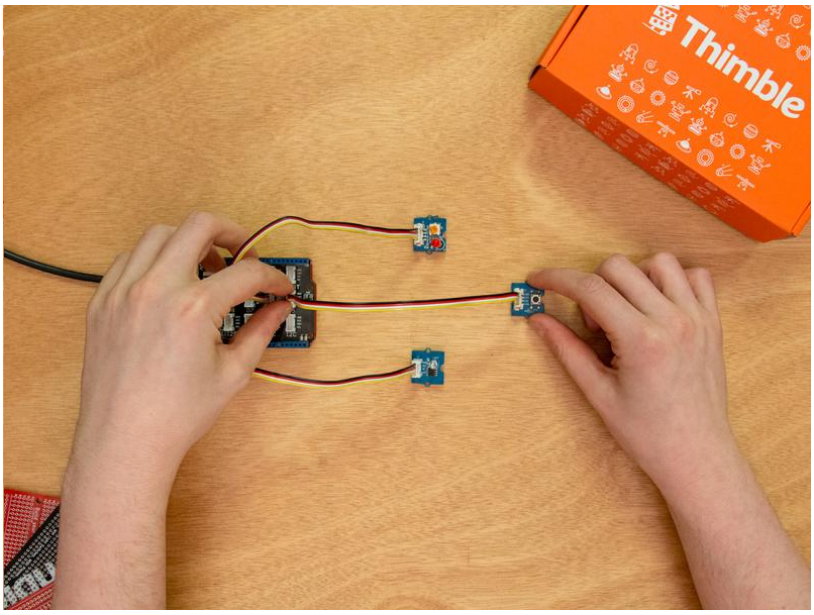
```
//If you use different socket numbers update them below
#define buttonSocket 5
#define ledSocket 6

void setup()
{
  pinMode(buttonSocket, INPUT);
  pinMode(ledSocket, OUTPUT);
}

void loop()
{
  if (digitalRead(buttonSocket)) {
    digitalWrite(ledSocket,HIGH);
  } else {
    digitalWrite(ledSocket,LOW);
  }
}
```

Observe

Pressing the button makes the LED turn on. This uses two states. An **On** state and an **Off** state. The button switches between those two. Now we'll take those two states and make the light sensor switch between them.

Nightlight

At this point, the button can be removed.



Remove the button

Upload

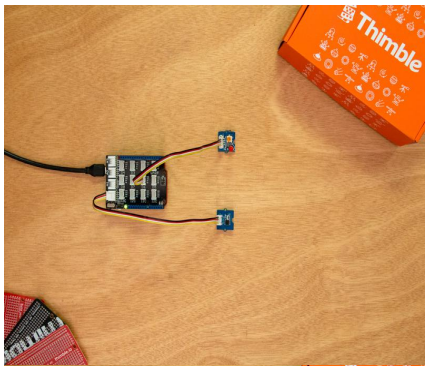
Upload the code shown below. Add in the covered light value from earlier in the `lowTrigger` variable. Add in the uncovered light value from earlier in the `highTrigger` variable.

Copy to clipboard

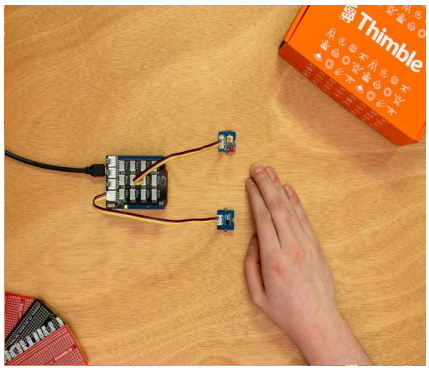
```
//If you use different socket numbers update them below
#define ledSocket 6
#define lightSensorSocket A0

int lowTrigger = 250;  //<- Change to YOUR measured value
int highTrigger = 300; //<- Change to YOUR measured value
int lightAmount;
int state;

void setup()
{
  pinMode(ledSocket, OUTPUT);
}

void loop()
{
  lightAmount = analogRead(lightSensorSocket);
  if (lightAmount < lowTrigger) {
    state = 1;
  } if (lightAmount > highTrigger) {
    state = 0;
  }

  if (state) {
    digitalWrite(ledSocket, HIGH);
  } else {
    digitalWrite(ledSocket, LOW);
  }
}
```

Observe

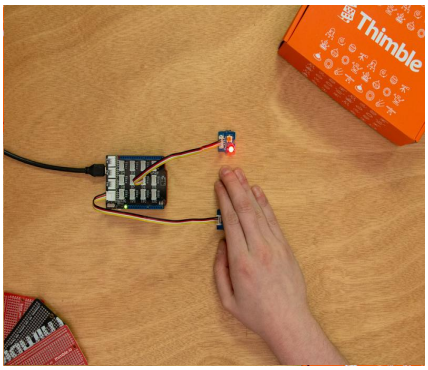
The led should now be light sensitive. When exposed to high levels of light it will turn off. Once it gets dark, it will turn on.



The finished nightlight project



Uncovered and Off



Covered to simulate the light level in the dark

Modify

Change the trigger number to more accurately have the nightlight turn on at the right darkness.

Challenge
Use the fading technique to set certain light amounts to LED brightness levels.

Coding Basics 201

Start

This section will go over some additional computer programming topics.

Loops

Loops are used when you want to do the same instruction or a set of instructions multiple times. There are two main types of loops a **for** loop and a **while** loop.

For

A **for** loop repeats *for* a set number of times. Its basic design looks like this.

Copy to clipboard

```
for (initial condition; stopping condition; incrementing value) {
  // Code goes here
}
```

Let's say I have 10 leds that I want to turn on. They are pins 1 to 10. I could write this piece of code.

Copy to clipboard

```
digitalWrite(1, HIGH);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
digitalWrite(5, HIGH);
digitalWrite(6, HIGH);
digitalWrite(7, HIGH);
digitalWrite(8, HIGH);
digitalWrite(9, HIGH);
digitalWrite(10, HIGH);
```

Or I could use a for loop.

Copy to clipboard

```
for(int i = 1; i <= 10; i++) {
  digitalWrite(i, HIGH);
}
```

Lets go over this code line by line:

`int i = 1`

We need a counting variable to keep track of where we are in our loop. That's what the integer i is doing.

`i <= 10`

We only want the loop to run if i is less than or equal to 10. We don't want to try and turn an led on pin 11 on.

`i++`

i starts at 1 and at the end of each loop `i++` runs and add one to i.

`digitalWrite(i, HIGH);`

There is no led pin i. The variable i changes it's value with each run of the loop. Starting at 1 then 2 and all the way to 10.

While

A **while** loop runs *while* some condition is true. You use it when you're not sure how many loops you'll need. The basic design looks like this.

Copy to clipboard

```
while (condition) {
  // Code goes here
}
```

Let's say you want to take a reading with a sensor. You send it a signal to take a reading and must wait for its answer but the time you have to wait is different with each measurement. You'd want to use a while loop.

Copy to clipboard

```
digitalWrite(sensorSignal, HIGH);

while (analogRead(sensorReading) == 0) {
  // Do nothing
}

reading = analogRead(sensorReading);
```

Our mystery sensor reports back `0` when it's taking a reading. The code above keeps running that empty while loop while the sensor is doing it's job. Once it is finished it reports back a non-zero number. That makes the condition of `analogRead(sensorReading) == 0` false, and the loop is over.

Arduino void loop()

In every Arduino sketch there is something called the `void loop()`. So is it a while loop or for loop? It's a different kind of loop. It will repeat as long as the Arduino is powered on, it never stops.

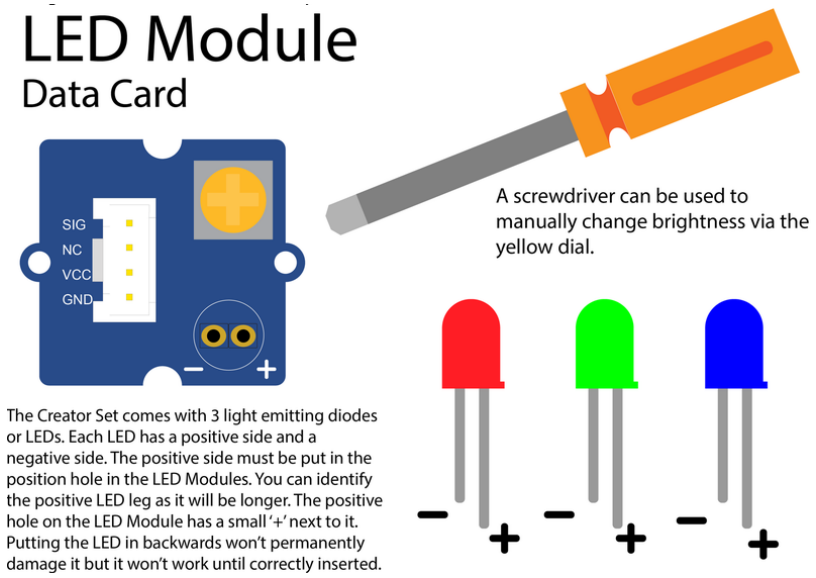
Clapper

Start

The clapper will turn an LED on and off using a clap or loud noise. This project uses the analog sound sensor as a digital sensor. Analog sensors are used when the output values can have a wide range. For the clapper, we only care about large changes in the sound level and not the sound level itself.

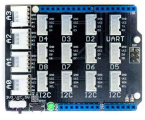
Modules

Gather the following parts to complete this project. If your LED module doesn't have an LED on it, then one must be inserted. The Creator Set comes with 3 light emitting diodes, or LEDs. Each LED has a positive side and a negative side. The positive side must be put in the position hole in the LED Modules. The positive LED side (leg) is the longer one. The positive hole on the LED Module has a small '+' next to it. Putting the LED in backwards won't permanently damage it, but it won't work until correctly inserted.



LED Module

Parts



All Parts x Qty

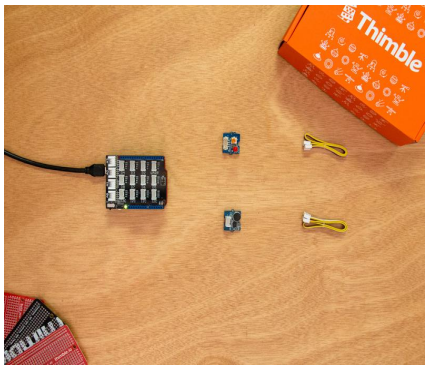
Sound x 1

LED x 1

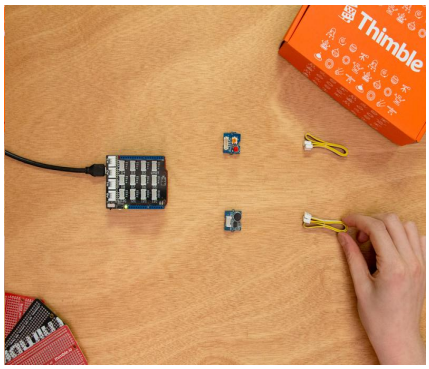
Cable x 2

Sound Sensor

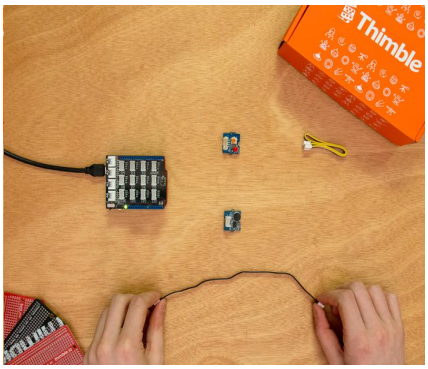
Take a cable and unwrap it. Plug one side into the sound sensor socket and the other into **Analog** socket A0.



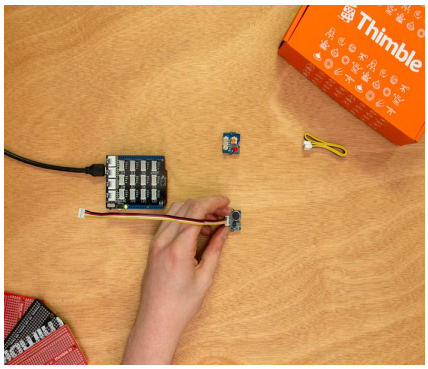
All the parts you'll need



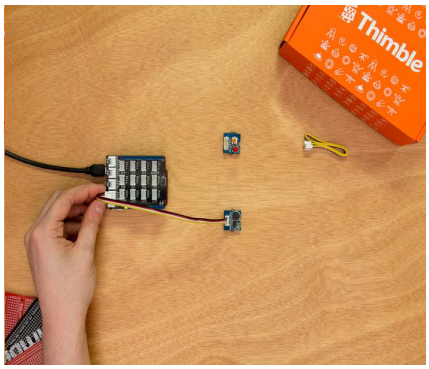
Take a cable...



... and unwrap it



Plug one side into the sound sensor socket



... and the other into any Analog socket

Upload

Upload the following code. The example below uses the *A0* analog pin. You can use any of them just remember to update the sketch.

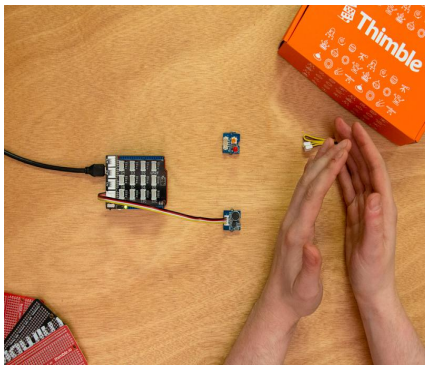
Copy to clipboard

```
//If you aren't using A0 as your socket change it below
#define sensorSocket A0

int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(sensorSocket);
  Serial.println(val);
}
```

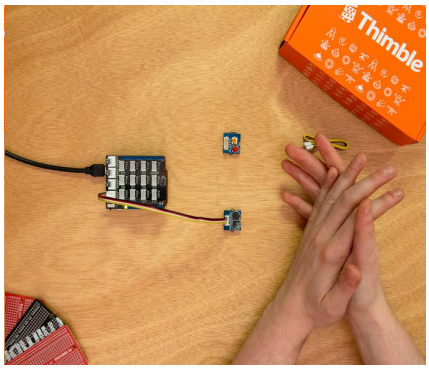
Observe

Open up the Serial Plotter and check out the what a normal sound level is. Try clapping or talking and watch the sound sensor react. You'll see a big jump in the level of sound. Find a value that the jump goes through but is far away from the

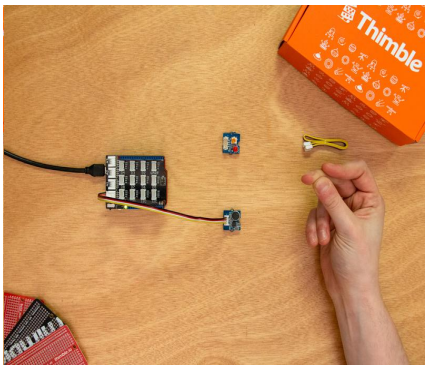
ambient sound level.



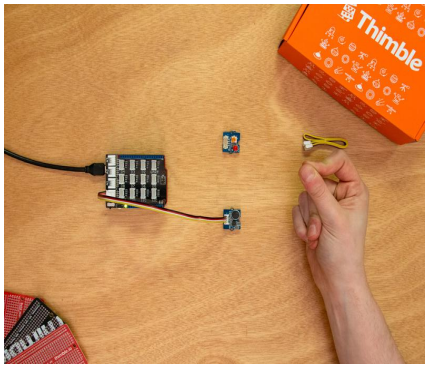
A clap



A clap



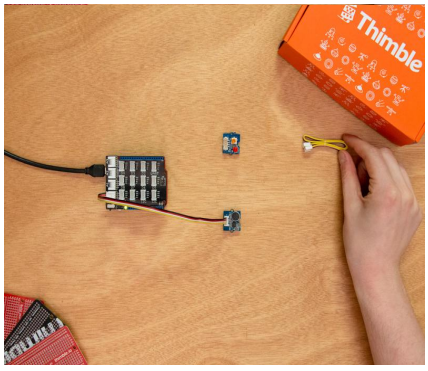
A snap



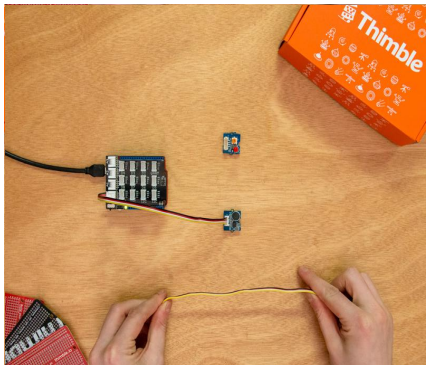
A snap

Clapper

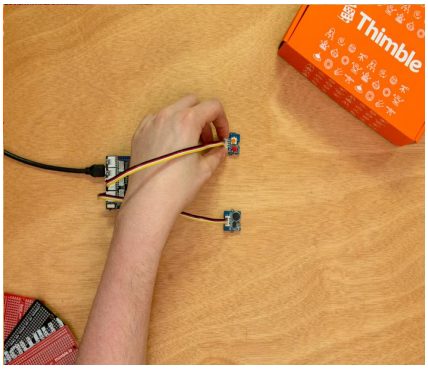
Take a cable and unwrap it. Plug one side into the led socket and the other into any **Digital** socket.



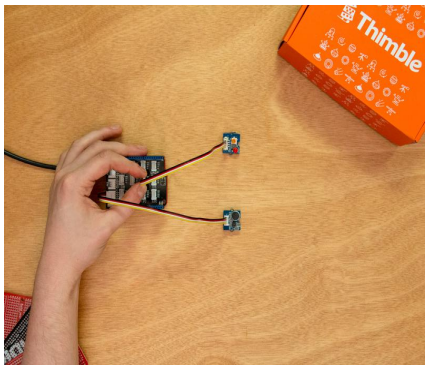
Take a cable...



... and unwrap it



Plug one side into the led socket



... and the other into any Digital socket

Upload

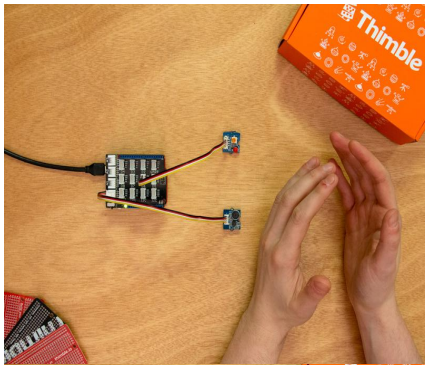
Upload the code below. This tutorial uses **Digital** socket 6. If you are using a different socket update the code after copying it. Set the `trigger` variable as that value you found from above.

Copy to clipboard

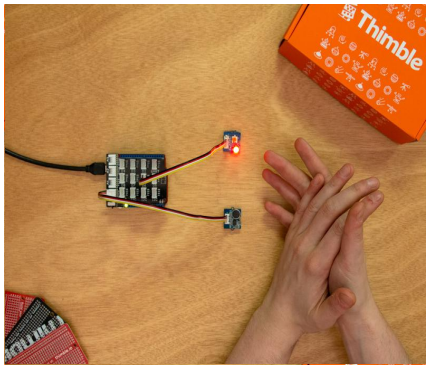
```
//If you aren't using A0 as your sound sensor socket change it below
#define soundSensorSocket A0

//If you aren't using pin 6 for your led socket change it below
#define ledSocket 6

int soundState;
int ledState;
int trigger = 600;

void setup()
{
  pinMode(ledSocket, INPUT);
  pinMode(ledSocket, OUTPUT);
}

void loop()
{
  if (analogRead(soundSensorSocket) > trigger) {
    soundState = 1;
  } else {
    soundState = 0;
  }

  ledState = digitalRead(ledSocket);
  if (ledState == 1 && soundState == 1) {
    delay(400);
    digitalWrite(ledSocket, LOW);
  }
  if (ledState == 0 && soundState == 1) {
    delay(400);
    digitalWrite(ledSocket, HIGH);
  }
}
```

Observe

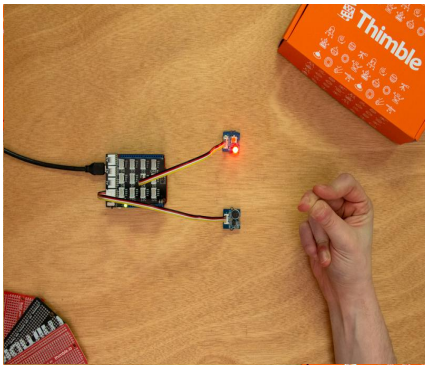
Try clapping or making a loud noise. The LED will switch its states.



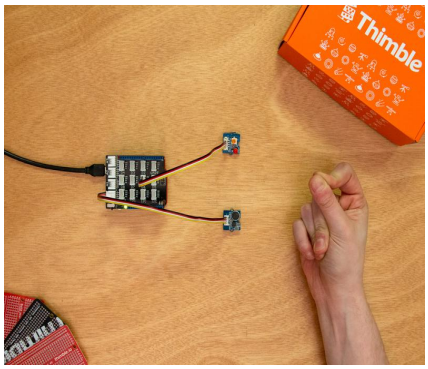
A clap



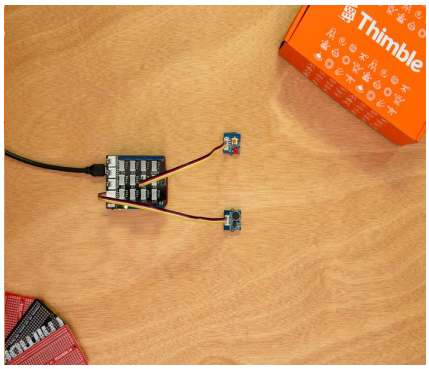
A clap



A snap



A snap



A finished project

8 Bit Music Conductor

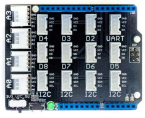
Start

This project only uses a Piezo Buzzer to create a musical master piece.

Modules

Gather the following parts to complete this project.

Parts

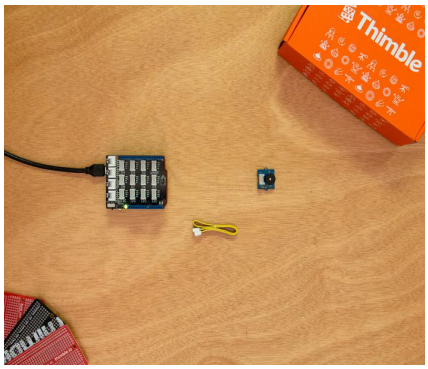

All Parts x Qty


Piezo Buzzer x 1

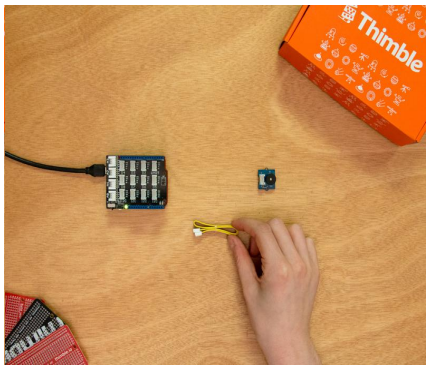

Cable x 1

Basic Music Player

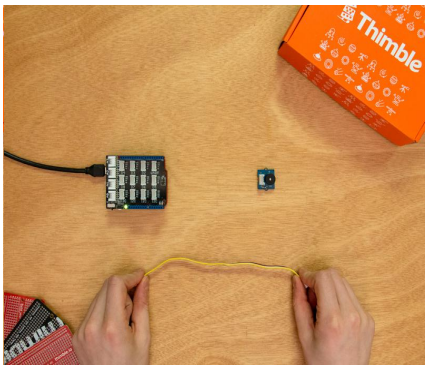
Take a cable and unwrap it. Plug one side into the buzzer socket and the other into **Digital** socket D6 to start making music.



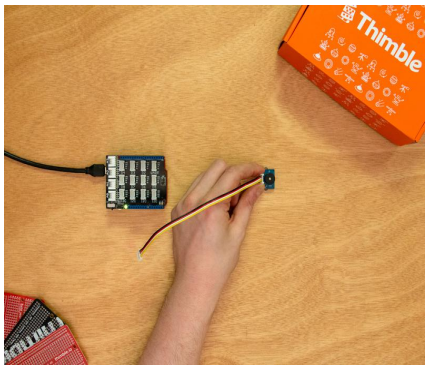
All the parts you'll need



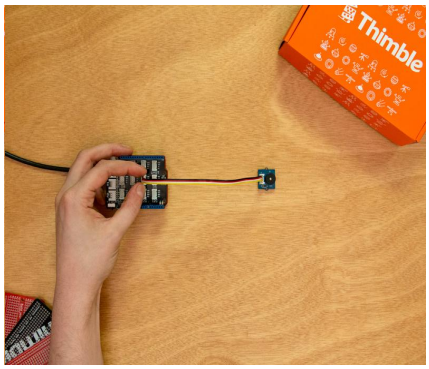
Take a cable...



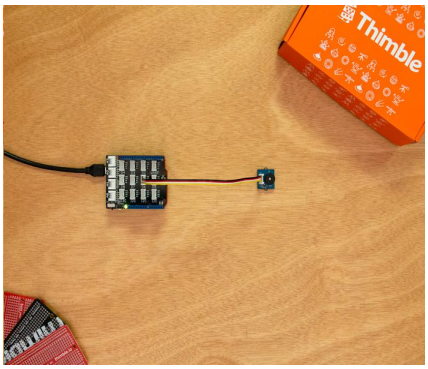
... and unwrap it



Plug one side into the buzzer socket



... and the other into Digital socket D6



All finished

Upload

Upload the code below. This tutorial uses **Digital** socket 6. If you are using a different socket update the code after copying it.

Copy to clipboard

```
//Change here if you're using a different socket
#define buzzerSocket 6

int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

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

  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}

void loop() {

}

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(buzzerSocket, HIGH);
    delayMicroseconds(tone);
    digitalWrite(buzzerSocket, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}
```

Observe

It plays a simple melody on start up.

Modify

You can change the notes and duration to make your own songs.

Experiment

Checkout the code below for some examples of how to put together a melody.

Some NOTEable tunes

Below are some pretty well known melodies to get you started. Replace the block of code that contains these variables in the example above.

Ode to Joy

Copy to clipboard

```
//All the code to make a song is in this block
int length = 15; // the number of notes
char notes[] = "eefggfedccdeedd"; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1 };
int tempo = 300;
```

Super Mario Brothers

Copy to clipboard

```
//All the code to make a song is in this block
int length = 8; // the number of notes
char notes[] = "eeecegc"; // a space represents a rest
int beats[] = { 1, 1, 2, 1, 1, 4, 1};
int tempo = 120;
```

Doorbell

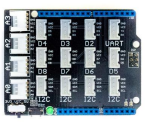
Start

This project will use the piezo buzzer and a button to create a doorbell

Modules

Gather the following parts to complete this project.

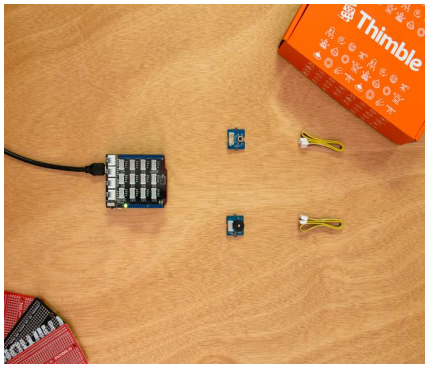
Parts



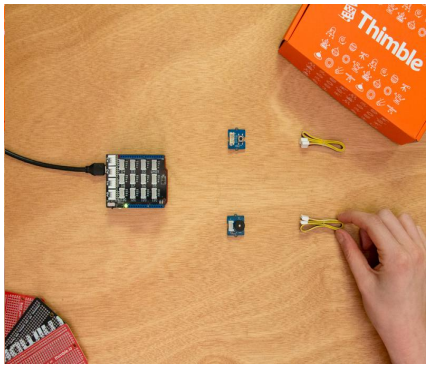

All Parts x Qty | Piezo Buzzer x 1 | Button x 1 | Cable x 2

Doorbell

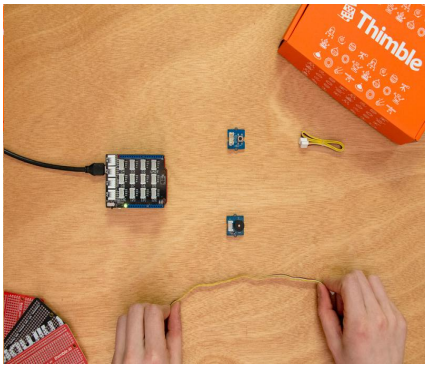
Take a cable and unwrap it. Plug one side into the buzzer socket and the other into **Digital** socket D6. Take another cable and unwrap it. Plug one side into the button socket and the other into **Digital** socket D5.



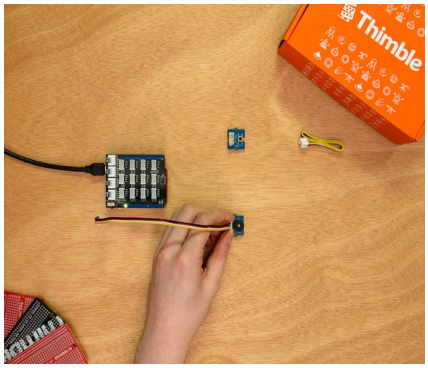
All the parts you'll need



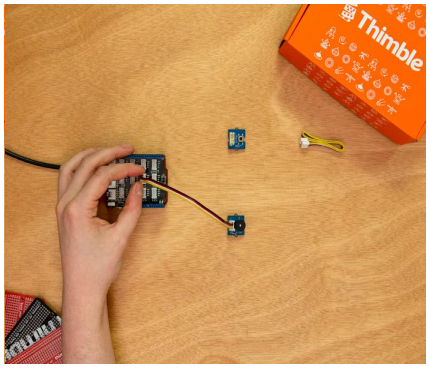
Take a cable...



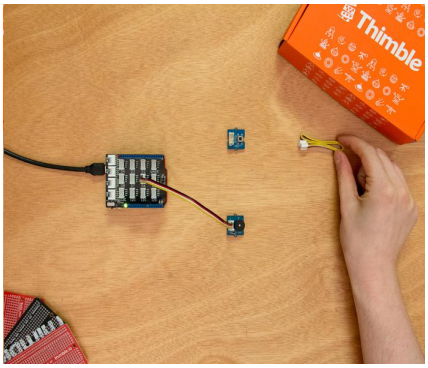
... and unwrap it



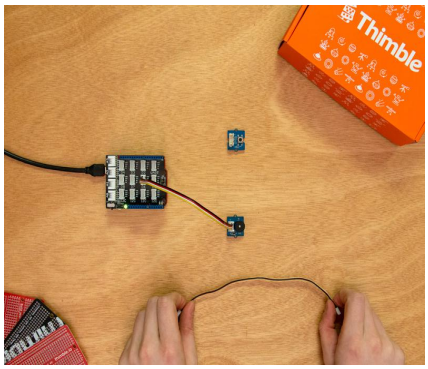
Plug one side into the buzzer socket



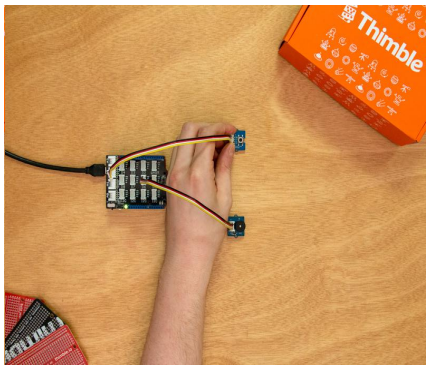
... and the other into Digital socket D5



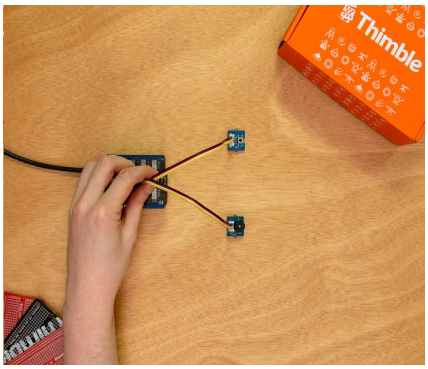
Take a cable...



... and unwrap it



Plug one side into the button socket



... and the other into Digital socket D6

Upload

Upload the code below. The example below uses the *5* and *6* digital pins. You can use any digital pin just remember to

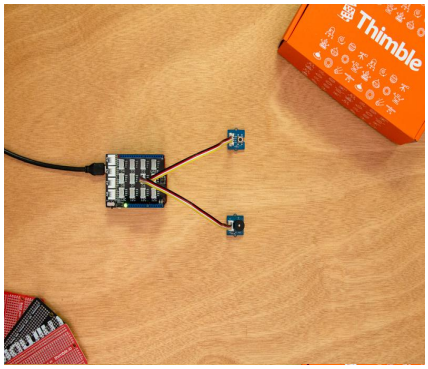
update the sketch.

Copy to clipboard

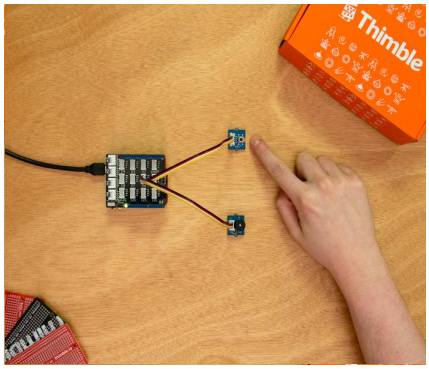
```
//If you use any different socket change them below
#define buttonSocket 5 // <- Socket for button
#define buzzerSocket 6// <- Socket for piezo buzzer

void setup()
{
  pinMode(buttonSocket, INPUT);
  pinMode(buzzerSocket, OUTPUT);
}

void loop()
{
  digitalWrite(buzzerSocket, digitalRead(buttonSocket));
}
```

Observe

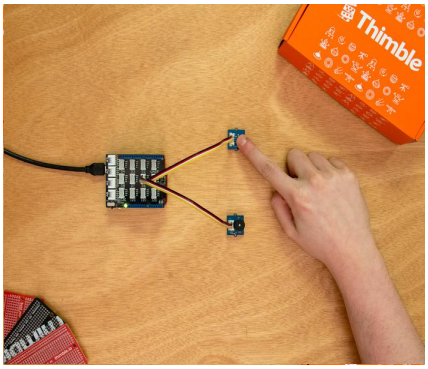
Press the button and listen to the annoying buzzer go.



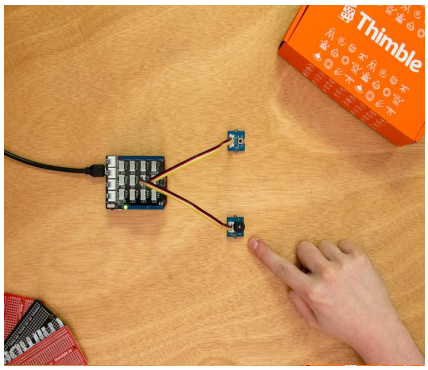
The finished doorbell



Not pressing the button and no sound



The buzzer plays once pressed



I would race to door if I kept hearing this thing

Modify

Use some tape to mount the button outside a door and the rest of the components inside.

Doorbell++

Some doorbells play a melody when rung. Buy changing just the code we can do that too.

Upload

Upload the code below. The example below uses the *5* and *6* digital pins. You can use any digital pin just remember to update the sketch.

Copy to clipboard

```
//If you use any different sockets change them below
#define buttonSocket 5 // <- Socket for button
#define buzzerSocket 6// <- Socket for piezo buzzer

void setup()
{
  pinMode(buttonSocket, INPUT);
  pinMode(buzzerSocket, OUTPUT);
}

void loop()
{
  if (digitalRead(buttonSocket)) {
    tone(buzzerSocket, 196);
    delay(600);
    tone(buzzerSocket, 329);
    delay(600);
    tone(buzzerSocket, 261);
    delay(600);
  } else {
    noTone(buzzerSocket);
  }
}
```

Observe

Now press the button. You'll hear a 3 note chime.

Modify

By changing the notes and their duration you can put together your own doorbell melodies and chimes.

Challenge
Try copying some iconic theme song into a melody that the buzzer can play.

Intruder Alarm

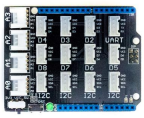
Start

This project uses the light sensor as a different kind of trigger than the Robot Buddy. Here we will use some household materials to create a door activated alarm.

Modules

Gather the following parts to complete this project.

Parts



All Parts x Qty

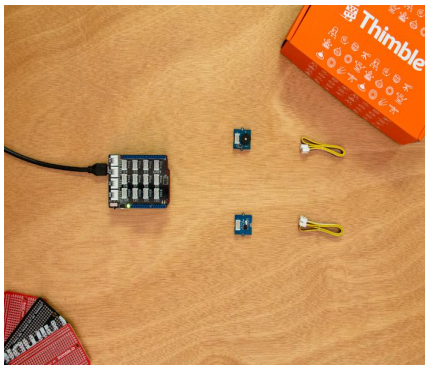
Light x 1

Piezo Buzzer x 1

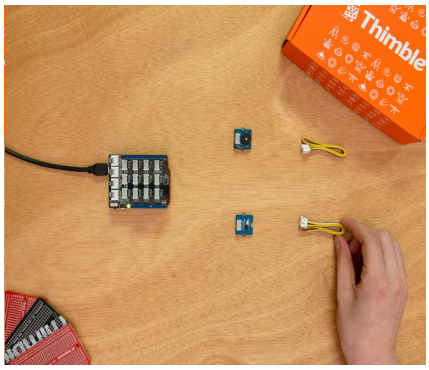
Cable x 2

Light Sensor

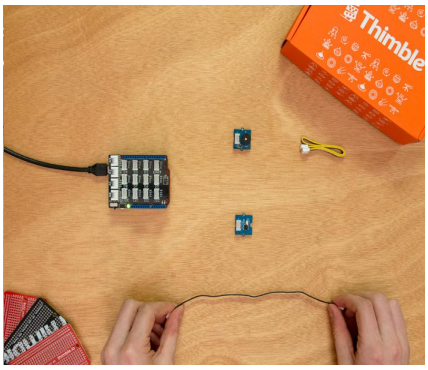
Take a cable and attach one end to the light sensor and the other to **Analog** socket A0.



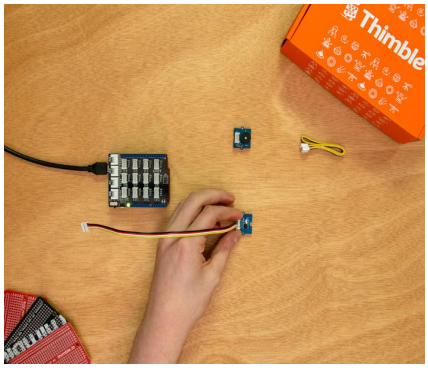
All the parts you'll need



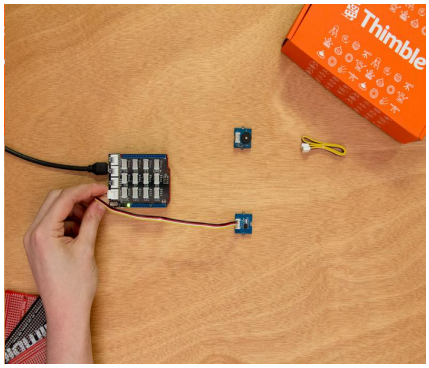
Take a cable...



... and unwrap it



Plug one side into the light sensor socket



... and the other into Analog socket A0

Upload

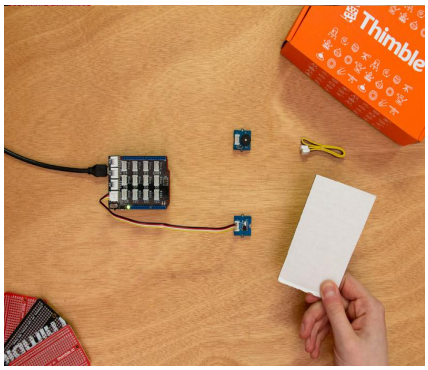
Upload the code below. This tutorial uses **Analog** socket A0. If you are using a different socket update the code after copying it.

Copy to clipboard

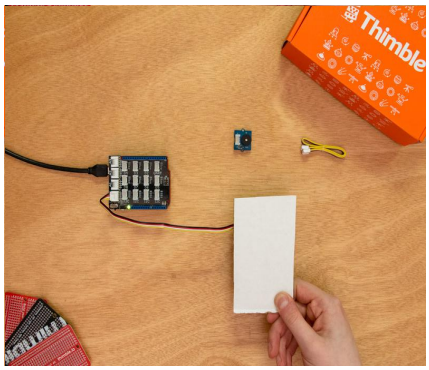
```
//Change here if you're using a different socket
#define sensorSocket A0

int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(sensorSocket);
  Serial.println(val);
}
```

Observe

Open up the Serial Plotter and move your hand or cardboard over the light sensor. Check out what values you can get. Look at the light value you get when the sensor is uncovered and covered. Remember those because we'll need them for the intruder alarm to work correctly.



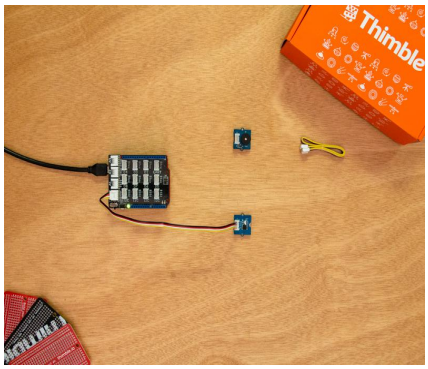
Uncovered

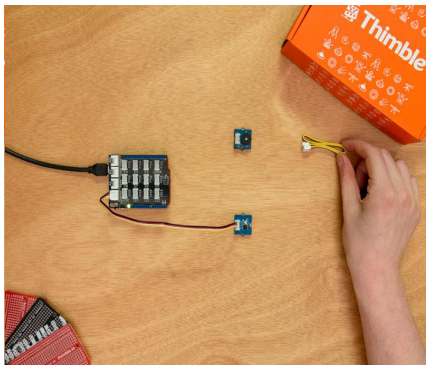


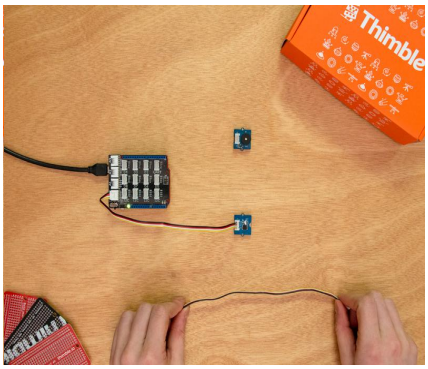
Covered

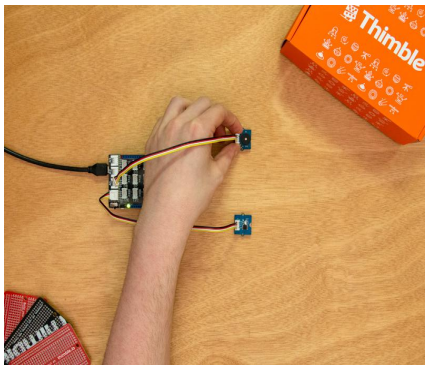
Buzzer

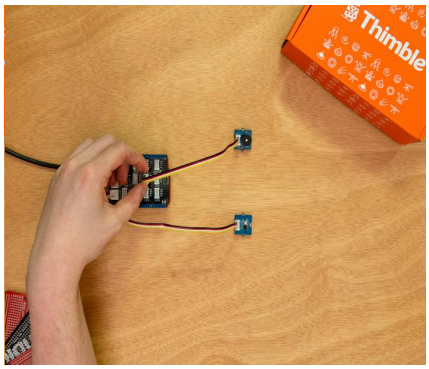
Take a cable and unwrap it. Plug one side into the buzzer socket and the other into **Digital** socket D6.


Parts


Take a cable...


... and unwrap it


Plug one side into the buzzer socket


... and the other into Digital socket D6

Upload

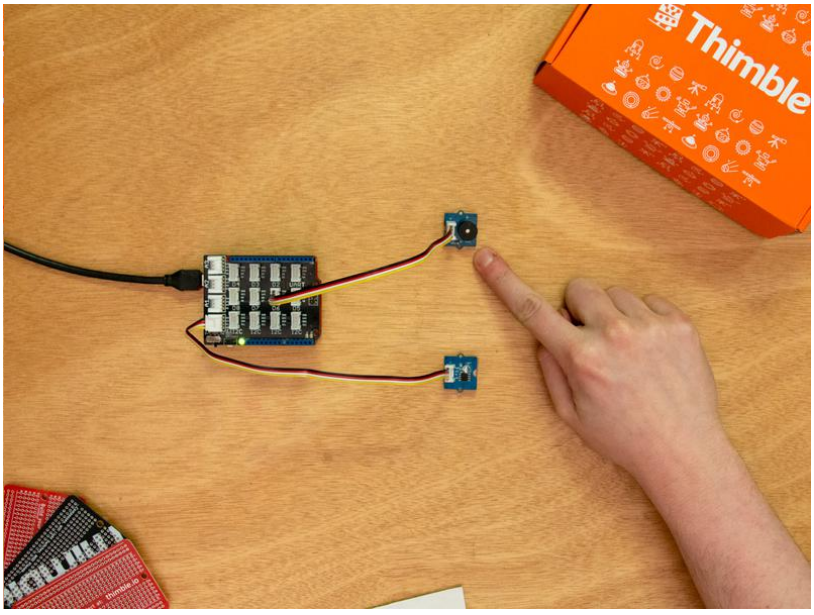
Upload this code below. This tutorial uses **Digital** socket 6. If you are using a different socket update the code after copying it.

Copy to clipboard

```
//Change here if you're using a different socket
#define buzzerSocket 6

void setup()
{
  pinMode(buzzerSocket, OUTPUT);
}

void loop()
{
  delay(2000);
  digitalWrite(buzzerSocket,HIGH);
  delay(100);
  digitalWrite(buzzerSocket,LOW);
}
```

Observe

The buzzer will wait for a second before playing a quick chirp.



Take a listen. It would be hard not to

Modify

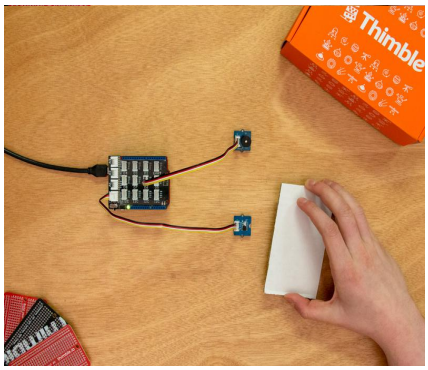
Change up the delays for a quicker or longer chirp sound.

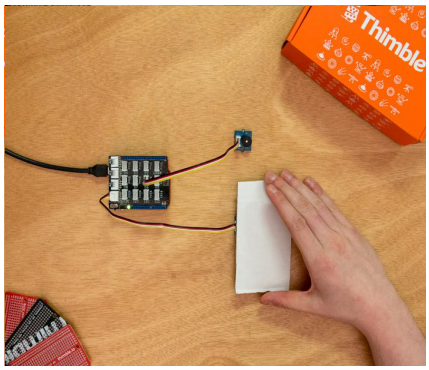
⚠This gets annoying quickly.

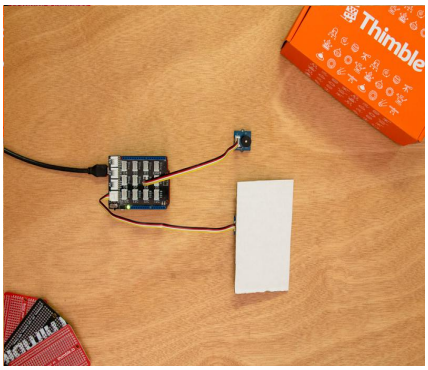
Challenge
Use a tone to make it less annoying and more melodic.

Intruder Alarm

Cover the light sensor before continuing.







Upload

Upload this code below. Add in your ambient or uncovered light value for the trigger.

Copy to clipboard

```
//If you use different sockets change them below
#define buzzerSocket 6
#define lightSensorSocket A0

int lightValue;
int trigger = 250;
int alarms = 3;
int alarmCount = 0;

void setup()
{
  pinMode(buzzerSocket, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  lightValue = analogRead(lightSensorSocket);
  //Serial.println(lightValue);
  if (lightValue > trigger) {
    while (alarms > alarmCount) {
      Serial.println(alarms);
      digitalWrite(buzzerSocket, HIGH);
      delay(200);
      digitalWrite(buzzerSocket, LOW);
      delay(500);
      alarmCount++;
    }
    alarmCount = 0;
    delay(3000);
  }
}
```

Observe

Uncover your light sensor and listen to that beautiful music? Let's transfer over to battery power with a 9 volt battery and install it.

Construction

Use a piece of cardboard taped to a door frame to cover the sensor. Attach the electronics to the wall. When the door is opened the sensor will move and no longer be covered. This triggers the alarm.