Back to full view (/modules/creator-set)

Thimble Learning Platform

Creator Set

Introduction

The Creator Set offers reusable modules each with their own unique function. There are 12 included projects but we here at Thimble know that there are much more to be made.

Objectives

- Learn the Grove/Module Ecosystem
- Explore the difference between Sensors, Indicators, and Actuators
- Learn our platform to start inventing your own projects
- Learn the basics of Arduino
- Learn the basics of coding for hardware
- Have fun!

I've Never Used an Arduino

Then this is a great place to start! If you're already familiar with setting up the Arduino and the IDE, you can skip to the next section.

Terms

When we talk about Arduino, the Arduino IDE, and coding, there are a lot of words being thrown around. This section will clear all of those up and get you ready to start building!

Arduino

The Arduino is a electronics prototyping platform. That means it's a flexible platform for building, testing, and, of course, prototyping electronics. That could be simple LEDs that light up at night, or advanced robots. When we talk

about the Arduino we are talking about the physical board itself. The one that comes in this kit is red, but they come in many different shapes and colors. The Arduino brand and original board were created by arduino.cc (http://www.arduino.cc) and you can read more about the details of its history here (https://arduinohistory.github.io/).

Arduino IDE

The Arduino IDE, or Integrated Development Environment, is not hardware, but a piece of software. If you wanted to write an essay, you might use Microsoft Word or Google Docs. If you wanted to edit pictures, you'd probably use Photoshop or Paint.net. If you want to program code for the Arduino, you'd use the Arduino IDE. Its a computer program (tool) that let's you develop and test out code. Once written, the code can be upload to the physical Arduino where it will stay until you upload new code.

Download

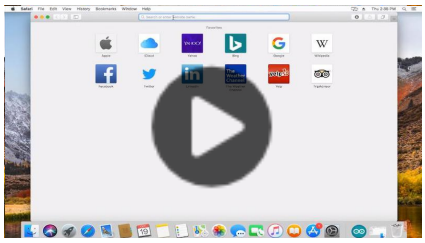
Since it is a program, you'll need to install it.

For Windows computers use this link Windows Installer (https://www.arduino.cc/download_handler.php?f=/arduino-1.8.5-windows.exe)

Windows 7 users will have an additional step at the bottom of this sections.

For Mac computers use this link Mac Installer (https://www.arduino.cc/download_handler.php?f=/arduino-1.8.5-macosx.zip)

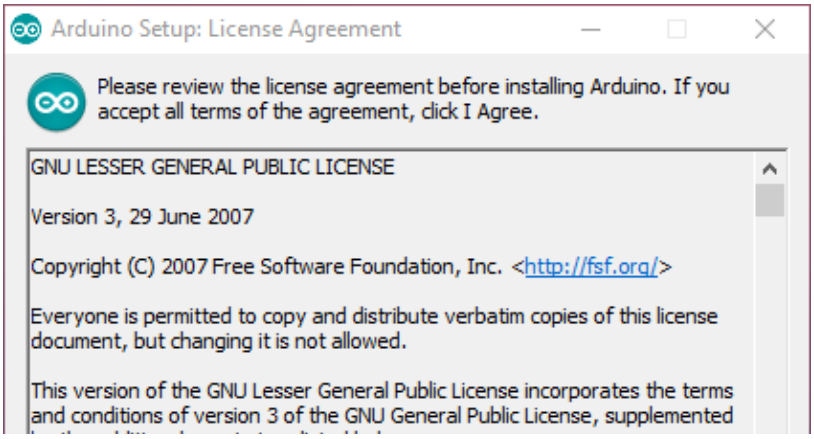
After following each link, press the **Just Download** button to begin downloading the Arduino IDE. The Mac version has a few more steps and we have a video just for that.

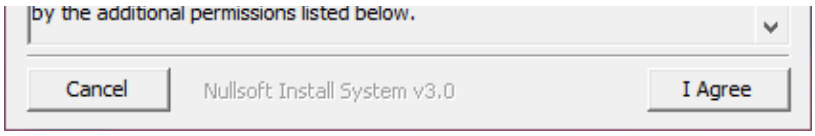


Mac Installation
For Mac users

Installation

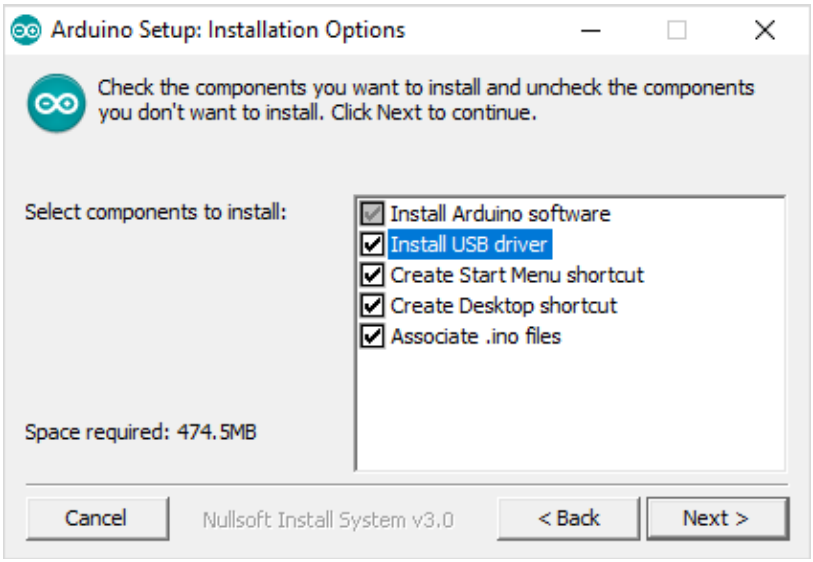
Open the executeable file you downloaded. You'll see a screen asking you to agree to their license. You can press **I Agree**.





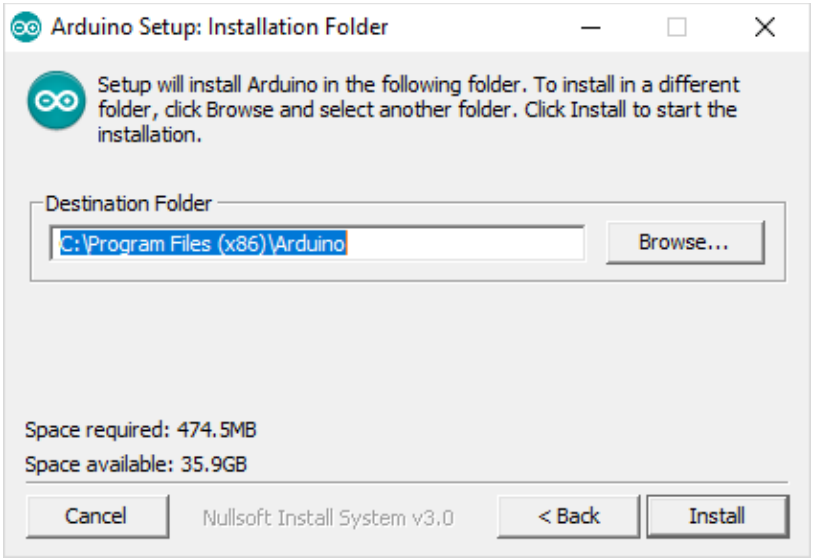
License

Next up are the installation options. The *Install USB Drivers* lets the Arduino board communicate with the IDE. We definitely want this. The *Associate .ino files* makes any Arduino file on your computer open up to the Arduino IDE. Very helpful, so we'll keep that too. Press **Next >**.



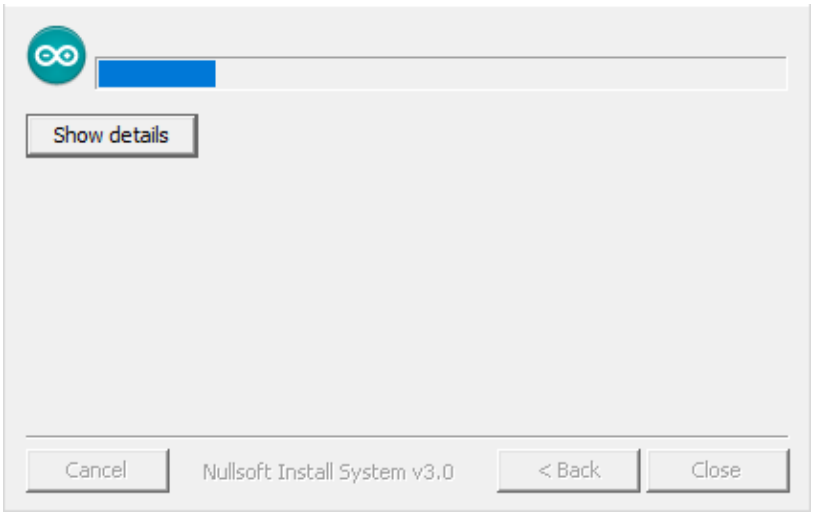
Options

Now you are being asked where you'd like the Arduino IDE program installed. I don't mind it in the standard Program Files so I just press **Install**. You can change where it will be installed, just remember what you chose. I recommend not changing it.



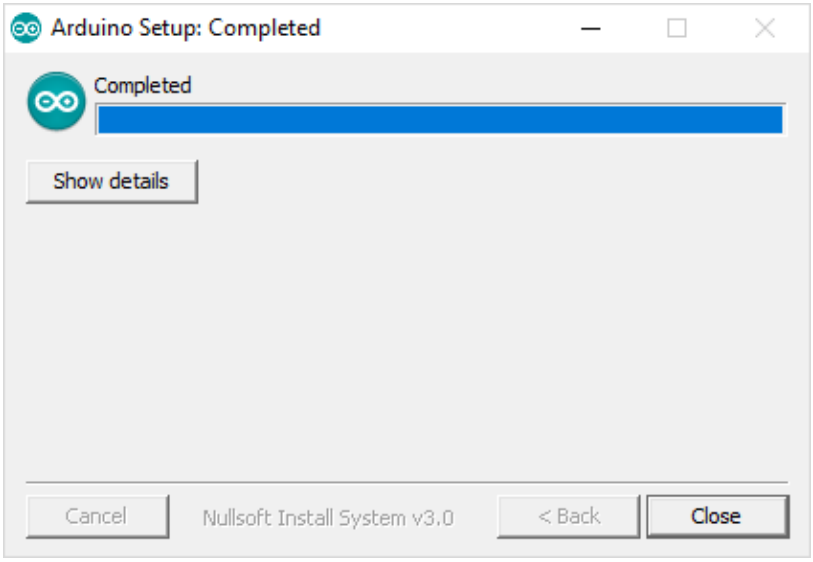
Folder

Now the IDE is installing.



Installing

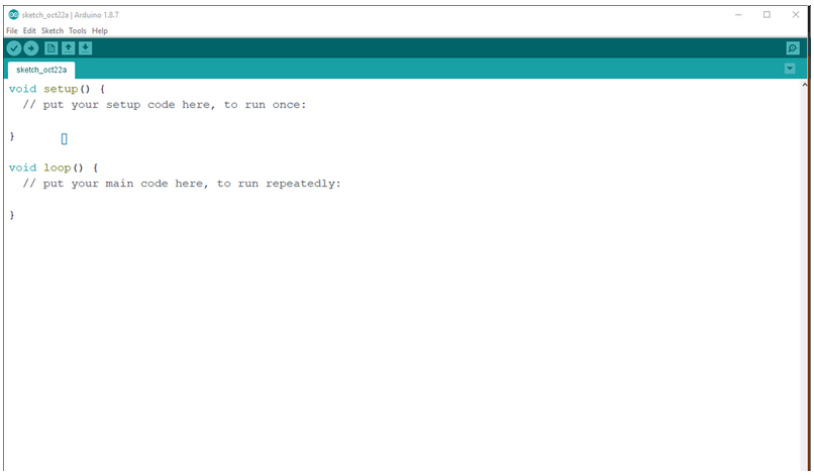
Once completed proceed to the next section.



Completed

Arduino IDE 101

Open up the Arduino IDE for the first time. So, let's figure out what's what.



Arduino IDE

There is a mostly empty space in the middle that has this text:

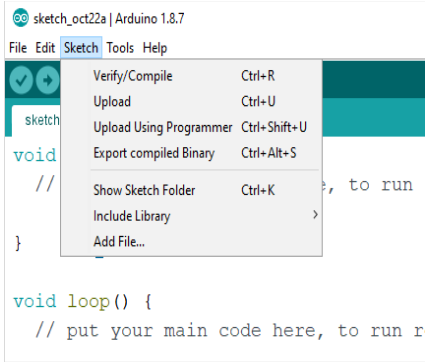
Copy to clipboard

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

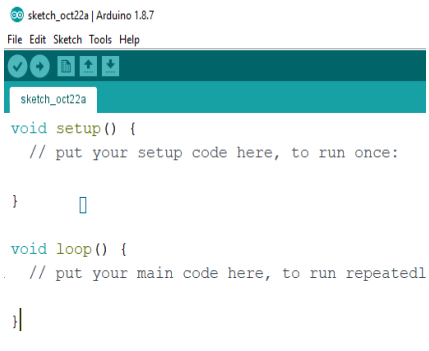
This is a simple Arduino Sketch. A Sketch is code you write that will later be uploaded to the Arduino board.

There are few steps to go from code written in a sketch to code that runs on the Arduino board. Those are **Verify > Compile > Upload**.

All of these you can get to by going under **Sketch** on the top bar of the program. Or using the Checkmark and Arrow icons right below that. The Checkmark verifies your code will compile, and the arrow button uploads the code to the Arduino.



Sketch



Icons

Verify - looks at your code and makes sure it can actually run. It checks that there are no errors or things out of place. If your code will work, it goes to the **Compile** stage.

Compile - The code you've written is great for human. It is readable and hopefully understandable. But this isn't great for a machine ie the Arduino. The **Compile** stage takes your written code and translates into something that the Arduino can understand and run, called machine code. It saves that compiled file and tried to **Upload** it to the board.

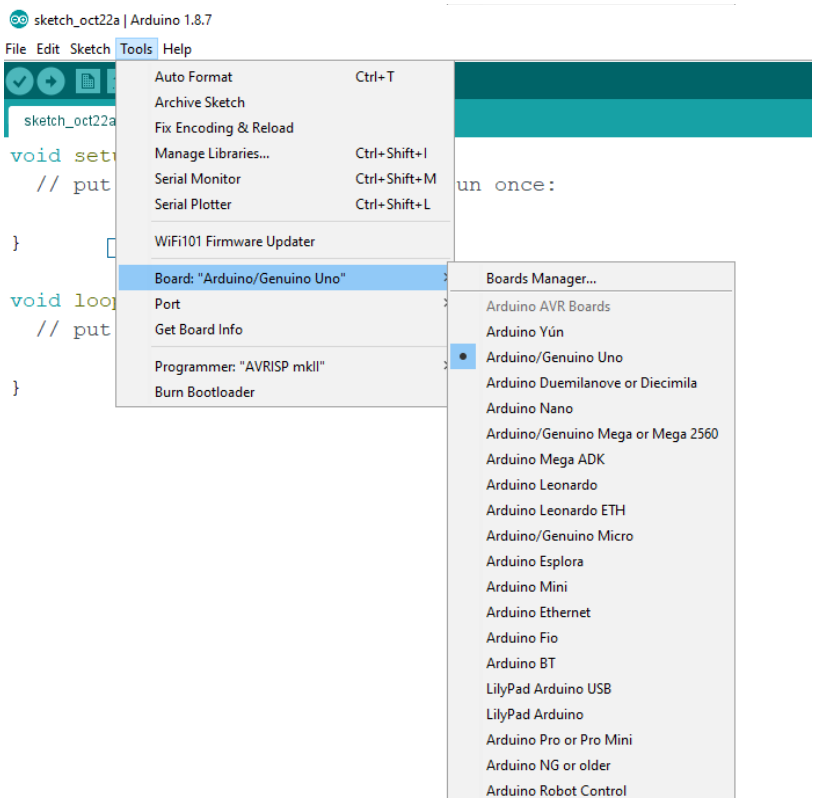
Upload - This takes the compiled files and tries to send it to a plugged in Arduino. The IDE doesn't know where you've plugged in your Arduino or what kind it is. You have to tell it. I'll show you how to set that up in the next section.

Arduino Board Setup

For this section, make sure your Arduino Board that came with your kit is plugged in. Use the included micro USB cable to plug your Arduino into any free USB port on you computer.

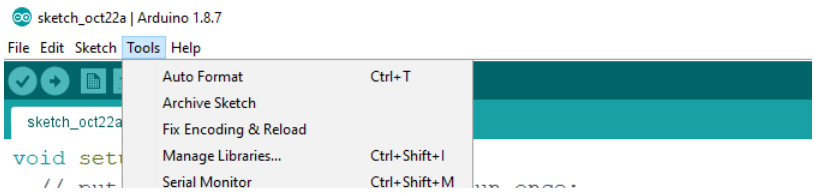
Navigate to the **Tools** section at the top of the Arduino IDE.

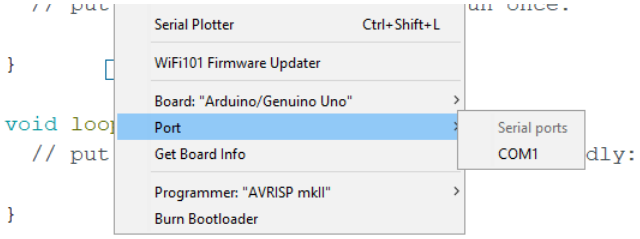
You'll be greeted with a scary drop down. To setup your Arduino to accept uploads, do the following: Go to **Board:** and make sure *Arduino/Genuino Uno* is selected.



Board

Next is the **Port**. For Mac users refer to the video for these instructions. With your Arduino plugged in, you'll see some options. Mine says *COM 15*. This is the channel that the IDE will use to communicate with your Arduino. If you have a lot of COM # devices and don't know which is your Arduino, unplug your Arduino and take note of that numbers are there. Then plug it back in. Whichever is the new one in the list is your Arduino. Select that port.





Port

With the board and port selected and ready to go, let's upload our first sketch.

Sketches

We are going to upload a simple sketch to make an LED on the board blink. Head to **File > Examples > Basics > Blink** . You should see this sketch.

Copy to clipboard

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Don't worry about the coding for now. Let's upload this sketch. You can press the Arrow icon, go to **Sketch** then *Upload*, or you can use the shortcut **Ctrl+U** in order to upload the code. Your sketch will auto-verify, compile, then upload to your Arduino board.

```
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Upload the Sketch

You will see the message *Done uploading* and your board should now have a blinking light.

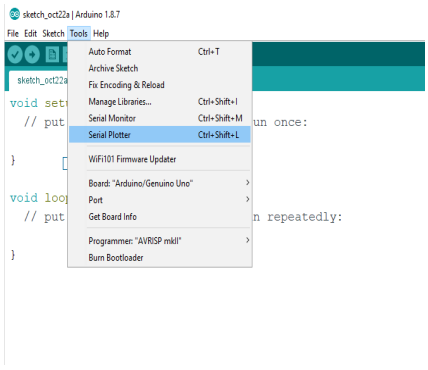
Uploading Sketches and Code From Thimble

⚠These instructions are very important for getting OUR code to work for YOU.

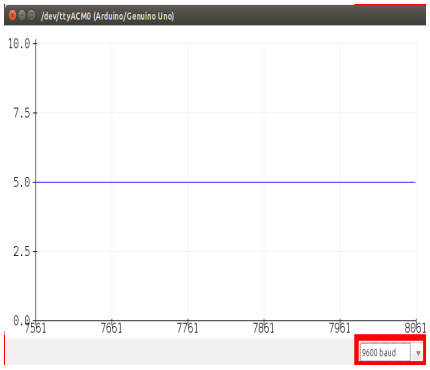
We provide example code for all our projects. To use those you'll need to copy them from our website. Then, go into the Arduino IDE, **File > New**. It's best to then delete all the code already there in the sketch before you paste our code in, as that will avoid any errors from copying over the old code.

Serial Plotter

You'll be asked to use the **Serial Plotter** on some projects. This reads any information coming from the Arduino and displays it on a graph. It can only be accessed **after** you've uploaded your code. It is in the **Tools** menu. Once opened, make sure the `Baud rate` is set to 9600.



Where to find the Serial Plotter



Baud rate

Troubleshooting

The most common issues are solved by double checking your **Board** and **Port** settings.

Attention Windows 7 Users

Windows 7 needs to have the Arduino drivers installed manually. Follow the instructions at the link below.

Arduino Drivers (http://wiki.seeedstudio.com/Seeeduino_v4.2/#software)

Libraries

What are Libraries?

Libraries are a reusable collection of code that has already been written. We use them when we want to add functionality without have to write everything from scratch. The advantage of using a library is code re-use.

Some of the projects in this set need libraries so this section will show you how to install those.

Libraries to be downloaded

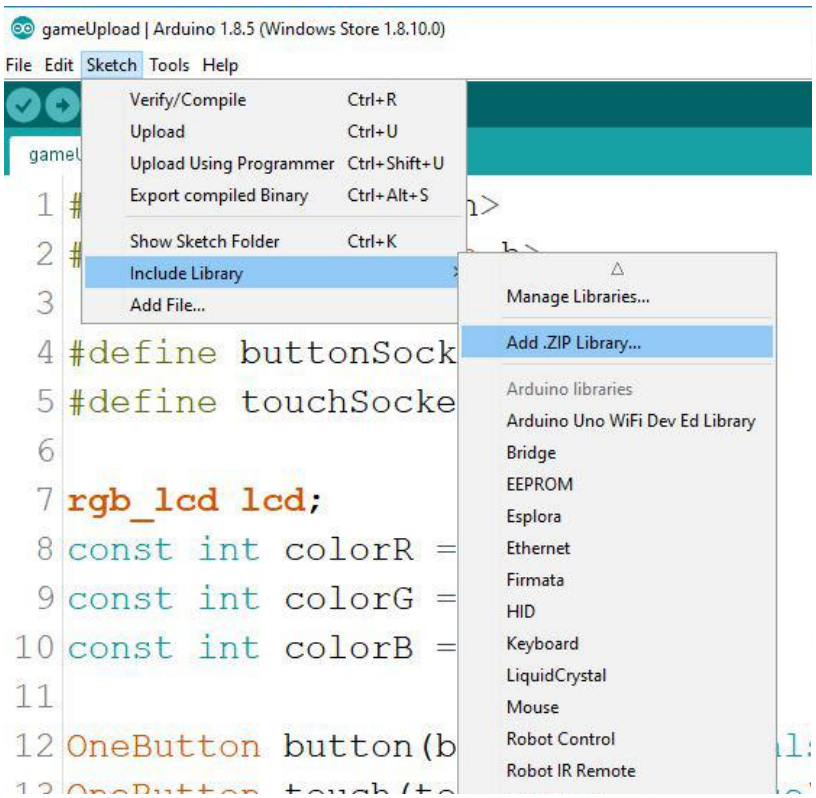
Our libraries come in the form of zip files that need to be downloaded, but **NOT** unzipped.

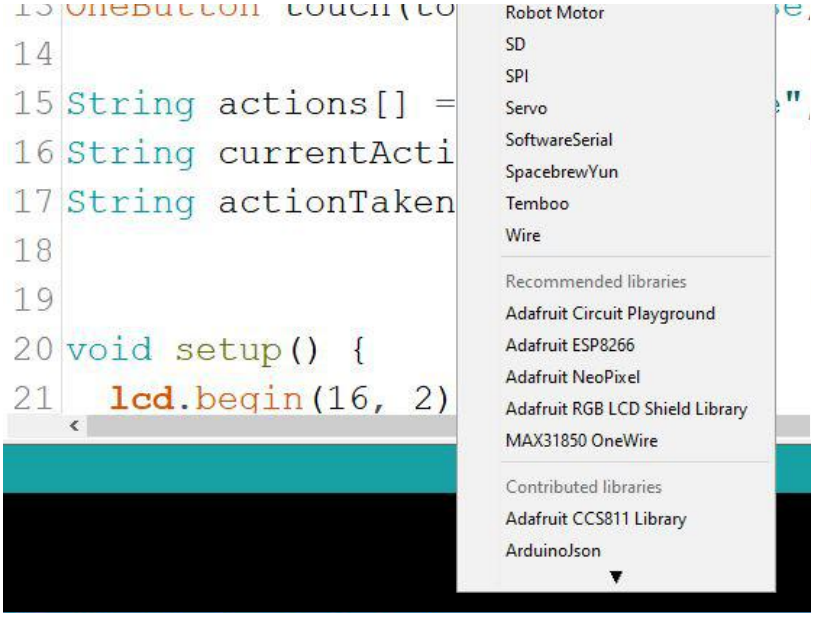
LCD Display Library (https://d2j2m4p6r3pg95.cloudfront.net/module_files/creator-set/assets/libraries/lcdDisplay.zip) - This is for the LCD Display and all its colors

OneButton (https://d2j2m4p6r3pg95.cloudfront.net/module_files/creator-set/assets/libraries/OneButton.zip) - This is to add some functionality to the buttons found in the set

Adding Them to the Arduino IDE

Open up the Arduino IDE and go to **Sketch > Include Library > Add .ZIP Library...**. After that navigate to the zip files you downloaded and add them one by one. And that's it!





Add a Library

Creator Set - Grove

What are These Things in My Set?

When you open up the Creator Set, you'll see a bunch of modules. Each has a certain function. Some are Sensors that measure the environment. That could be the amount of light or sound, for example. Some are Indicators, which show a state or information. These could be an LED or a display. The last kind are Actuators, that interact with the environment. Actuators are things like motors that move or spin.

Why Use These?

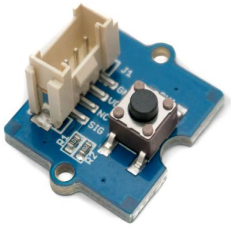
These modules are reusable. You can use them in one project and then reuse them in another one. They are much harder to break then the average electrical component. They share a connector that is universal so a cable for one module works for all the rest.

We want future projects to be more hackable and tinker friendly and using these modules is the answer.

What Modules Do I Get?

I'm glad you asked!

⚠If you have a Creator Set with a green box then it also included a relay, seen below. A projects that would have used the relay usually have you cutting and stripping wires that would plug into the wall. Since that's not a beginnier task, the relay was removed and replaced with a WiFi Module. Have no fear, with either kit you can complete all 12 projects.



A Button

A Piezo Buzzer

An LED Driver

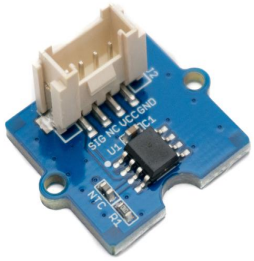
With 3 LED colors

A Light Sensor

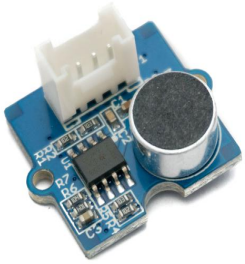
A Rotary Potentiometer

A Servo

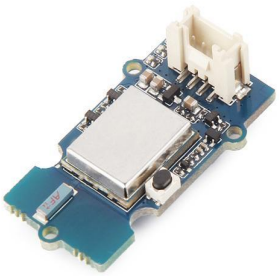
A Capacitive Touch Sensor



A Temperature Sensor



A Sound Sensor



A WiFi Module. Looks may vary but on the back it will say WiFi no matter the model.

An LCD Display

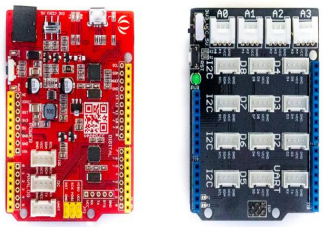
How Do I Use These?

There is one step you have to do before getting started with the 12 projects. Open up your kit. Remove the LCD Display at the top and you'll find a board protected by pink foam. This board is called the base shield. Remove the base shield from the foam and put it on top of your Arduino. There are pins on the shield that will match up and slide into the Arduino. This lets us use all the cool features of the modules with a normal Arduino. Now you're ready to get started.

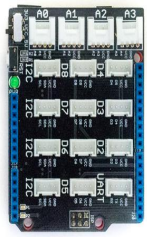
The Creator Set

Opened

Take the LCD and foamed board



Base Shield and Arduino



Base shield installed on top of the Arduino

So What Are These Modules Again?

Hey You

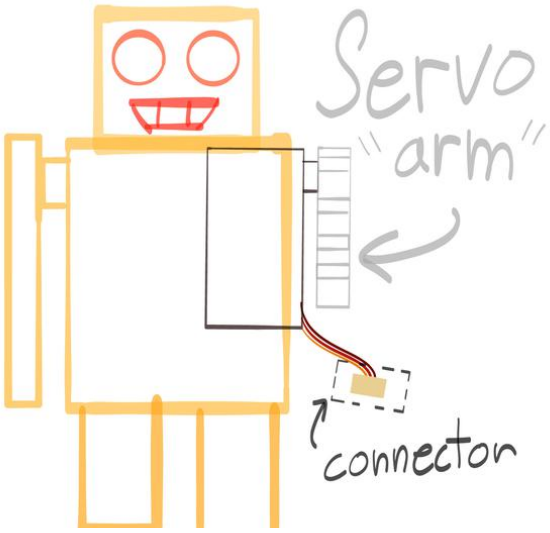
We want your feedback. It helps us know what's working and what's not. Everyone here at Thimble wants to teach Electronics and Programming the best that we can and your feedback helps us do that. While you build these projects take some notes about what was cool and what could use some work. Post on the forums with your experiences and hacks.

Happy Hacking!

Robot Friend

Start

Let's build the first project in your 12-in-1 project kit. A Robot friend that reacts to your environment. He'll end up looking something like this.

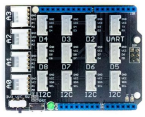


Robot Friend Blueprint

Modules

Gather the following parts to complete this project.

Parts



All Parts x Qty

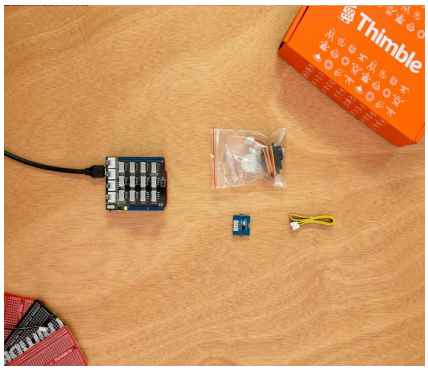
Servo x 1

Light x 1

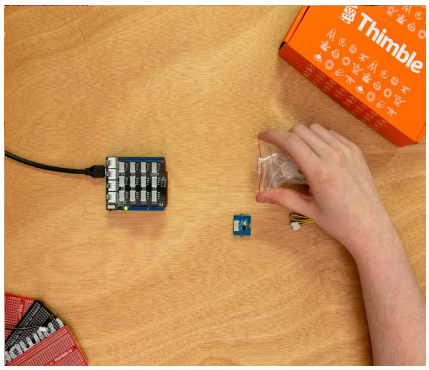
Cable x 1

Servo

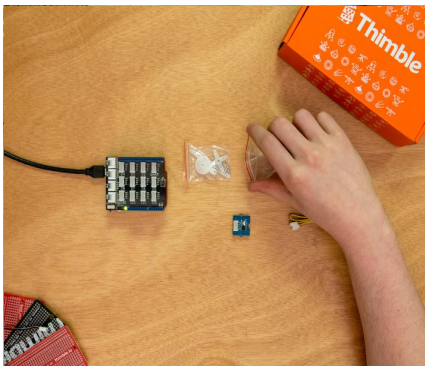
The servo comes in a bag that also has a smaller bag of components in it. You'll need both for this project. Take the servo out as well as the small servo arm and a screw for it. One of the longer screws is better. Take the servo arm and servo and connect them. The arm should be pointing up with none of it hanging off the servo body. After unwrapping the servo cable, plug it into **Digital** socket D6.



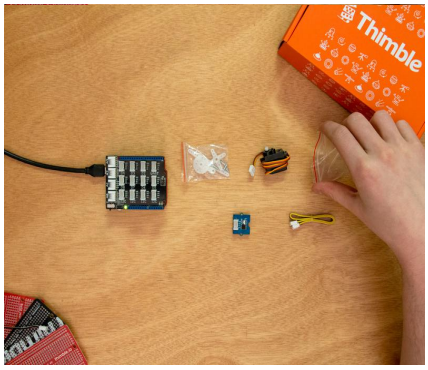
All the parts you'll need



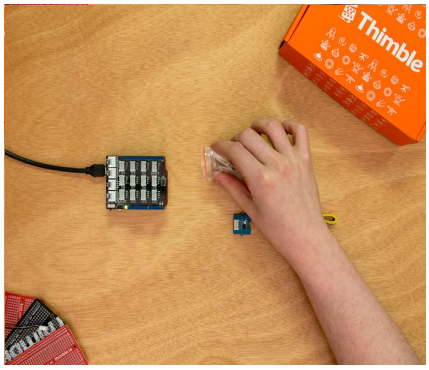
Open the plastic bag



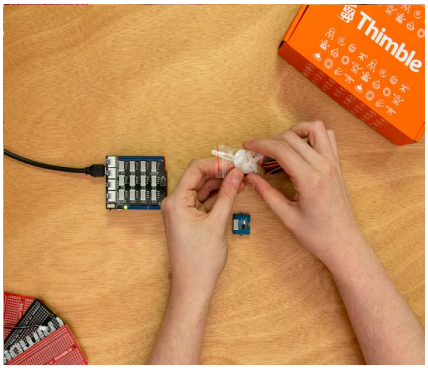
Take out the bag of parts inside...



... and the servo

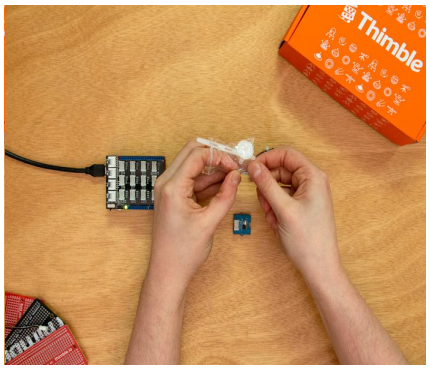


Open the smaller bag...



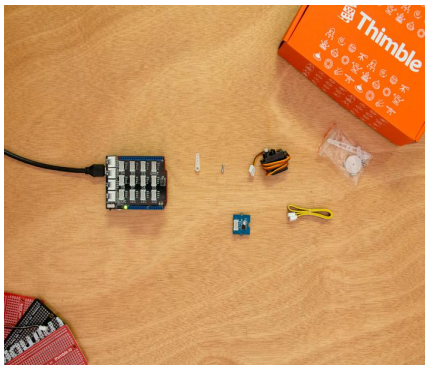
Be careful! There are many little parts

Take out the small 'Servo Arm'...

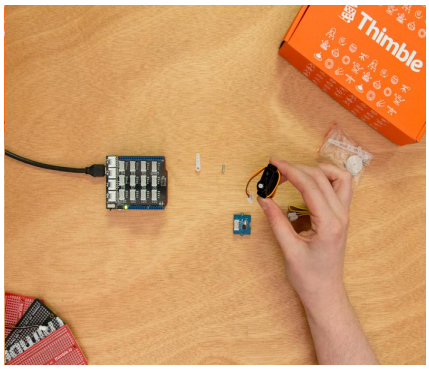


... and ...

... one longer screw



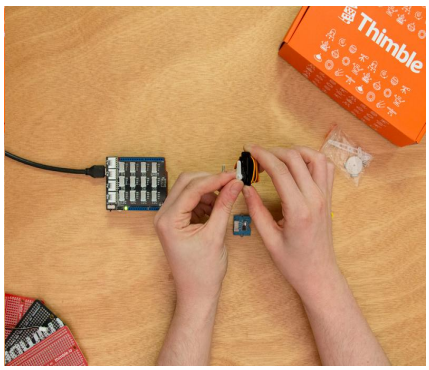
Now we can get started



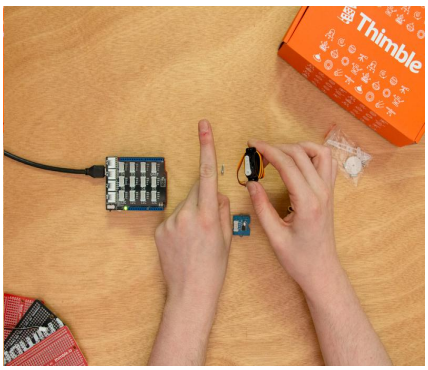
Take the servo...

... and the 'Servo Arm'

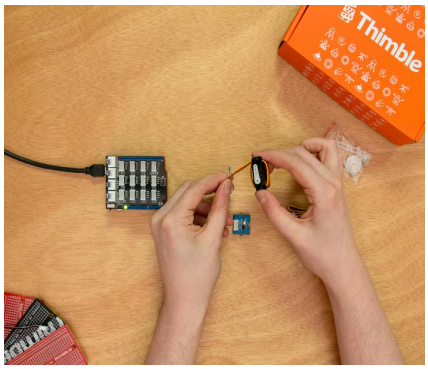
Line up the circles



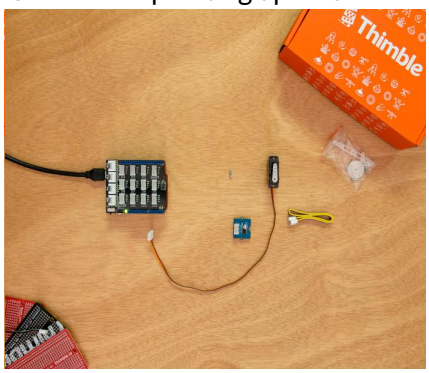
Place them together with the 'Servo Arm' pointing up on the



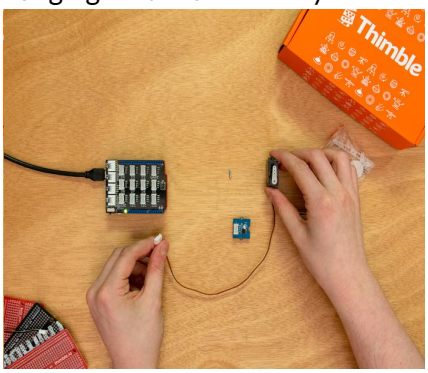
The 'Servo Arm' shouldn't be hanging off the servo body



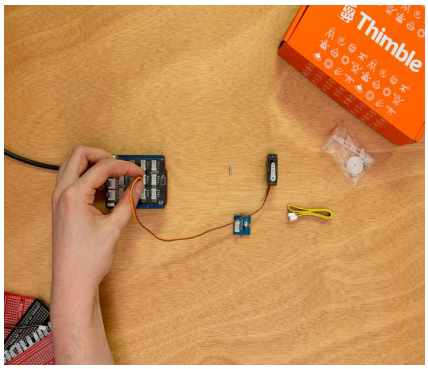
Take the wire around the servo...



... and unwrap it



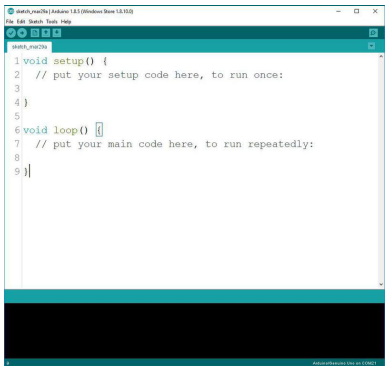
Take the cable ...



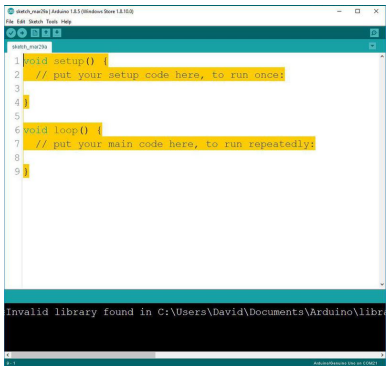
... plug into Digital socket D6

How To Upload These Examples

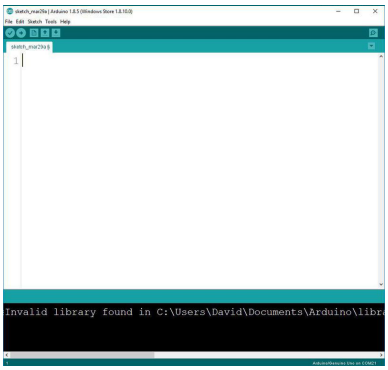
This is the procedure for each **Upload** section within each tutorial. First, open a new Arduino sketch in the Arduino IDE. Next, select all of the text and delete it so the sketch is now completely blank. Switch back over to this Learning Module and press **Copy to Clipboard**. Go back to the Arduino IDE and paste the code you just copied into the sketch. Before uploading, check in the **Tools** menu on top that 1) Board is selected as **Arduino/Genuino UNO** and 2) your port is correct for your computer, then press upload. A popup will ask you to save. Change the name to whatever project you're working on and press save. The code will now upload to your Arduino Board.



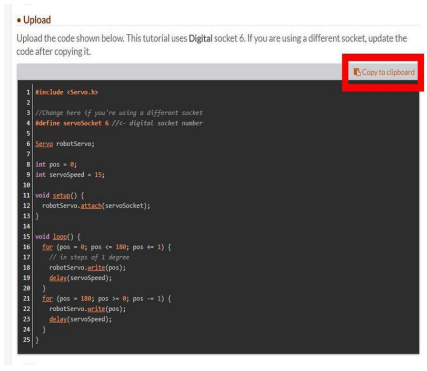
Open a new sketch



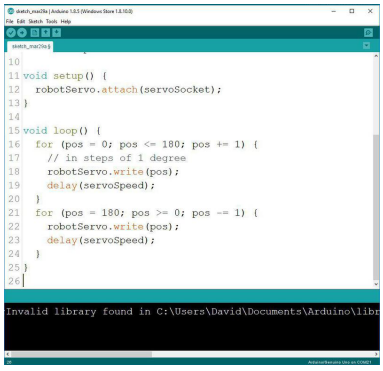
Highlight all code and delete it



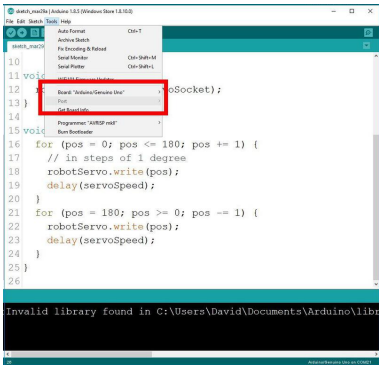
A blank sketch



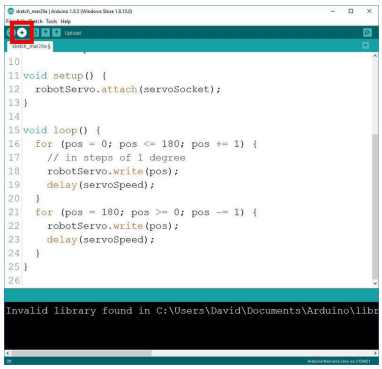
Copy code from our website



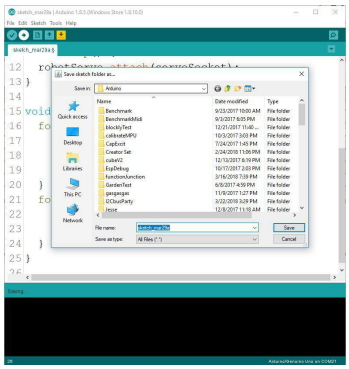
Paste that into the blank sketch



Double check



Press Upload



Save your sketch

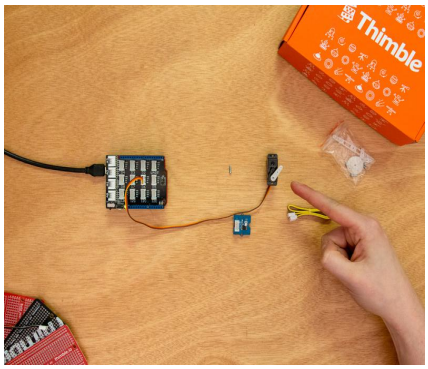
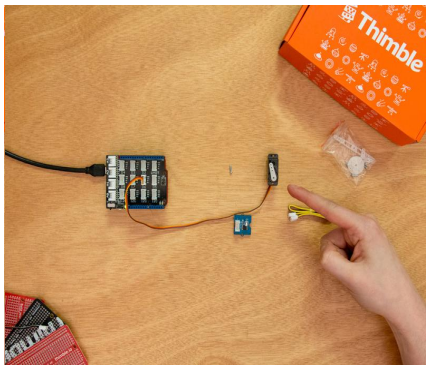
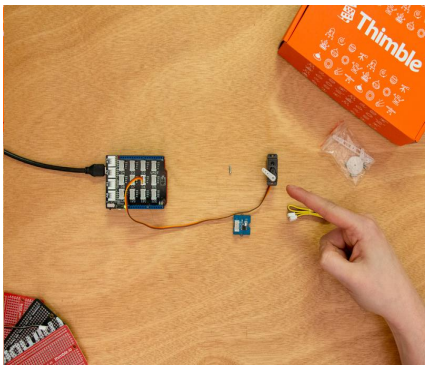
Upload

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

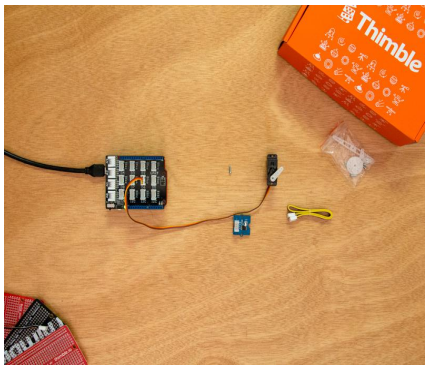
Copy to clipboard

```
#include <Servo.h>

//Change here if you're using a different socket
#define servoSocket 6 //<- digital socket number

Servo robotServo;

int pos = 0;
int servoSpeed = 15;

void setup() {
  robotServo.attach(servoSocket);
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    // in steps of 1 degree
    robotServo.write(pos);
    delay(servoSpeed);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    robotServo.write(pos);
    delay(servoSpeed);
  }
}
```

Observe

Watch the servo move back and forth



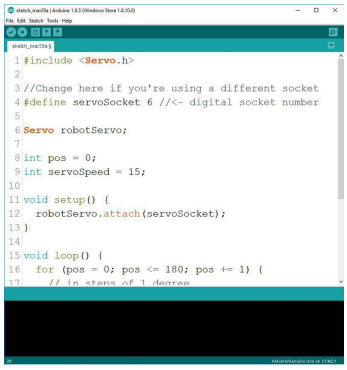
Look at how the servo swings back and forth. There are two loops in the code. One for swinging clockwise and the other for counter-clockwise.

Modify

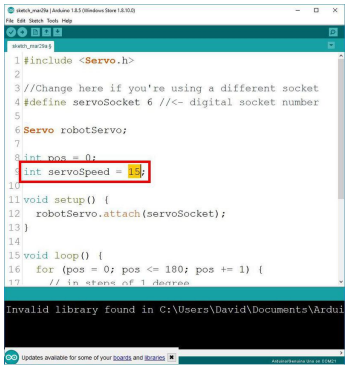
Increase or decrease the `servoSpeed` variable and watch what changes. It might not do what you'd think.

How to Modify Code

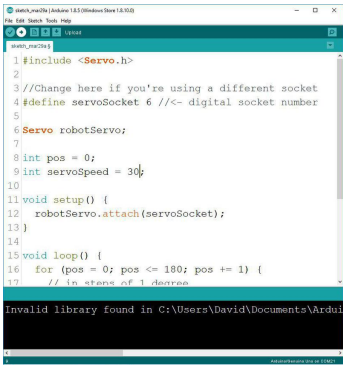
All code modifications will be done via variables that are at the top of the sketch. In the copied over Arduino sketch you can change the value of variables, upload the code, and observe what effect your change had.



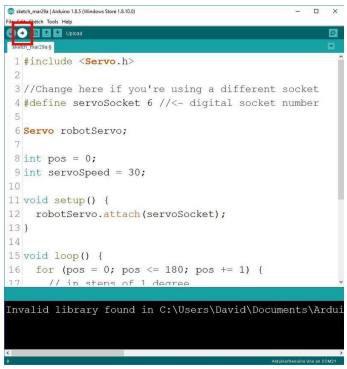
The prewritten sketch



Variable to change



Changes variable



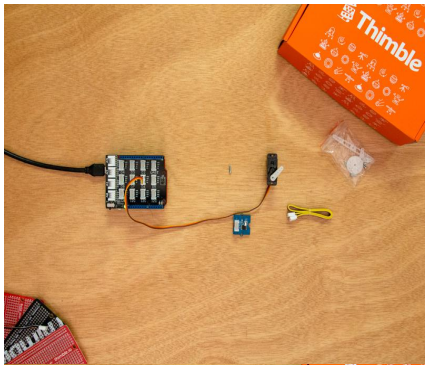
Upload and observe

Experiment

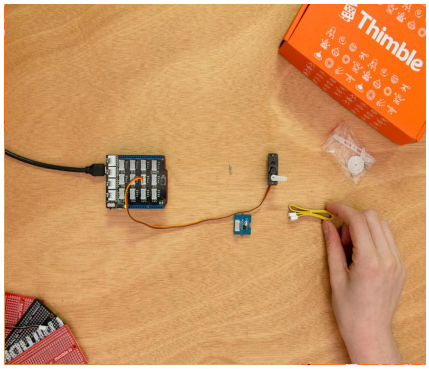
Play with that variable to make the servo go faster or slower.

Light Sensor

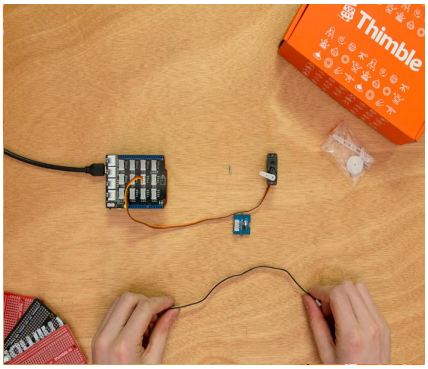
Take a cable and attach one end to the light sensor and the other to an **Analog** socket, A0.



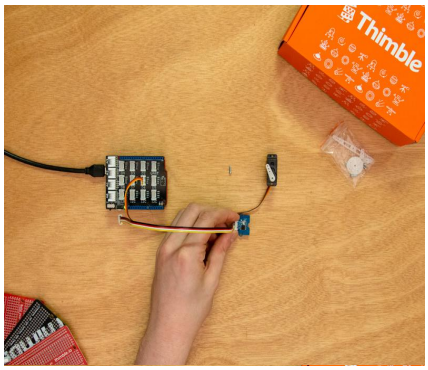
All the parts for the next step



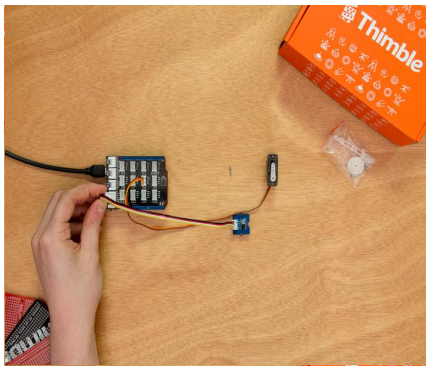
Take a cable



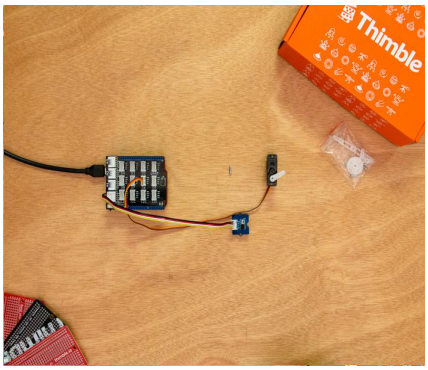
Unwrap it



Connect one side to the sensor socket



The other to Analog socket A0



Ready to upload code

Upload

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

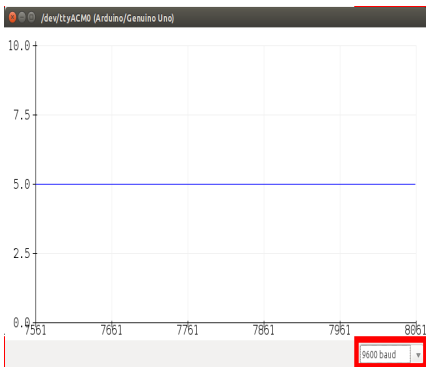
Copy to clipboard

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

int val;

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

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

Observe

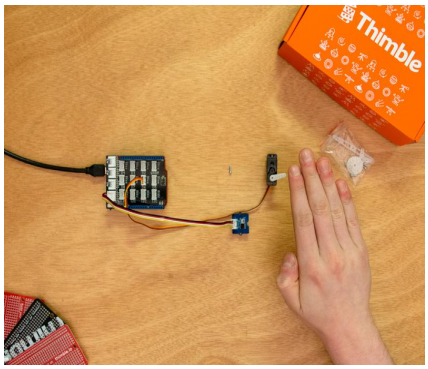
You'll be asked to use the **Serial Plotter** on some projects. This reads any information coming from the Arduino and displays it on a graph. It can only be accessed **after** you've uploaded your code. It is in the **Tools** menu. Once opened, make sure the `Baud rate` is set to 9600.

Where to find the Serial Plotter

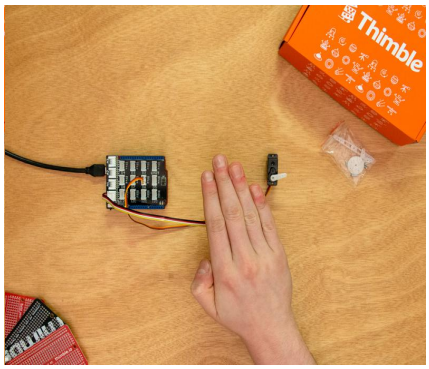


Baud rate

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. Remember those because we'll need them for your Robot Friend.



Record uncovered light value



And covered value

Robot Buddy

Let's finish up the code for the first project blueprint.

Upload

Upload the code shown below. The variable trigger value should be a number close to the covered sensor light value. You'll want it a bit (ie: 10-30) higher than what you measured. For example, if you measured 80, 100 would be a good value to use.

Copy to clipboard

```
#include <Servo.h>

#define servoSocket 6
#define lightSensorSocket A0

Servo robotArm;

int pos;
int val;
int trigger = 100; //<- Change to YOUR measured value
int waveSpeed = 5;

int leftWave = 90;
int rightWave = 0;

void setup() {
  robotArm.attach(servoSocket);
}

void loop() {
  val = analogRead(lightSensorSocket);
  if (val < trigger) {
    for (pos = rightWave; pos <= leftWave; pos += 1) {
      robotArm.write(pos);
      delay(waveSpeed);
    }
    for (pos = leftWave; pos >= rightWave; pos -= 1) {
      robotArm.write(pos);
      delay(waveSpeed);
    }
    for (pos = rightWave; pos <= leftWave; pos += 1) {
      robotArm.write(pos);
      delay(waveSpeed);
    }
  }
}
```

Observe

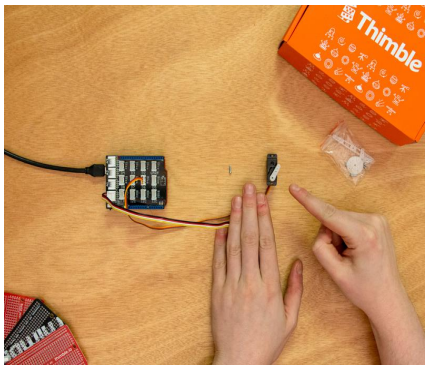
Put your hand over the light sensor and watch the servo move. The change in light triggers the servo to move back and forth, or *wave* to you.

Modify

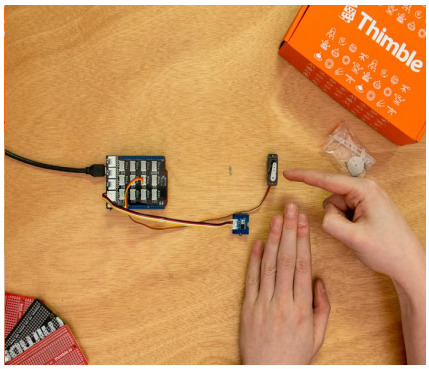
Increasing the `trigger` will make it more sensitive to the change in light.

Experiment

Changing the `leftWave` and `rightWave` variables will change how far the Robot waves. `waveSpeed` changes how



The arm moves when covered



And not when exposed to light

fast the Robot waves.

Robot Template

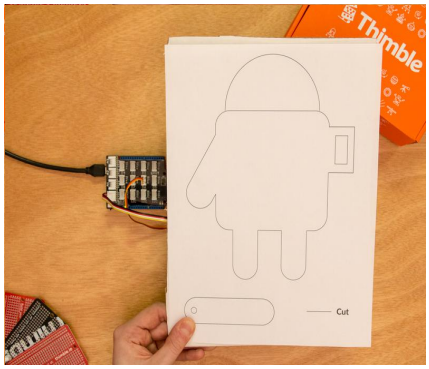
Recommended - PDF Template (https://d2j2m4p6r3pg95.cloudfront.net/module_files/creator-set/assets/templates/RobotFriendTemplate.pdf)

SVG Template (https://d2j2m4p6r3pg95.cloudfront.net/module_files/creator-set/assets/templates/RobotFriendTemplate.svg)

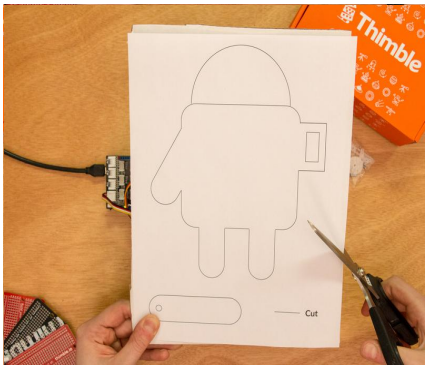
AI Template (https://d2j2m4p6r3pg95.cloudfront.net/module_files/creator-set/assets/templates/RobotFriendTemplate.ai)

Print out the template and assemble your new buddy! Find some cardboard and cut out the template. Disconnect the servo from it's socket. Place it through the rectangle hole in the Robot's missing arm. Seat the servo and use some tape to keep it in place. Place another piece of tape behind the servo arm then attach the Robot's arm. Use the screw from the beginning and screw it into the hole in the arm and into the servo. With the construction complete, plug the servo back into the same socket. Wave at your Robot and watch it wave back!

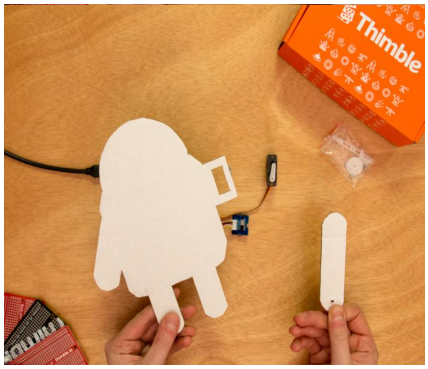
Some cardboard and the printed template



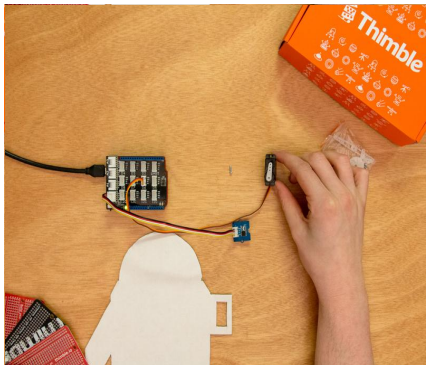
Line up your materials



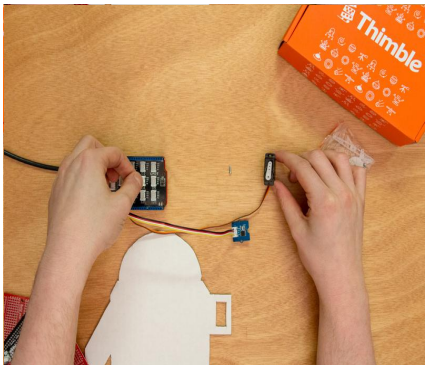
Cut the solid lines



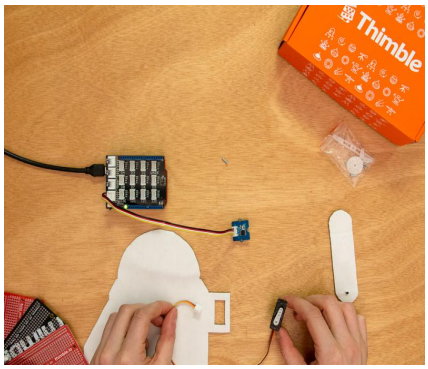
All cut out

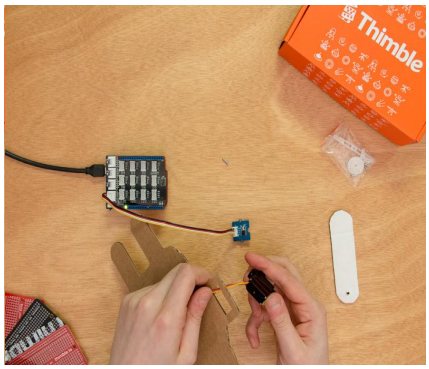


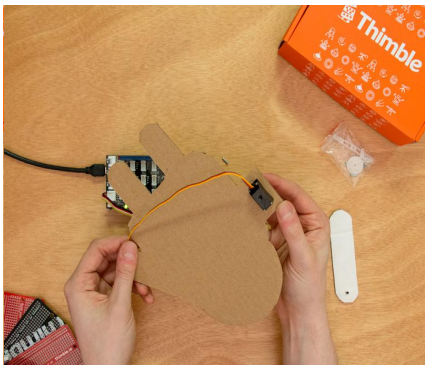
Take the servo



Disconnect it from the socket

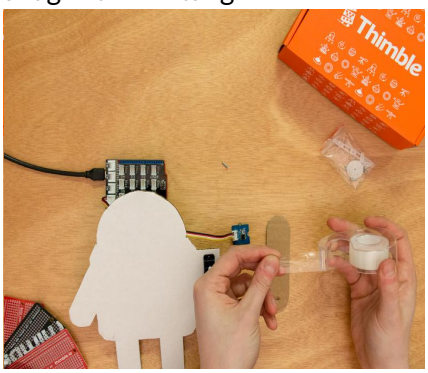

Grabbing the socket end...

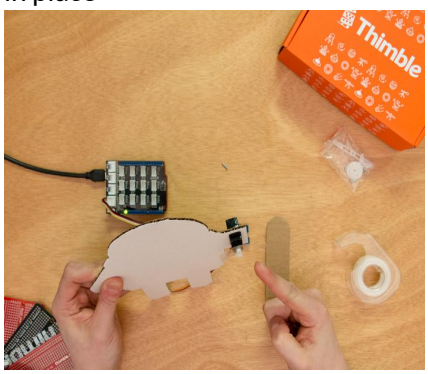

... stick it through the Robot's missing arm

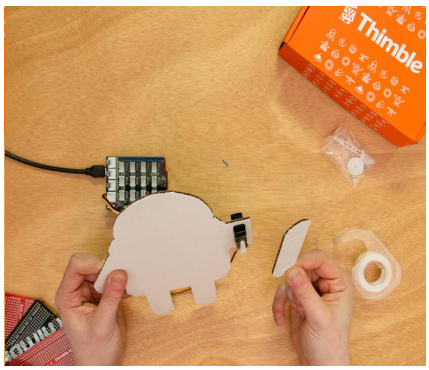

All the way through so the servo is snug in the rectangle hole

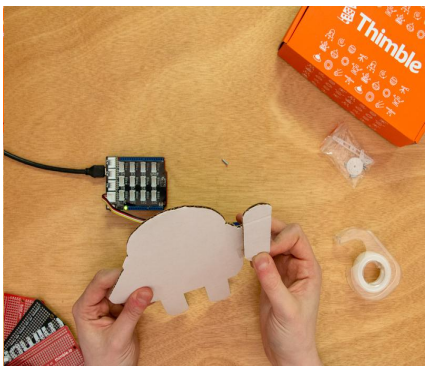

You can add some tape to keep it in place

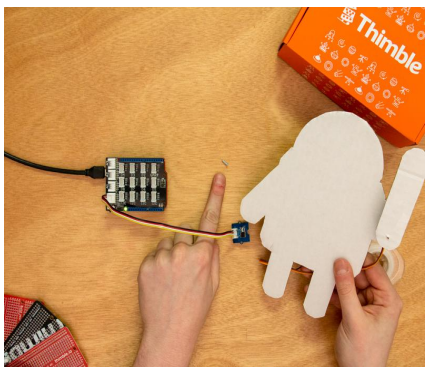

Now we'll attach the arm

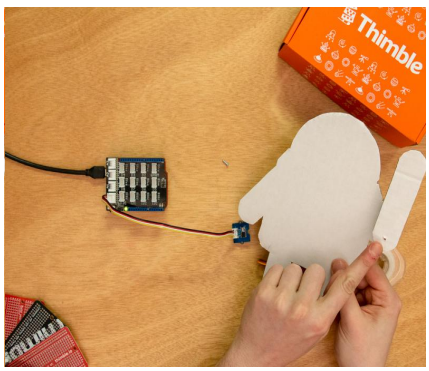

Get a piece of tape

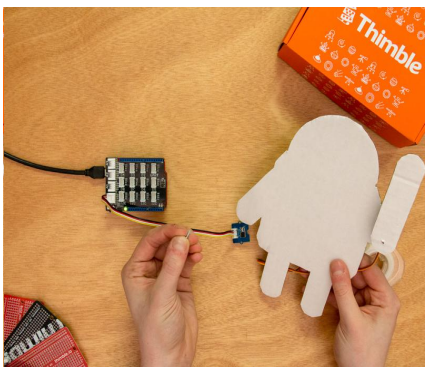

Attach it behind the 'Servo Arm'


Get the Robot's arm...

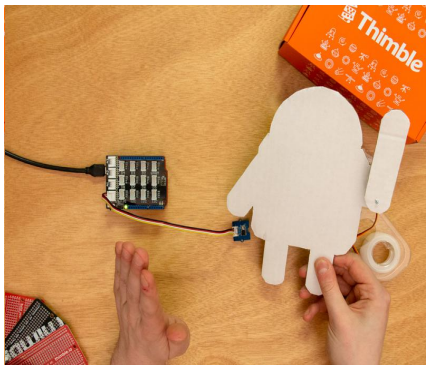

... place it on top of the 'Servo Arm'


Find that screw from earlier


It will go through the Robot's arm and into the servo

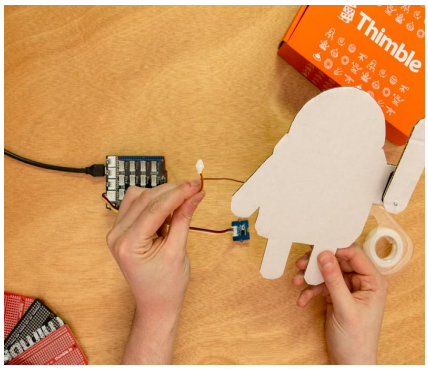

Be careful! It's small and pointy

Through the mounting hole and into the servo

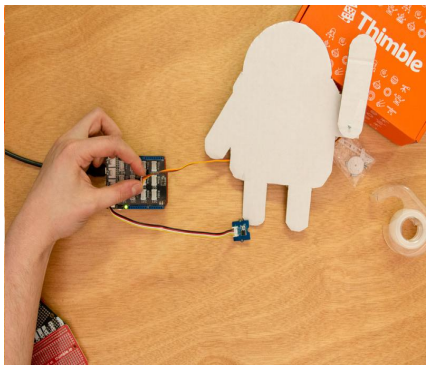


You're all done with the construction

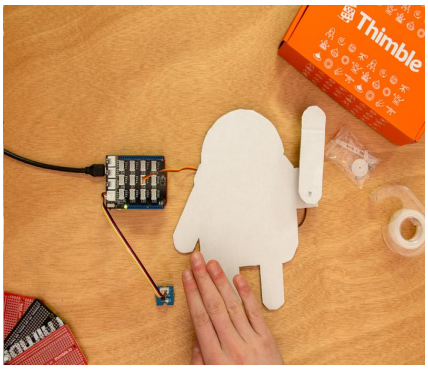
High five!



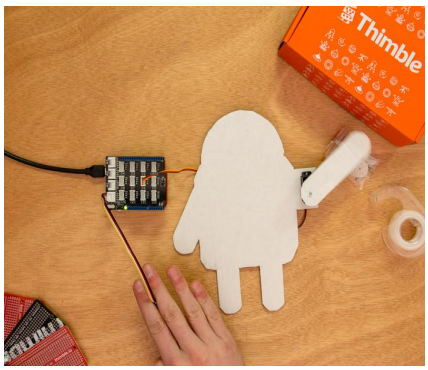
Take the servo cable...

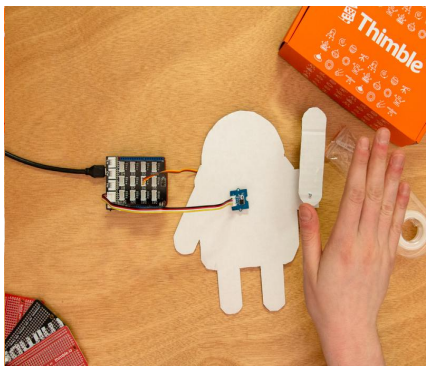


... place back into the same socket

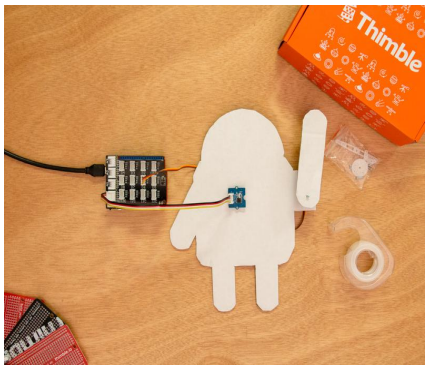


Test out his waving feature!





Moving the sensor might make the waving more natural



You're first blueprint is finished!

Taking it further

Feel free to use this code to explore what else you can do with the servo. Maybe your robot is afraid of light and covers his eyes, rather than waves? Maybe you want your robot to wave at you when triggered by sound? Maybe you want to completely change the look of your robot and not use our template? Maybe you don't want it to be a robot at all?

As you progress through the project blueprints, revisit this project. Try connecting different sensors as you learn how to use them in the other projects. Don't be afraid to experiment and share any cool modifications with the community. We love seeing and sharing what you make!

Coding Basics 101

Start

This section will go over the basics of computer programming.

Variables

In math, we use variables to solve equations. For example, when we say x = 2, we know that x is a variable and it equals the value '2'. Since we know that x = 2, we can say that x + x = 4. Variables are useful because they give insight to a problem. For example, let's say I want to calculate the volume of a sphere. We can do the following calculation:

$$\frac{4*3.1415*3^3}{3} = 113.1$$

If we showed just this equation to someone else, and asked them what this equation is for without any context, they might be confused. Let's replace the numbers with a more meaningful equation:

$$\frac{4\pi r^3}{3} = \text{volume of the sphere}$$ Where $$\pi = 3.1415$$

Now we know that if we take two constant numbers 4/3 and pi, and multiply it by a radius, we can get the volume of a sphere. Naming a variable can help a person understand what the problem is doing. Furthermore, since you know that the equation calls for a radius, you can calculate several volumes of different spheres if given different radii.

This is just one example of a variable in a real life application. Another example is a bank account - consider a single bank account number. This particular bank account stores money and can vary its monetary value by withdrawing or depositing into the account.

Types

There are also variables in code. We can make different types of variables; we will focus on the most basic ones:

`int`, any integer, such as 1, -5, or 391

`char`, any character, such as a, f, k

`string`, any group of character, such as hello, thisIsAlsoAString

When you want to declare a variable, simply state its type and the variable name:

```
type nameOfVariable;
```

Notice how the name of the variable does not include any spaces, and has a semicolon at the end. These are simple "grammar" rules in coding that we must follow so that the computer can understand what we want it to do. If we want