

Lenovo Educational Board Beginner Manual

Lenovo

Contents

Introduction to the Lenovo Educational Board	5
Components Schematic	7
Lesson B – 0: Getting Started.....	8
MATERIALS.....	8
Setting Up Your Lenovo Educational Board	9
Connecting To Your PC.....	15
Result	16
Lesson B – 1: Make It Blink.....	17
OVERVIEW.....	17
Teacher Guide.....	18
Background Info.....	20
Procedure.....	20
Result	26
Lesson B – 2: Neopixel LED	27
OVERVIEW.....	27
Teacher Guide.....	28
Background Info.....	30
Procedure.....	31
Result	36
Lesson B – 3: Press the Button!	37
OVERVIEW.....	37
Teacher Guide.....	38
Background Info.....	40
Procedure.....	40
RESULT	42
Lesson B – 4: Make it Buzz!.....	43
OVERVIEW.....	43
Teacher Guide.....	44
Background Info.....	45
Procedure.....	45
Result	47
Lesson B – 5: What’s the Temp?	48
OVERVIEW.....	48

Teacher Guide.....	49
Background Info.....	51
Procedure.....	51
Result	54
Lesson B – 6: EEPROM.....	55
OVERVIEW.....	55
Teacher Guide.....	56
Background Info.....	57
Procedure.....	57
Result	59
Lesson B – 7: Color Sensor	60
OVERVIEW.....	60
Teacher Guide.....	61
Background Info.....	62
Procedure.....	62
Result	65
Lesson B – 8: What Was That Sound?	66
OVERVIEW.....	66
Teacher Guide.....	67
Background Info.....	68
Procedure.....	69
Result	71
Lesson B – 9: Which Way Is Up?	72
OVERVIEW.....	72
Teacher Guide.....	73
Background Info.....	74
Procedure.....	74
Result	77
Lesson B – 10: Detect Brightness Levels	78
OVERVIEW.....	78
Teacher Guide.....	79
Background Info.....	80
Procedure.....	81
Result	82
Lesson B – 11: WiFi Scan	83
OVERVIEW.....	83

Teacher Guide.....	84
Background Info.....	85
Procedure.....	86
Result	88
Serial Monitor Information	89
How to open the Serial Monitor.....	89
How input text in the Serial Monitor	89
How to change the baud rate	90
Glossary of Terms	91
Software Terminology.....	91
Advanced Terminology	92
Hardware Terminology	93

Introduction to the Lenovo Educational Board

Welcome to the wonderful and exciting world of microcontroller programming! If this is your first time programming or programming microcontrollers, don't worry, we'll walk you through it. In this section we are going to introduce some basic concepts about microcontrollers, development boards, and how to get started in programming these small, miniature computers that can perform fun and exciting activities based on the simple instructions you will write yourself!

What exactly is the Lenovo Educational Board?

The Lenovo Educational Board is a specially designed development board made by Lenovo. It contains a microcontroller as well as a set of ready-to-use devices and sensors already connected for you. Unlike most development boards, you don't have to go find other parts to connect to it to make it do some fun and/or challenging tasks. Some of the devices included on the board are buttons, LEDs, Neopixel LEDs, a light sensor, a buzzer, a microphone, a temperature sensor, and many more. You will learn more about each of these devices that are included on the board and how they work, as well as how to program them in this course!

The Lenovo Educational Board is an extension to the Arduino board concept. The board is an open-source development board intended to provide an easy-to-use hardware and software solution for microcontroller-based projects. The Lenovo Educational Board takes this one step further and provides additional devices that the microcontroller will interface with.

Just like other Arduino development boards, you can tell your board what to do by sending a set of instructions to the microcontroller on the board by programming it with the Arduino programming environment (IDE). This board can be used in many fun electronic projects, such as a temperature sensor that changes color of a Neopixel light, a sound board, or a Morse code encoder/decoder. The possibilities are endless!

Why use a Lenovo Educational Board?

The Lenovo Educational Board is a "development board." A development board allows you to develop code and experiment with connecting additional hardware that allows you to create useful projects. There are many variations of development boards available. The following are the benefits of developing on the Lenovo Educational Board.

Affordability — The Lenovo Educational Board is relatively inexpensive compared to other development boards. With other development boards, you will need to purchase additional sensors (LEDs, buttons, etc.) in

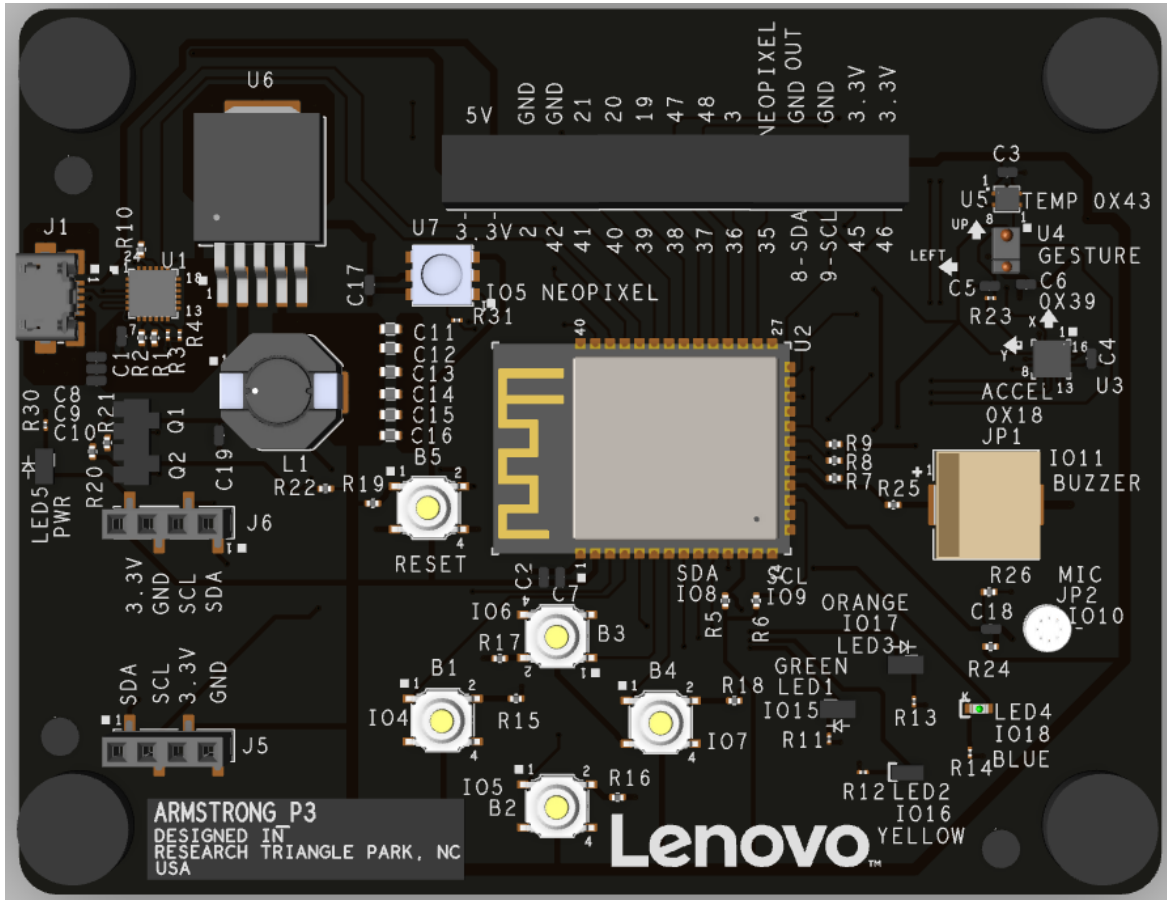
addition to the development board itself. Since the board is already equipped with many sensors and devices you will be able to create dozens of projects without having to purchase additional hardware.

Support and Compatibility – There is a wealth of support available for Arduino based development boards, such as tutorials and premade sketches for different projects. Getting started with the HPE Educational board is made very easy because there is a wealth of sample code, forums, guides, and tutorials, all online!

Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

Open source and extensible hardware - The schematics for the Arduino boards are published under a Creative Commons license. This allows experienced circuit designers to take a development board and expand upon it to make their own boards. Even relatively inexperienced users can use this knowledge to experiment and experiment and hack together their own circuits to interface with many different styles of boards.

Components Schematic



Reference ID	Component	Function
B1	Button 1	Input button
B2	Button 2	Input button
B3	Button 3	Input button
B4	Button 4	Input button
B5	Button 5	Reset
IO15	LED 1	Green LED Output
IO16	LED 2	Yellow LED Output
IO17	LED 3	Orange LED Output
IO18	LED 4	Blue LED Output
IO10	Microphone	Audio Input/Output
IO11	Buzzer	Audio Output
IO1	Neopixel	LED Output
J1	Port 1	USB port
L1		Inductor
U1		
U2	CPU	
U3	Accelerometer	
U4	Gesture Sensor	
U5	Temperature Sensor	
U6		
U7	Neopixel	

Lesson B – 0: Getting Started

MATERIALS

You will need the following items:

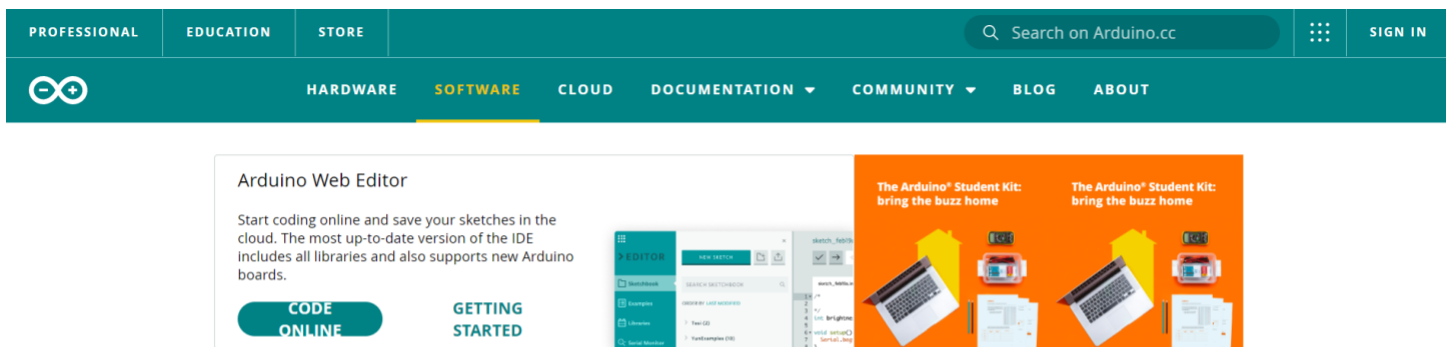
- PC (Windows)
- Lenovo Educational Board
- USB to micro-USB cable
- Download the free Arduino software

Setting Up Your Lenovo Educational Board

1. Download and open the free Arduino software (IDE).

Click [here](#) and refer to the **Getting Started** page for Installation instructions. We recommend downloading the **desktop IDE** version so you can have fun programming from anywhere without relying on an Internet connection. Please choose the Windows version below.

Here is the URL: <https://www.arduino.cc/en/software>

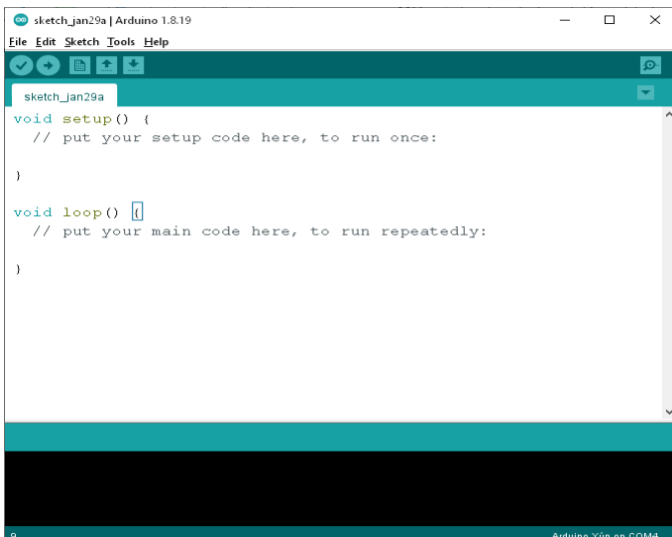


Downloads

The image shows the download page for Arduino IDE 1.8.19. On the left, there is the Arduino logo and the text 'Arduino IDE 1.8.19'. Below this is a description: 'The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.' On the right, there is a 'DOWNLOAD OPTIONS' section with a table of options. The 'Windows ZIP file' option is highlighted with a green box. A 'Help' button is visible on the far right.

DOWNLOAD OPTIONS	
Windows	Win 7 and newer
Windows	ZIP file
Windows app	Win 8.1 or 10 Get
Linux	32 bits
Linux	64 bits

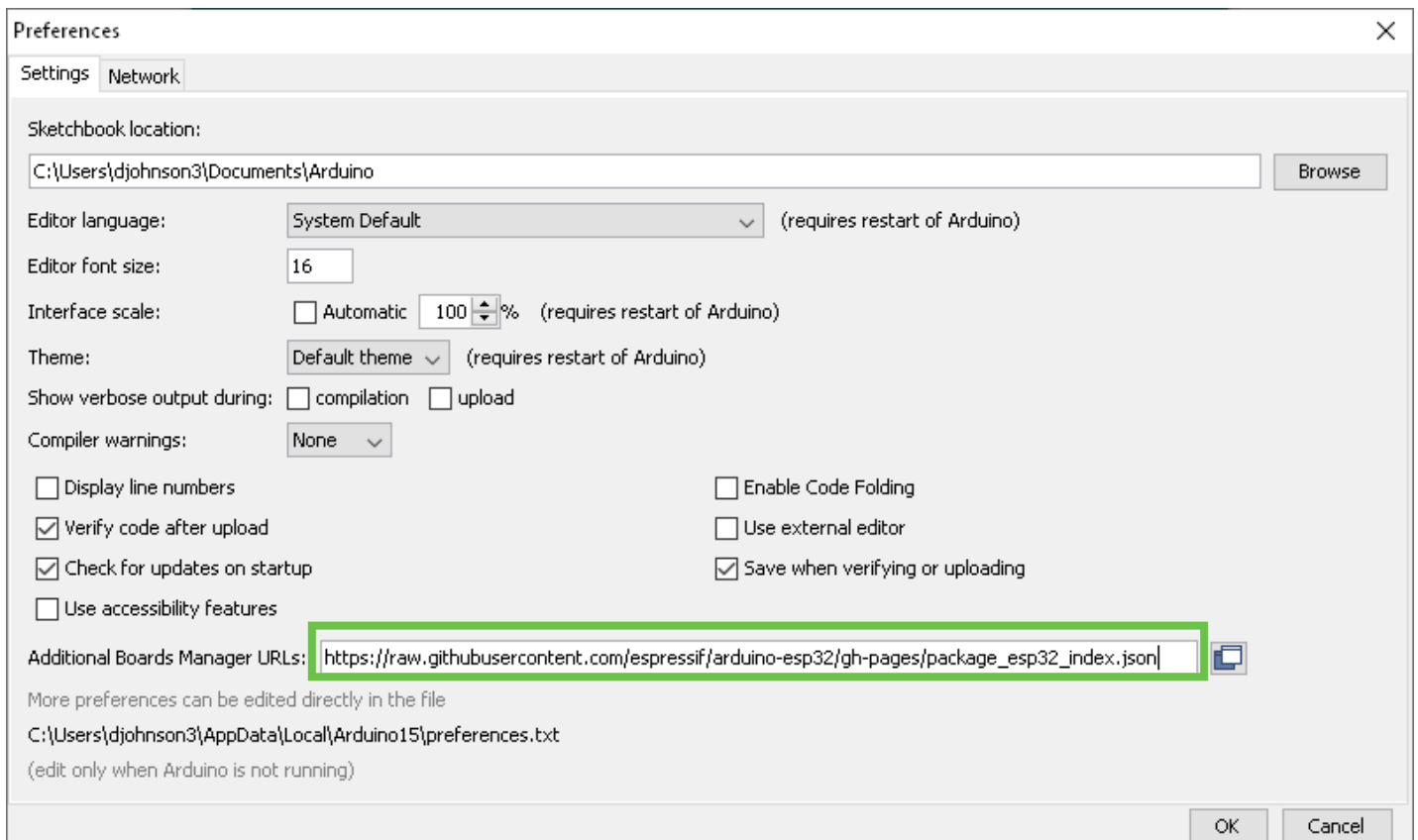
Once you have downloaded the Arduino program to your PC, navigate to and launch the Arduino IDE. The resulting window should look something like this:



2. Choose File > Preferences.

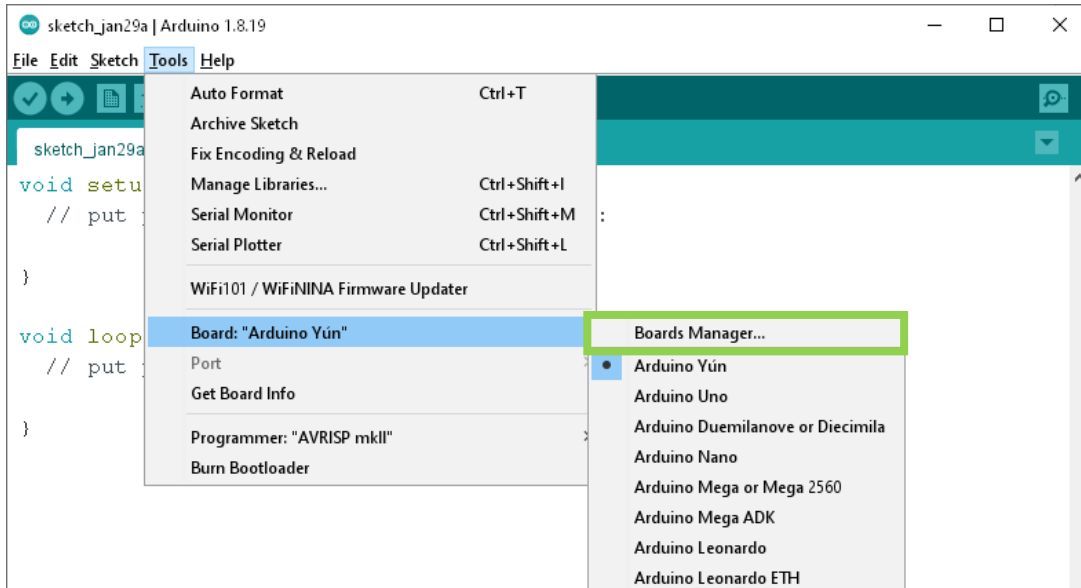
3. In the “Additional Boards Manager URLs:” text field, copy and paste the following URL:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



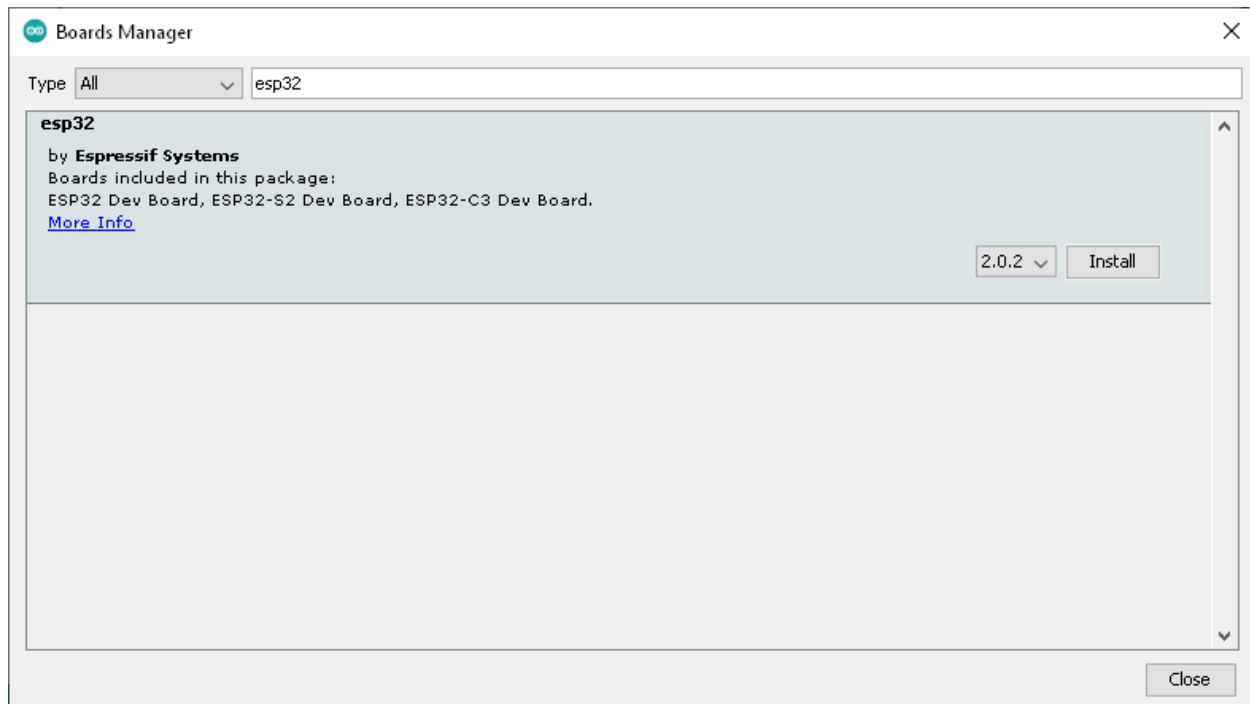
4. Click OK.

5. Choose Tools > Board > Board Manager...



6. Type **ESP32** in the text field located at the top of the resulting window.

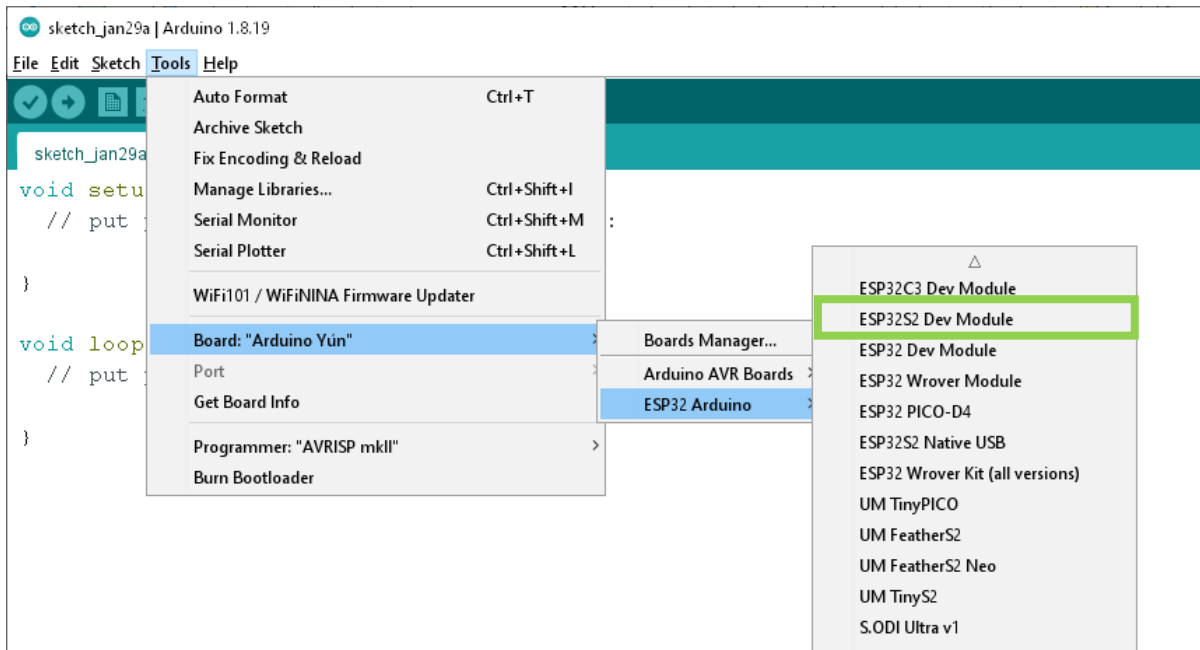
The following option should appear below:



7. Click Install and Close.

8. Choose Tools > Board: > ESP32 Arduino > ESP32S3 Dev Module.

The Arduino IDE will now recognize the Lenovo Educational Board once it is connected.



9. Click [here](#) to download the CP210x driver.

Here is the URL: <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>

10. Click the Downloads tab then click the CP210X Windows Driver to download the driver.

OVERVIEW

DOWNLOADS

TECH DOCS

COMMUNITY & SUPPORT

Download and Install VCP Drivers

Downloads for Windows, Macintosh, Linux and Android below.

*Note: The Linux 3.x.x and 4.x.x version of the driver is maintained in the current Linux 3.x.x and 4.x.x tree at www.kernel.org.

Software Downloads

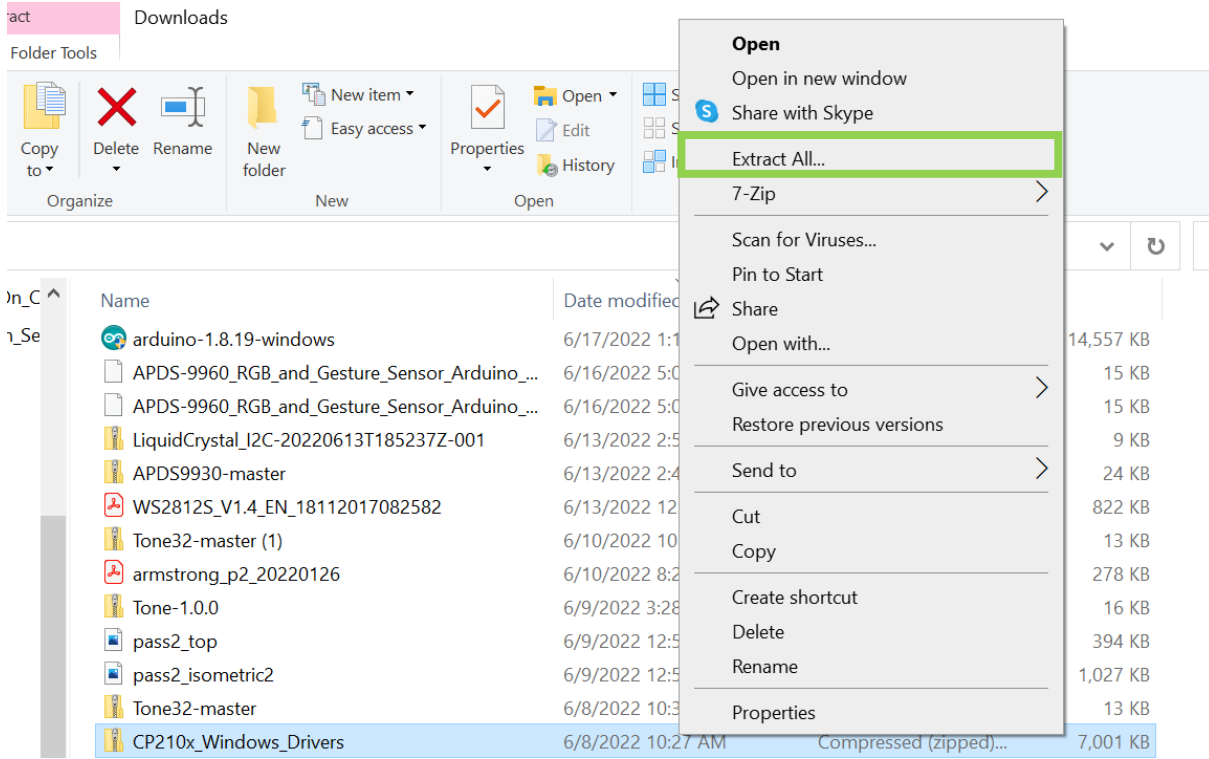
Software (11)

Software · 11

CP210x Universal Windows Driver	v11.1.0 3/22/2022
CP210x VCP Mac OSX Driver	v6.0.2 10/26/2021
CP210x VCP Windows	v6.7 9/3/2020
CP210x Windows Drivers	v6.7.6 9/3/2020
CP210x Windows Drivers with Serial Enumerator	v6.7.6 9/3/2020

11. Navigate to your Downloads folder and right click on the Zip folder, then choose Extract

All...



12. Navigate the now unzipped folder and open it.

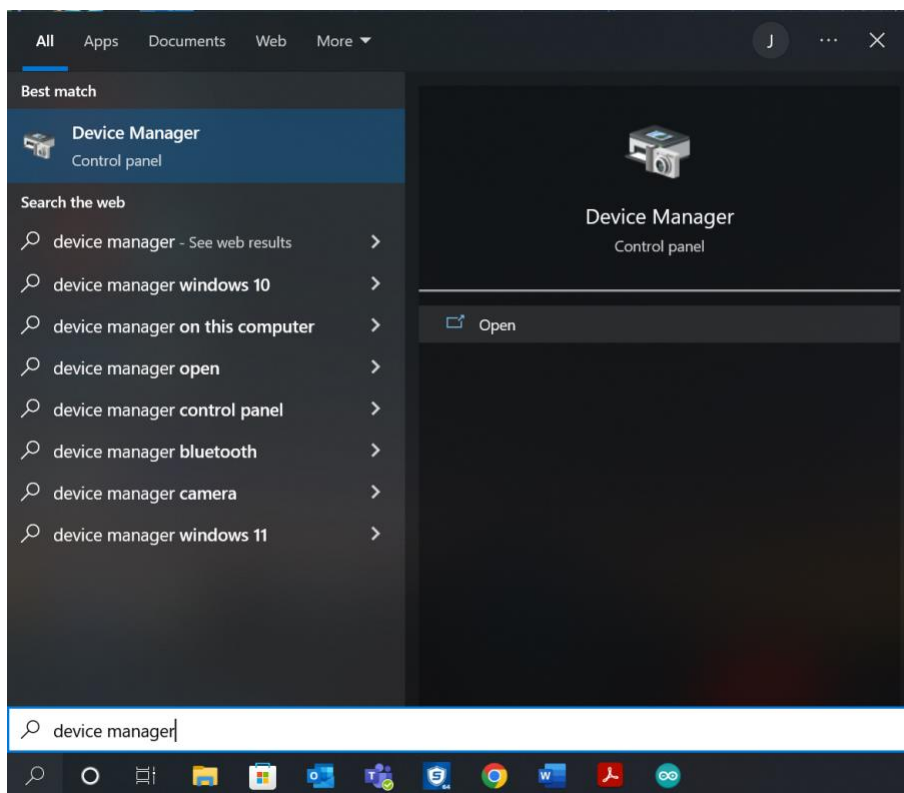
13. Open the CP210xVCPInstaller_x64 file and agree to the terms to finish the installation.

Name	Date modified	Type	Size
x64	6/8/2022 10:29 AM	File folder	
x86	6/8/2022 10:29 AM	File folder	
CP210xVCPInstaller_x64	6/8/2022 10:29 AM	Application	1,026 KB
CP210xVCPInstaller_x86	6/8/2022 10:29 AM	Application	903 KB
dpinst	6/8/2022 10:29 AM	XML Document	12 KB
SLAB_license_Agreement_VCP_Windows	6/8/2022 10:29 AM	Text Document	9 KB
slabvcp	6/8/2022 10:29 AM	Security Catalog	11 KB
slabvcp	6/8/2022 10:29 AM	Setup Information	8 KB
v6-7-6-driver-release-notes	6/8/2022 10:29 AM	Text Document	16 KB

Connecting To Your PC

Now that the Arduino IDE can recognize your Lenovo Educational Board, we can begin connecting the board to your PC.

1. Insert your USB to micro-USB cable to your PC and the Lenovo Educational Board.
2. Search for Device Manager on your computer and click Open.



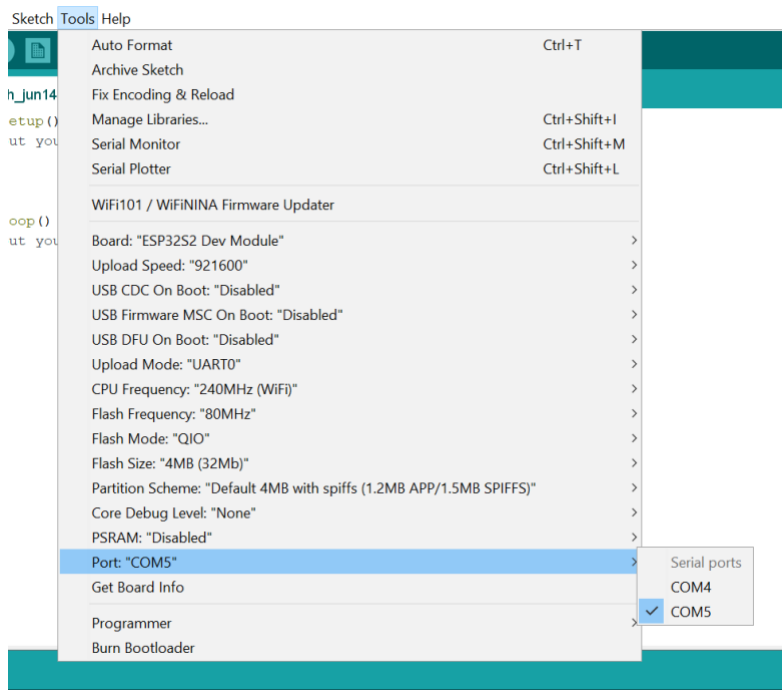
3. Scroll down to Ports (COM & LPT) near the bottom and expand the drop down.



4. Find the COM Port labeled CP210x and refer to the Port number at the end of the line (this one is COM5).



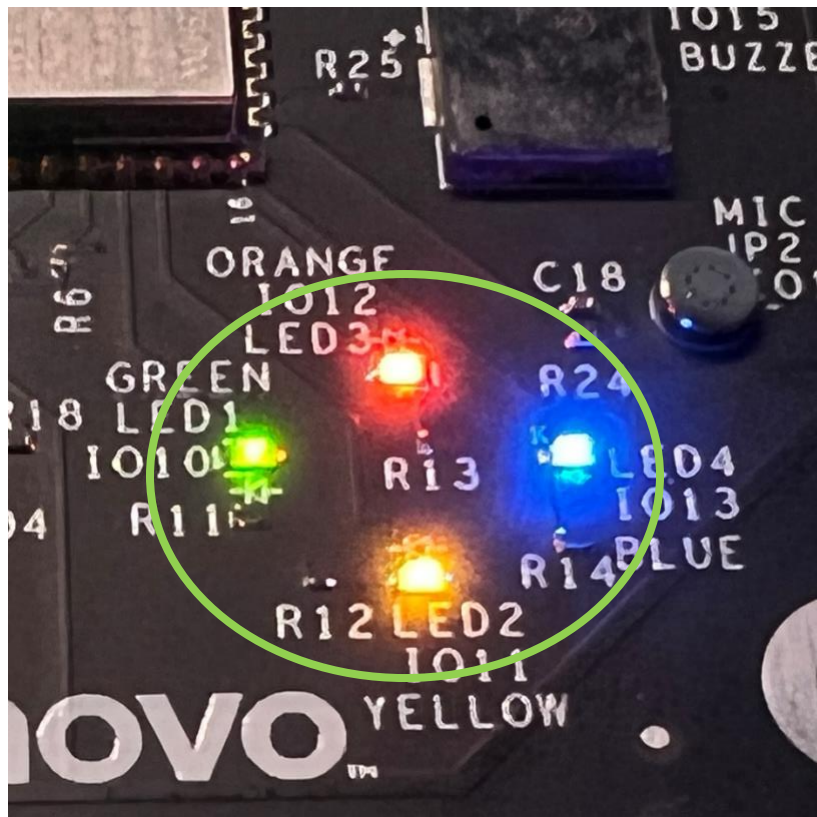
5. Go back to Arduino and select Tools > Port: > COM5 (select the port number that you found in the device manager).



Result

Congratulations! You have connected your Lenovo Education Board to your PC. The Arduino IDE has found your board and you can now have fun programming your board! Please proceed to the next lesson.

Lesson B – 1: Make It Blink



OVERVIEW

In this lesson we will:

- Learn about LEDs
- Learn about the language used to code in Arduino IDE
- Write a program to activate/deactivate one LED repeatedly

Teacher Guide

The Arduino IDE is an important element of this course. Getting familiar with all its tools and features can be very helpful when building or troubleshooting a project. This guide will help you discover all the necessary components in Arduino IDE that will prepare you for the course.

Looking at each of the tabs from the top left corner of the Arduino IDE, we have the **File**, **Edit**, **Sketch**, **Tools**, and **Help** tabs.

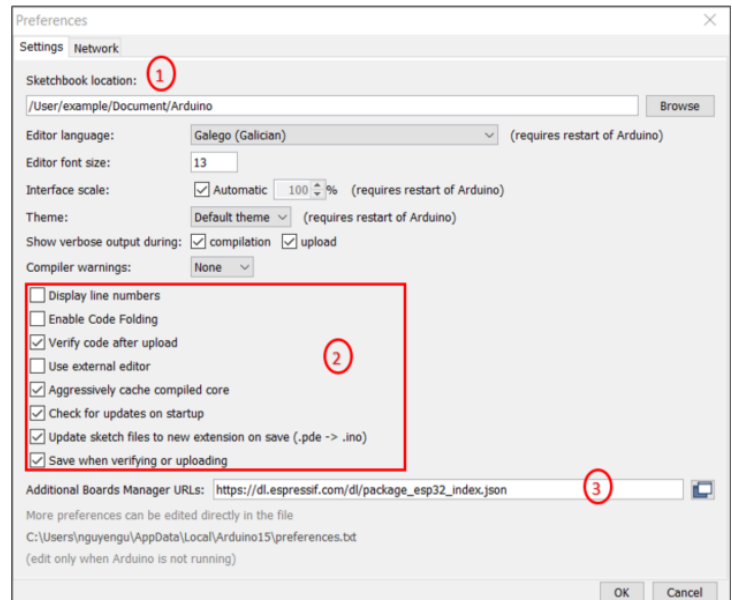
First, let's examine the **File** tab. This is where you can open, create, save, and close your sketches.

If you select on the **Example** option, you can see all the pre-made sketches for you to test on your board directly. We highly recommend you to look through some of these examples and try them on your board! Keep in mind that many of these sketches are made for different versions of the Arduino board, so some will not work on the Edu board. More examples will be added when you install additional libraries later.

File	Edit	Sketch	Tools	Help
New			Ctrl+N	
Open...			Ctrl+O	
Open Recent				>
Sketchbook				>
Examples				>
Close			Ctrl+W	
Save			Ctrl+S	
Save As...			Ctrl+Shift+S	
Page Setup			Ctrl+Shift+P	
Print			Ctrl+P	
Preferences			Ctrl+Comma	
Quit			Ctrl+Q	

You can set your preferences for the IDE here as well. Click on the **Preference** option or press Ctrl + Comma buttons to open up the preferences dialog.

- The **Sketchbook location** is where all of your libraries will be located. You can change this to any directory that is most convenient for you.
- In the middle of the Preferences window, you will find a list of checkbox items. We recommend you to leave this as default.
- In our set up guide from Lesson 0, we asked you to insert a URL for the Additional Board Manager URLs field. This URL points to an index file that the Board Manager uses to build the list of available and installed boards. You can add multiple URLs by separating them with commas.



Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

What is an LED?!

LED stands for **light emitting diode**. A **diode** is an electrical device that allows current to flow in one direction. When the current travels through the diode, the semiconductor material of the diode lights up. This is why LEDs are called **solid-state devices**; they differ from other forms of lighting such as incandescent lights (a normal lightbulb!) which uses **heated filaments** (where a wire is heated to a temperature that produces light).

If you look at the diagram, an **anode** is a terminal where electricity flows into, while a **cathode** is a terminal where electricity flows out of.

Procedure

How do we code in the Arduino IDE?

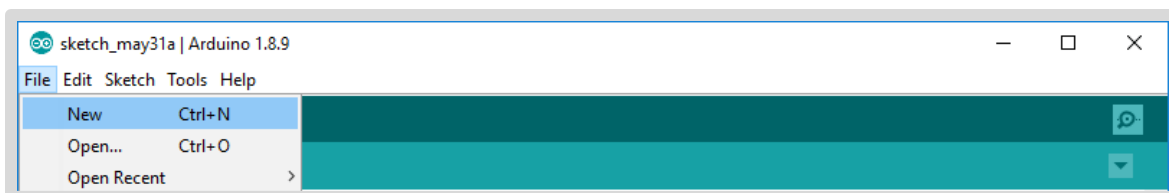
Arduino's "language" is mostly based off C. C is a **procedural programming language**, which means it does not support classes or objects. However, it provides the functionality we need with functions and variables to program our hardware.

In this tutorial, we will write a sketch from the Arduino IDE to light up an LED on the Arduino board.

First, make sure you've gone through all of Lesson B – 0 in how to set up your Dos board and the Arduino IDE before proceeding.

Now it's time to code! Let's create a blank file in your Arduino IDE.

1. Choose File > New



You should see the following starting code in your file:

```
sketch_may31a
void setup() {
  // put your setup code here, to run once:
}

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

NOTE: In the C++/C programming language, the “//” represents a start of a **comment**. Anything behind the comment symbol will not be executed.

These two **void setup()** and **void loop()** functions are going to be used in most, if not all, of our programs. **Functions** are pieces of code that take in an explicit input and give an output (usually). In this case, both **setup()** and **loop()** return **void**, or nothing.

Comment – a statement that explains or annotates the source code of a program. Compilers generally skip through the comments and not run them.

Defining Your LEDs

Before we dive into the **setup()** and **loop()** methods, we need to define our LEDs as variables. In the next steps, we will define each LED color with a value from the [Lenovo Educational Board components schematic](#). Having incorrect values for each variable can result in unexpected output and require additional debugging of the code.

1. Move your mouse before the function “**void setup()**” and click to place your cursor. Press Enter to create a new line above.
2. Using the example shown as your guide, input the following text to declare constants for your LEDs in the new line:

```
// constants will not change
const int GreenledPin = 15; // green LED pin
const int YellowledPin = 16; // yellow LED pin
const int OrangeledPin = 17; // orange LED pin
const int BlueledPin = 18; // blue LED pin
```

In this example, we have given each LED their own unique name and assigned it to the IO numeric value specified on the board's schematic.

The keyword `const` is used to declare **constants**. Constants do not change. The keyword `int` is short for **integer** and is a variable type used to define numeric variables.

3. Press Enter on your keyboard to begin a new section.

4. Input the following text to define your LEDs as a variable:

```
// variables will change
int ledState = LOW; // ledState used to set the LED
```

For this example, declaring the LED state as LOW means the LED will be set to the off position (LOW = off and HIGH = on).

Setup and Loop Functions

Setup Function

The `setup()` function is where the program will execute only once at the beginning of each power up or reset of the board. In this method, you would want to initialize variables, pin modes, and/or start using libraries.

1. Set the digital pins as OUTPUT as seen in the example below.

The method `pinMode()` is used in our code to prepare the LEDs to OUTPUT. You can also use `pinMode()` to set INPUT, but we do not need that in this lesson.

```
void setup() {
  // set the digital pins as output
  pinMode(GreenledPin, OUTPUT);
  pinMode(YellowledPin, OUTPUT);
  pinMode(OrangedPin, OUTPUT);
  pinMode(BlueledPin, OUTPUT);
}
```

Loop Function

The `loop()` function is called after the `setup()` function. This is where the code in your program will run continuously. The program starts directly after the opening curly bracket (`{`), runs the code until it reaches the closing curly bracket (`}`), and returns to the beginning of the loop to repeat the function.

The `loop()` function will continue to run until you press the board's RESET button, power off your board, or upload a new sketch.

For this example, we will set up an **if-else statement** to make our LEDs blink. We will learn more about if-else statements in future lessons.

1. Input the if-else statement within the `void loop()` curly brackets as seen in the example below.

Earlier, we set our `ledState` to `LOW` in our `void setup()` function. This allows our **condition** (`ledState == LOW`) to result as true. Since our condition results true, the next line (`ledState = HIGH`) is executed, turning the LED output to `HIGH`.

Now that `ledState = HIGH`, our condition becomes false and passes along to the `else` statement. This will allow our LEDs to blink due to the resulting true/false outcomes.

```
void loop() {  
  // if the LED is off turn it on and vice-versa  
  if (ledState == LOW){  
    ledState = HIGH;  
  } else {  
    ledState = LOW;  
  }  
}
```

2. Press Enter on your keyboard to start a new section.

3. Set the LEDs with the `ledState` of the variable using the `digitalWrite()` method.

```
void loop() {
  // if the LED is off turn it on and vice-versa
  if (ledState == LOW){
    ledState = HIGH;
  } else {
    ledState = LOW;
  }
  // write the value to the LED
  digitalWrite(GreenledPin, ledState);
  digitalWrite(YellowledPin, ledState);
  digitalWrite(OrangedPin, ledState);
  digitalWrite(BlueledPin, ledState);
}
```

The method `digitalWrite` will turn on each LED to LOW or HIGH depending on the current status of our `ledState`.

4. Press Enter to create a new line.

5. Add a `delay()` function at the end of the `void loop()` function before the last curly bracket and enter **1000** as it's parameter.

The `delay()` function pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Creating this delay at the end of the loop will cause the LEDs to blink on and off (HIGH and LOW) for the duration of second each time.

```
digitalWrite(GreenledPin, ledState);
digitalWrite(YellowledPin, ledState);
digitalWrite(OrangedPin, ledState);
digitalWrite(BlueledPin, ledState);

// delay 1 second
delay(1000);
}
```

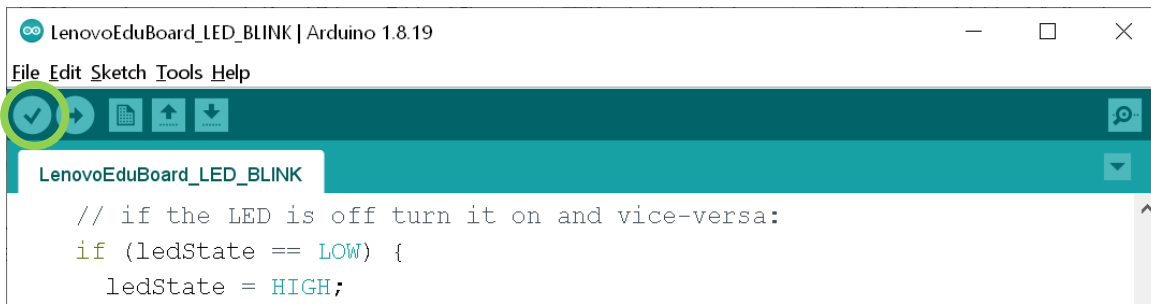
Compiling and Uploading

After you have finish coding the Blink program, now it's time to compile and upload the sketch on to the board!

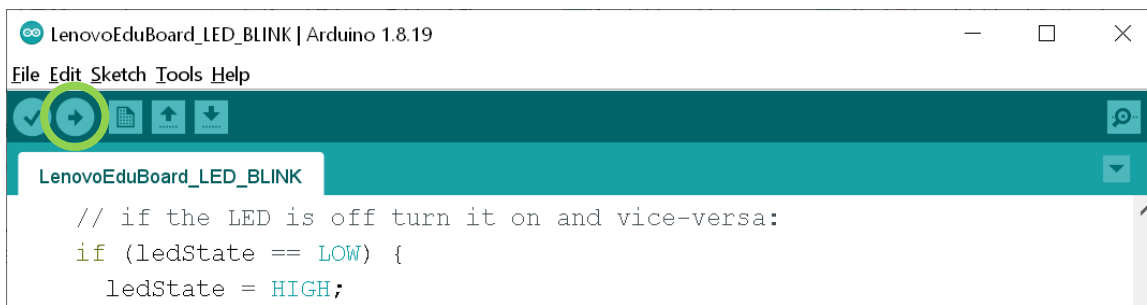
1. Make sure you have the correct COM port and Board selected before compiling and uploading.

2. Click the verify button located at the top left corner of the page to compile your sketch onto the board.

Wait until you get a message indicating that the program is done compiling.



3. Click on the upload button to upload your sketch onto the board.



Compiler – a program that processes the written code in a particular programming language and turns them into machine language that a computer’s processor uses.

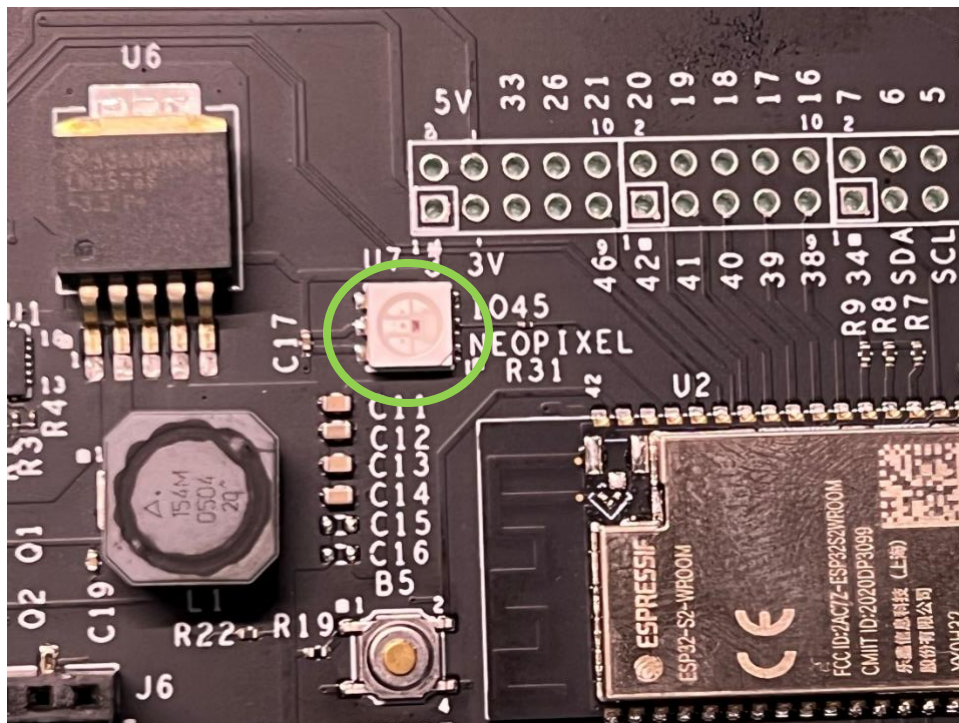
Result

You have just learned how to code an Arduino board! You now know how to control LEDs and have an introduction to how the Arduino language and IDE work.

Now It's Your Turn!

- Can you change how long it takes for the LEDs to blink?
- Can you try to make the LEDs blink in a pattern?

Lesson B – 2: Neopixel LED



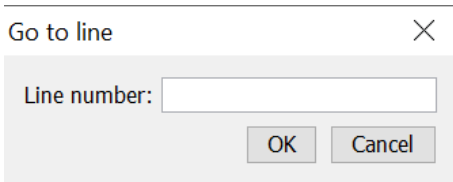
OVERVIEW

In this lesson we will:

- Learn about the Neopixel LED
- Learn how to interact with the Lenovo Educational Board through Serial monitor
- Write a simple program to change the color of the Neopixel LED

Teacher Guide

To continue exploring the Arduino IDE, we will examine the Edit tab. This is where you can edit the content of your sketches. Some common actions that you might be familiar with are undo, redo, find a certain word, cut, copy and paste.



The Go to line... option directs you to the line number that you specified. You can comment out any text that you don't want the program to execute, or uncomment text that you want the program to execute.

On the right side of the Edit drop-down menu is a list of hot-keys for each action. For example, instead of clicking on the Edit tab and then select the Go to line... option, you can just press the Ctrl and L keys on your keyboard.

Edit	
Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Copy for Forum	Ctrl+Shift+C
Copy as HTML	Ctrl+Alt+C
Paste	Ctrl+V
Select All	Ctrl+A
Go to line...	Ctrl+L
Comment/Uncomment	Ctrl+Slash
Increase Indent	Tab
Decrease Indent	Shift+Tab
Increase Font Size	Ctrl+Plus
Decrease Font Size	Ctrl+Minus
Find...	Ctrl+F
Find Next	Ctrl+G
Find Previous	Ctrl+Shift+G

In this lesson, we will be working with Neopixel LED lights, which changes colors depending on the Red, Green, and Blue values that we set. The values range from 0-255. To adjust the values, you would have to input the red, green, and blue values in the respective order in the serial monitor. For example, the values “128, 0, 128” will result in the color purple. If you want to know the RGB values for a specific color or shade, you can search on the internet for “RBG values” and it will give you the exact values for that color. For additional information, please visit <https://learn.adafruit.com/adafruit-neopixel-uberguide>. This page will provide a more in-depth explanation of the Adafruit Neopixel.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board properly connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

4. Compiling Error

Close out and restarted your IDE

Try to recompile and upload your program again.

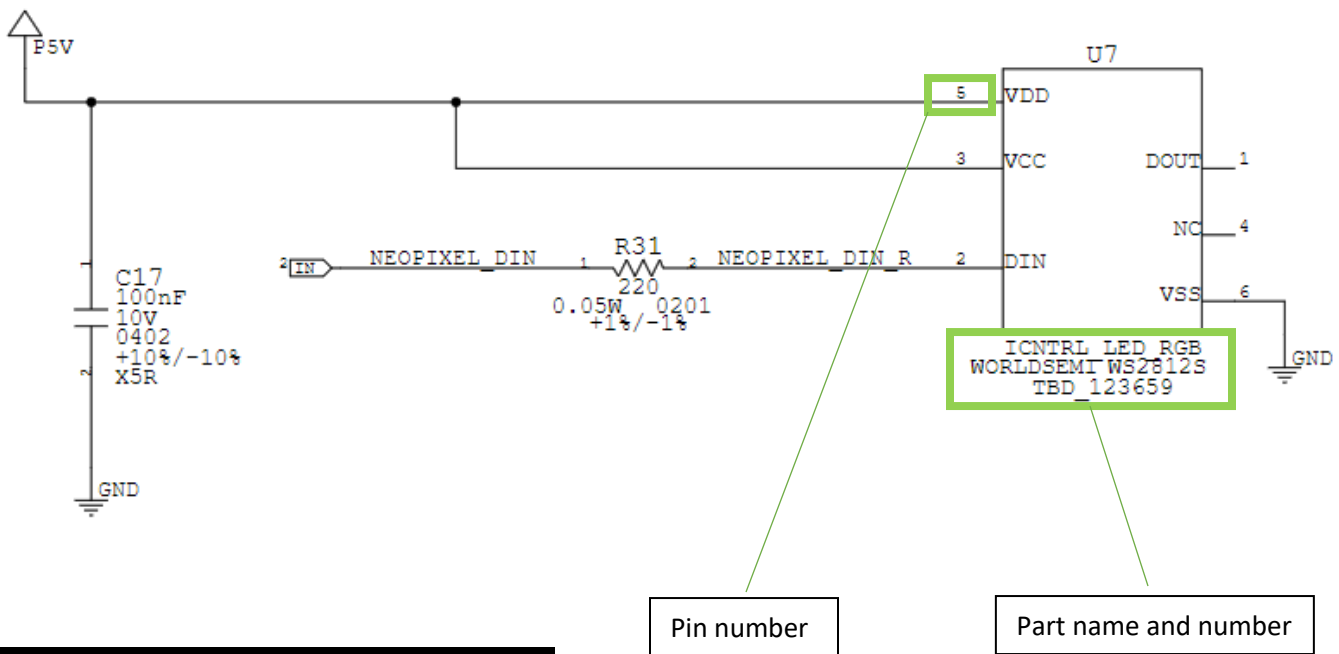
5. If error persists, identify what your error is and try searching for a solution online.

Background Info

What is a Neopixel LED? What's RGB?

A **Neopixel** is an RGB LED created by the electronic components company Adafruit. This means it has a red, green, and blue LED already set up inside it, so all you have to do is tell it what RGB values you want it to have.

The **RGB color model** is the default way for a computer to interpret and display colors. All colors from your computer screen (and the Neopixel) are created from a mix of red, green, and blue light! (If you're wondering why green is used instead of yellow, look up **Additive color vs Subtractive color**) With the Neopixel, you can get any color you want to display.



Schematic – a diagram produced by engineers to display the elements of a device in graphical terms rather than using real pictures (these are often used in electronics and circuitry)

Procedure

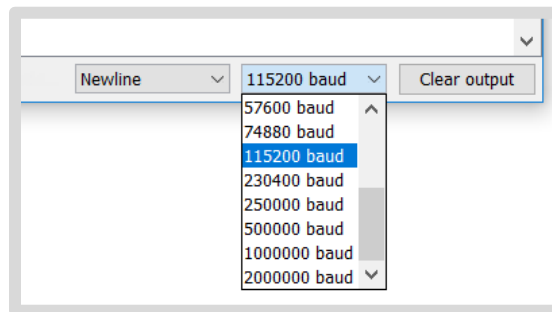
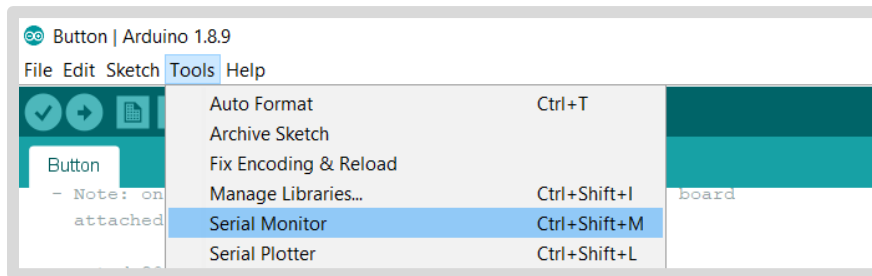
First, we will go over how you can see serial outputs of your board through a serial monitor.

1. Choose File > New.

Make sure you have a COM port selected.

2. Choose Tools > Serial Monitor.

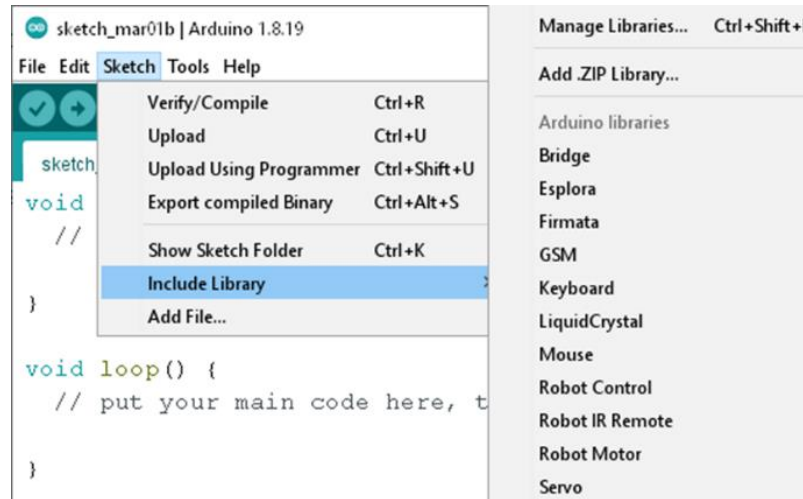
3. Change the speed to **115200** baud from the bottom right corner the serial monitor.



Importing Libraries

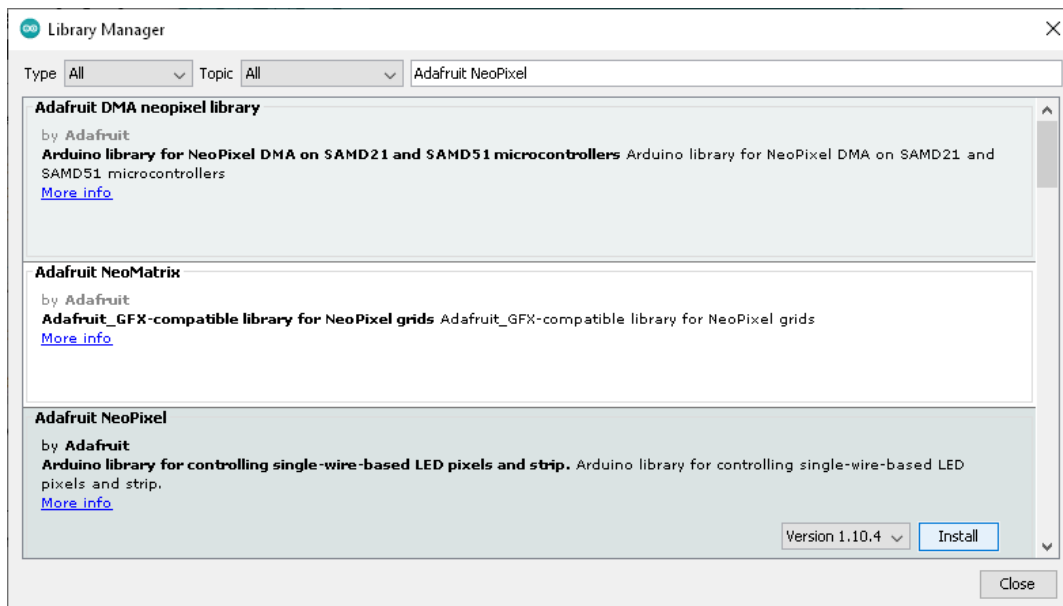
For this program, we will need to import the **Adafruit NeoPixel library**. For reference, a **library** is a collection of unchanging code/resources that you can use in your program. You can download various libraries online and use them in your program!

1. Install the Adafruit NeoPixel library by clicking Sketch > Include Library > Manage Libraries.



2. In the resulting dialog box, type **Adafruit NeoPixel** into the search bar.

3. Navigate to the Adafruit NeoPixel search result and click Install then Close.



4. Import the Adafruit NeoPixel library to your program by typing the following command at the beginning of your sketch above the `setup()` function:

```
#include <Adafruit_NeoPixel.h>
```


5. Next, you will need to define the following variables and instantiate a NeoPixel object:

REMINDER: In C++/C, the “//” represents a start of a comment. Anything behind the comment symbol will not be executed by the compiler.

The very last line (Adafruit_NeoPixel ...) is how we initialize objects in Arduino.

```
#define NEOPIXEL_PIN 1 // Neopixel is on pin 45
#define NEOPIXEL_COUNT 1 // There is only 1 RGB LED on DOS
#define NEOPIXEL_INDEX 0 // RGB LED is at index 0

#define MIN_COLOR_LIMIT 0 // Min R,G,B value
#define MAX_COLOR_LIMIT 255 // Max R,G,B value

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
// Create a NeoPixel object
Adafruit_NeoPixel Pixel(NEOPIXEL_COUNT, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
```

6. Input the following setup() function:

```
void setup() {
  // Set baud rate on the serial port
  Serial.begin(115200);
  Serial.println("HPE ESP32 Dos says Hello!");

  // Initialize the NeoPixel object
  Pixel.begin();

  // Set all pixel colors to OFF
  Pixel.clear();
  Pixel.show();

  // Show syntax for providing R,G,B values
  Serial.println();
  Serial.println();
  Serial.println("Type comma separated values [0...255]");
  Serial.println("for R, G and B in the text window above");
  Serial.println("and then press ENTER.");
  Serial.println("Example: 128,0,128<ENTER> to glow purple!");
  Serial.println();
}
```

7. Input the following `loop()` function:

NOTE: You should have the Serial Monitor up and running from the beginning of this procedure. When you compile your code at the end of the procedure, these messages will be displayed on the Serial Monitor. You will then be able to directly type in colors in the Serial Monitor according to the formatted example above.

```
void loop() {
  // Read values from the serial port
  while(Serial.available() > 0)
  {
    // separate integer values by comma
    int red = Serial.parseInt();
    int green = Serial.parseInt();
    int blue = Serial.parseInt();

    // A newline character indicates end of input
    if(Serial.read() == '\n')
    {
      // Adjust the values if out of range
      red = constrain(red, MIN_COLOR_LIMIT, MAX_COLOR_LIMIT);
      green = constrain(green, MIN_COLOR_LIMIT, MAX_COLOR_LIMIT);
      blue = constrain(blue, MIN_COLOR_LIMIT, MAX_COLOR_LIMIT);

      // Display the color values to the serial
      Serial.print("Setting the RGB LED with");
      Serial.print(" R=");
      Serial.print(red, DEC);
      Serial.print(" G=");
      Serial.print(green, DEC);
      Serial.print(" B=");
      Serial.println(blue, DEC);

      // setPixelColor() takes RGB values, from 0,0,0 up to 255,255,255
      Pixel.setPixelColor(NEOPIXEL_INDEX, red, green, blue);

      // Push the color data to the hardware
      Pixel.show();
    }
  }
}
```

All of the code in our `loop()` method is also in a `while()` loop. A while loop only runs when the condition in its parentheses is true but will run forever until that condition becomes false.

In this case, the while loop can only run if we have input available to read in from the Serial Monitor. This is exactly the purpose of the `Serial.available()` method; it checks if there are any characters to read in and returns the number of them. If the number is greater than 0, that means there is input to read in, and the code in the while loop will run!

Parsing is how a computer reads through string/text input and turns it into usable data. In this code, we will use the method `Serial.parseInt()` to read in the next available **integer** (whole number) from our input numbers.

```
int red = Serial.parseInt();
int green = Serial.parseInt();
int blue = Serial.parseInt();
```

The variables red, green, and blue will each store a separate number from your input in the Serial Monitor because they are separated by commas, so the commas serve as barriers between the numbers.

Many times, the end of an input line has an invisible `'\n'` or **newline character** associated with it. This is the character your computer reads and stores when you press 'enter' while typing something. Our code will check for this newline character which indicates we have finished inputting our numbers. We can't display RGB values outside of the range [0, 255], so the `constrain()` method makes sure if the number is below 0 it is set as 0 and if it is above 255 it is set to 255.

```
// A newline character indicates end of input
if(Serial.read() == '\n')
```

We print to the Serial Monitor the integers we are using to set the RGB values of the Neopixel. The **DEC** in `Serial.println()` stands for **decimal**, or **base 10**, which is the normal number system we use (if you think of place values, each of our place value is 10 to the power of something, starting from 0 as an exponent).

```
Serial.print("Setting the RGB LED with");
Serial.print(" R=");
Serial.print(red, DEC);
Serial.print(" G=");
Serial.print(green, DEC);
Serial.print(" B=");
Serial.println(blue, DEC);
```

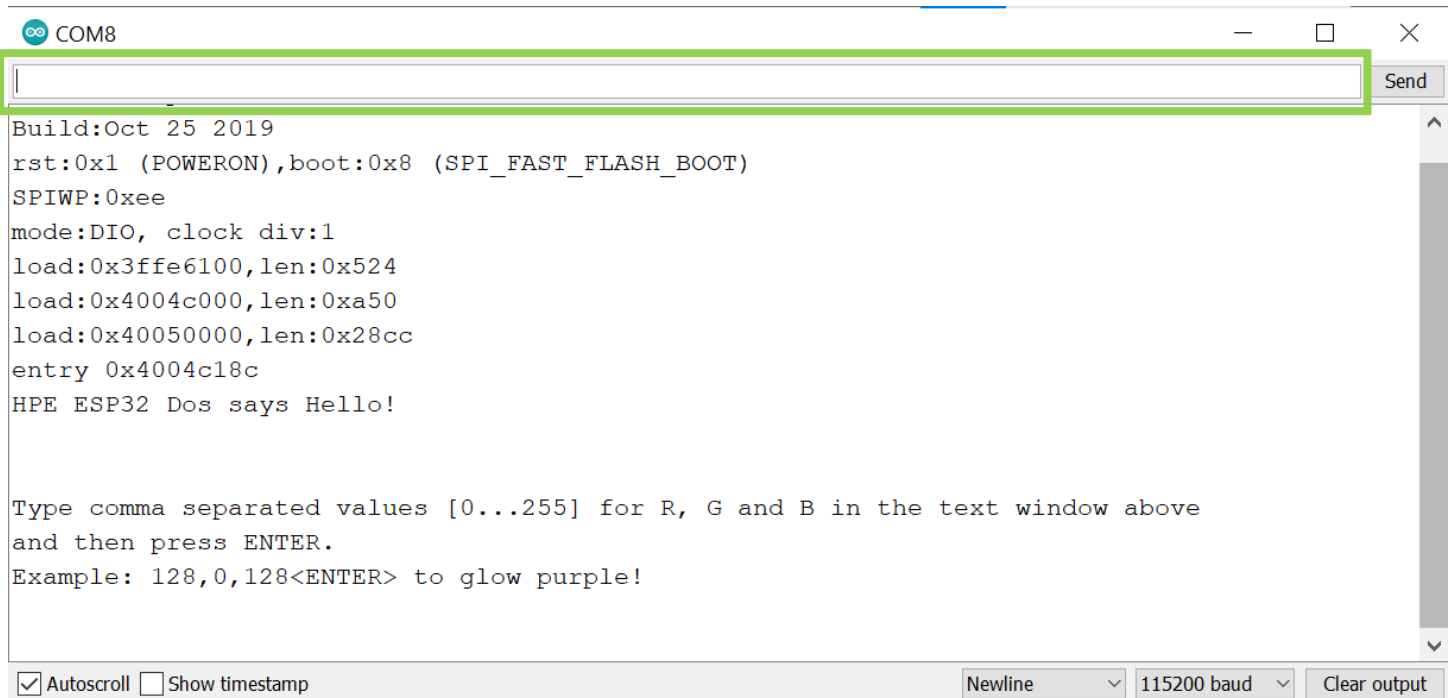
If you are familiar with the concept of **binary** or **base conversion**, `Serial.print()` also supports printing to common bases other than DEC. If you are not, don't worry! We won't be using it in any lessons, but if you are curious to learn more feel free to search online for additional learning resources. Technically, we do not need to write DEC to print the numbers correctly to the Serial Monitor so don't panic if you don't have it in your code/don't understand it.

8. Compile, Upload, and Verify.

After that we finally set the Pixel color to our variables and update the Neopixel.

9. Make sure to open the Serial Monitor to see results.

Open the Serial Monitor as seen at the beginning of this lesson and then type RGB values in the textbox in the format "0,255,0"



```
COM8
Build:Oct 25 2019
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3ffe6100,len:0x524
load:0x4004c000,len:0xa50
load:0x40050000,len:0x28cc
entry 0x4004c18c
HPE ESP32 Dos says Hello!

Type comma separated values [0...255] for R, G and B in the text window above
and then press ENTER.
Example: 128,0,128<ENTER> to glow purple!
```

Note: For Rev 2 of the Lenovo Educational Board, your Neopixel may not light up as expected due to an expected issue

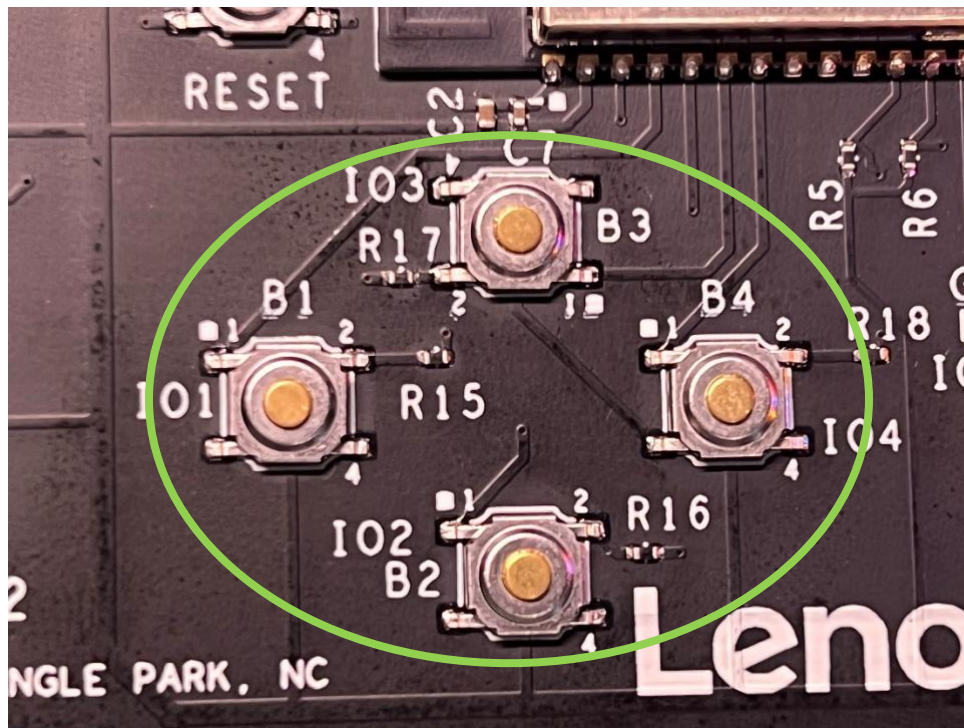
Result

After you compile and upload the sketch, should be able to enter the RGB values from the serial monitor to change the color of the Neopixel! The RGB values are between 0-255, so your input can be something like **220, 30, 125**. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you try to make the color purple out of the Neopixel LED?

Lesson B – 3: Press the Button!



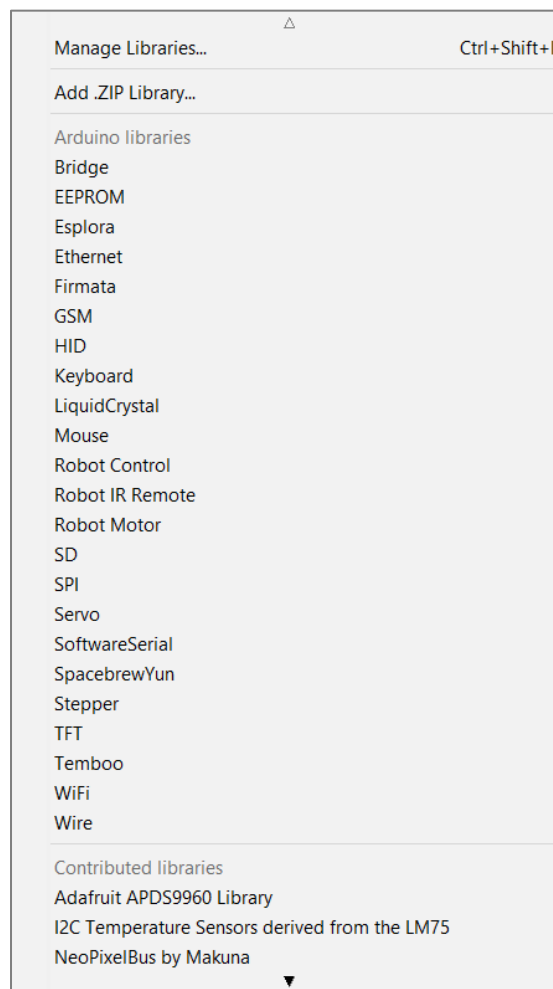
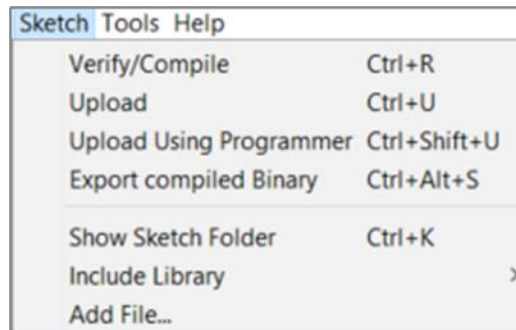
OVERVIEW

In this lesson we will:

- Learn about Push-Buttons
- Write a simple program to utilize the function of a Push-Button

Teacher Guide

Here we will look into the Sketch tab of the IDE Menu. The Sketch tab allows you to Verify/Compile and Upload your sketches. You can import additional libraries here as well as access libraries that are preinstalled.



Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

You've probably seen and used buttons all your life. They come in a wide variety of shapes and sizes, but their function is usually similar: to activate something. A push button achieves this by completing an electric circuit when pressed, allowing electric current to pass and performing an action.

There are two types of activation: momentary and non-momentary. **Momentary** buttons (ex: calculator, buzzer) work as long as you are actively pressing them, while **non-momentary** buttons (ex: TV power button) only take one press to turn them on/off. On our board, we can program our buttons to be either type! In this lesson, we will be coding a momentary button.

Procedure

1. In a blank sketch, set two constant integers to the pin number of the IO devices. See code below:

```
// constants used here to set pin numbers
const int buttonPin = 4; // the number of the pushbutton pin
const int ledPin = 18; // the number of the Blue LED pin
```

2. Then create a variable to read the pushbutton status to see if it is pressed or not.

```
// variables will change
int buttonState = 0; //variable for reading the pushbutton status
```

3. Input the following **setup()** and **loop()** functions.

Just like we did in the previous lessons, we set the pinModes respectively.

```
void setup() {
  // initialize the LED pin as output
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as input
  pinMode(buttonPin, INPUT);
}
```



```
void loop() {
  // read the state of the pushbutton value
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH
  if (buttonState == HIGH) {
    // turn LED on
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off
    digitalWrite(ledPin, LOW);
  }
}
```

digitalRead() will return either 1 or 0, so those are the only values buttonState can be. The Arduino constant **HIGH** automatically equals an integer of value 1, and the constant **LOW** equals an integer of value 0. You can think of this like a **boolean** value, meaning it will always be **true** or **false**, or **1** or **0** respectively.

The **double equals (==)** symbol compares the two values and returns true if they are equal, false otherwise. In this case, if the buttonState is 1, buttonState == HIGH will return true and the first case of the if statement will run and turn on the LED. If it was 0, the else part of the if statement would run and turn off the LED.

4. Compile and Verify your program.

RESULT

Now, you can compile and upload your sketch onto the board. After doing so, you can press the button with the pin number 4 and the blue LED should light up!

Now It's Your Turn!

- Can you try to make each color light up every time you press a different button?
- Can you adjust the RGB values of a Neopixel by using the 3 buttons?

Lesson B – 4: Make it Buzz!



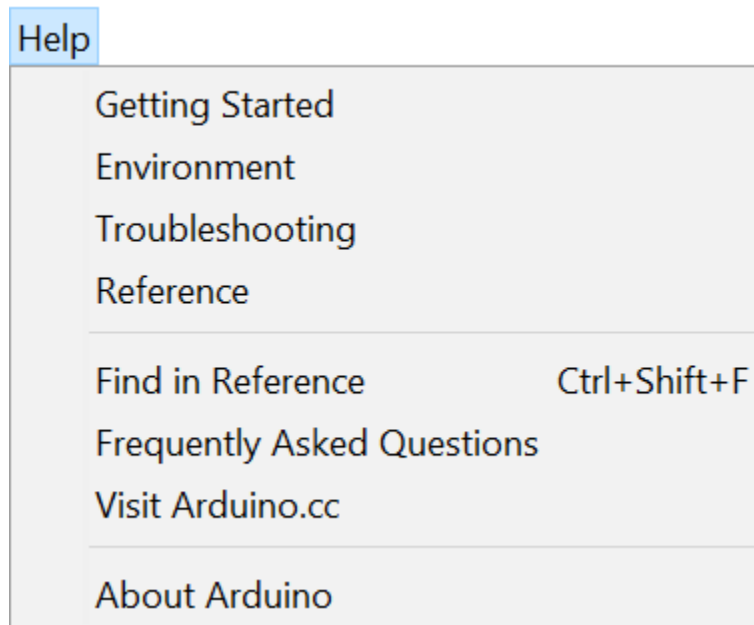
OVERVIEW

In this lesson we will:

- Learn about the Piezo Buzzer
- Write a simple program to make the Piezo sound

Teacher Guide

In this lesson, the students will be interacting with the Piezo Buzzer. In this lesson we will look into the Help tab on the IDE Menu. The Help tab will be very useful when trying to troubleshoot problems as it contains information about the device and troubleshooting information. Additionally, it has sections for FAQs as well as a reference page and a link to the Arduino website.



Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Piezo buzzer, short for **piezoelectric speaker**, is a loudspeaker that uses the piezoelectric effect for generating sound. The word “Piezoelectric” comes from the Greek word “piezo,” which means to push. Piezoelectric effect occurs when certain materials generate an electric charge as a result of an applied mechanical stress. This motion is typically converted into audible sound like a buzz.



Piezoelectric speakers are often used in computers, alarms, timers, watches and other electronic devices to generate sound. They are also used as tweeters in less-expensive speaker systems, such as computer speakers and portable radios.

Procedure

1. Set the pin numbers for the Button and the Buzzer as well as the status of the Button.

```
const float midC = 256.0; // the hz value of middle C
const int piezoPin = 11; // the number of the buzzer pin
const int buttonPin = 4; // the number of the pushbutton pin
int buttonState = 0; // variable for reading the pushbutton status
```

2. Input the following Setup() function.

```
void setup() {
  // initialize the pushbutton pin as input
  pinMode(buttonPin, INPUT);
  // initialize the buzzer pin as output
  pinMode(piezoPin, OUTPUT);
}
```

Float – a variable type used to store values that contain decimals (ex: 65.603 or 12.0)

Just like we did in the previous lessons, we set the **pinMode(s)** respectively.

3. Input the following `loop()` function.

```
void loop() {
  // read the state of the pushbutton value
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed
  // If it is, the buttonState is HIGH and turn on the buzzer
  if (buttonState == HIGH) {
    myTone(piezoPin, midC, 1); // play middleC for 1 millisecond
  }
}
```

Just like from the last lesson, we check if Button is pressed; if so, we play a sound on the buzzer.

4. Create the `myTone` method to play the frequency (in hertz) on the buzzer.

```
void myTone(byte pin, uint16_t frequency, uint16_t duration)
{ // input parameters: Arduino pin number,
  // frequency in Hz, duration in milliseconds
  // plays the tone we want by turning the buzzer on and off
  unsigned long startTime=millis();
  unsigned long halfPeriod= 1000000L/frequency/2;
  while (millis()-startTime< duration)
  {
    // turn buzzer on then delay then turn buzzer
    // off and repeat for the duration length
    digitalWrite(pin,HIGH);
    delayMicroseconds(halfPeriod);
    digitalWrite(pin,LOW);
    delayMicroseconds(halfPeriod);
  }
}
```

Method – a procedure to run a block of code that can be called just once or multiple times. Also, it is similar to a function but returns nothing or **void**

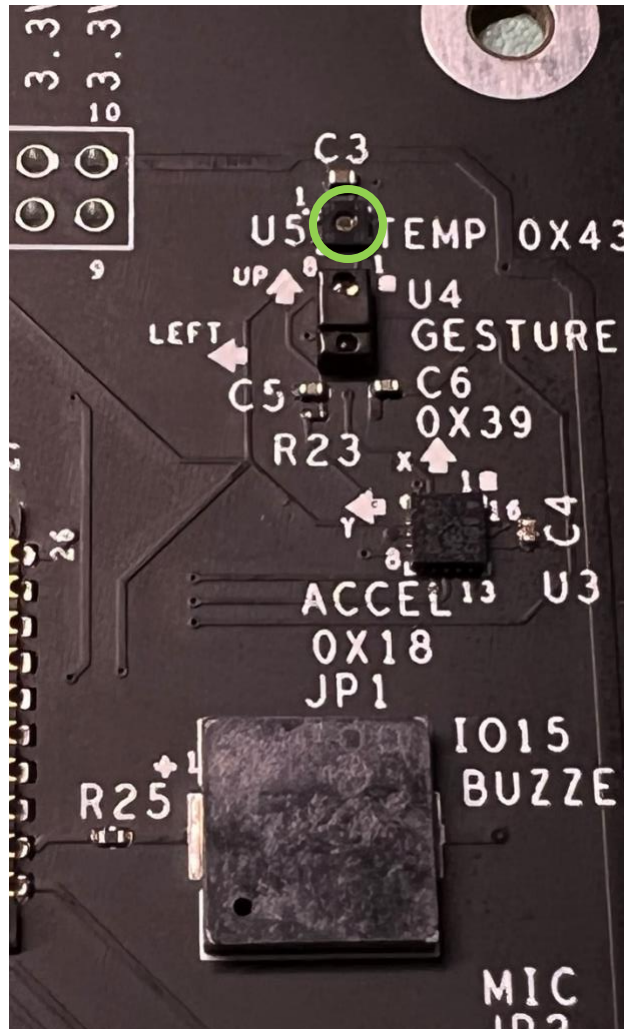
Result

After you compile and upload the sketch, try pressing the button at pin 4! It should cause the buzzer to sound for as long as you hold the button.

Now It's Your Turn!

- Can you make the buzzer play different sounds for each button by modifying the existing code?

Lesson B – 5: What's the Temp?



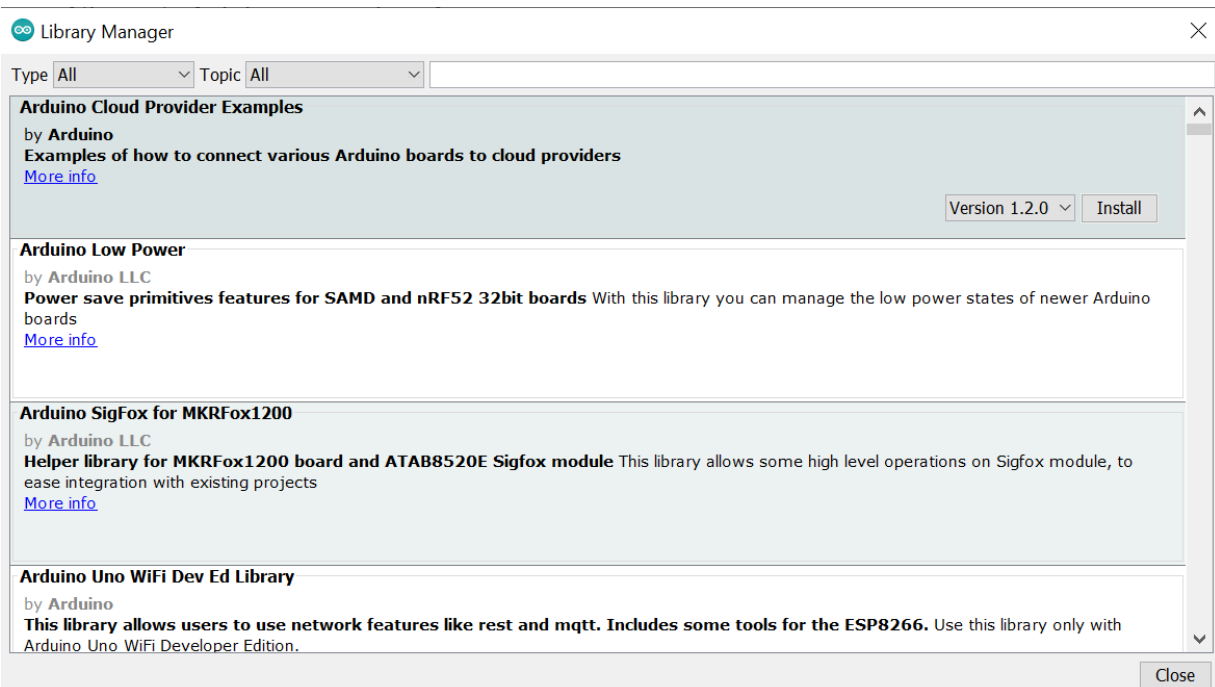
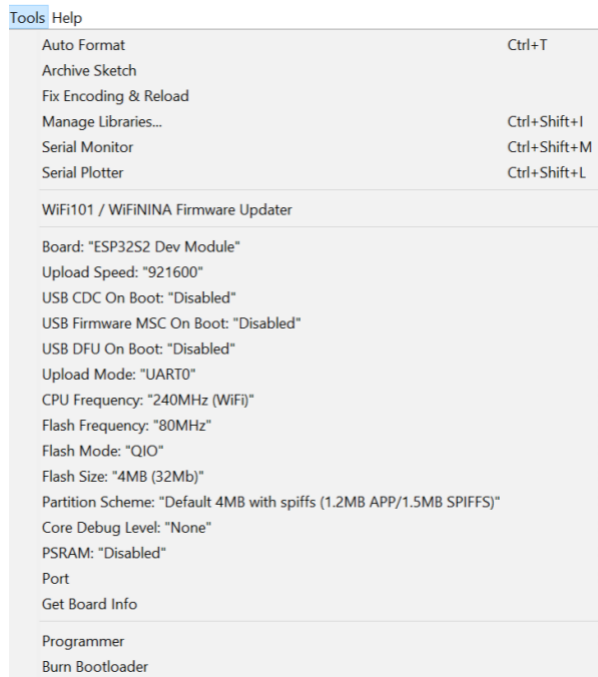
OVERVIEW

In this lesson we will:

- Learn how the temperature sensor works
- Write a simple program to get the current temperature
- Monitor if the temperature has increased or decreased

Teacher Guide

In this lesson, the students will be interacting with the temperature sensor on the device. This allows them to monitor the temperature and humidity. An important feature from this lesson and beyond is the Manage Libraries... tab which brings up a dialog box that you can search for libraries to install. This can be found in the Tools tab in the Arduino IDE. Additionally, for this lesson please review the installation steps for the ENS210 library as this will be downloaded from an online site, GitHub.



Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Temperature sensors are used in diverse applications such as food processing, HVAC environmental control, medical devices, chemical handling and automotive under the hood monitoring (e.g., coolant, air intake, cylinder head temperatures, etc.). Temperature sensors tend to measure heat to ensure that a process is either, staying within a certain range, providing safe use of that application, or meeting a mandatory condition when dealing with extreme heat, hazards, or inaccessible measuring points.



There are two main types: contact and noncontact temperature sensors.

Contact sensors include thermocouples and thermistors that touch the object they are to measure, and noncontact sensors measure the thermal radiation a heat source releases to determine its temperature. The latter group measures temperature from a distance and often are used in hazardous environments.

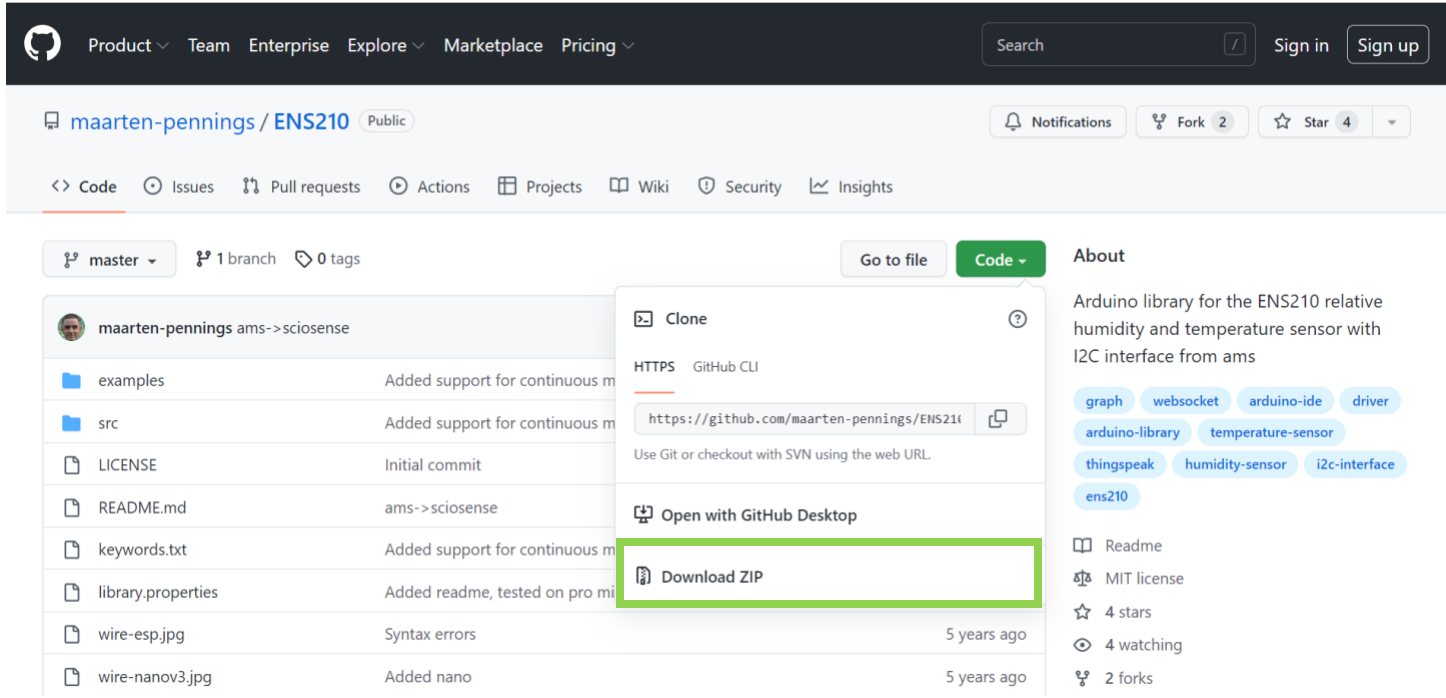
The measurement of the temperature sensor is about the hotness or coolness of an object. The working base of the sensors is the voltage that read across the diode. If the voltage increases, then the temperature rises and there is a voltage drop between the transistor terminals of base & emitter, they are recorded by the sensors. If the difference in voltage is amplified, the analogue signal is generated by the device and it is directly proportional to the temperature.

Procedure

First, we must install the “ENS210” library.

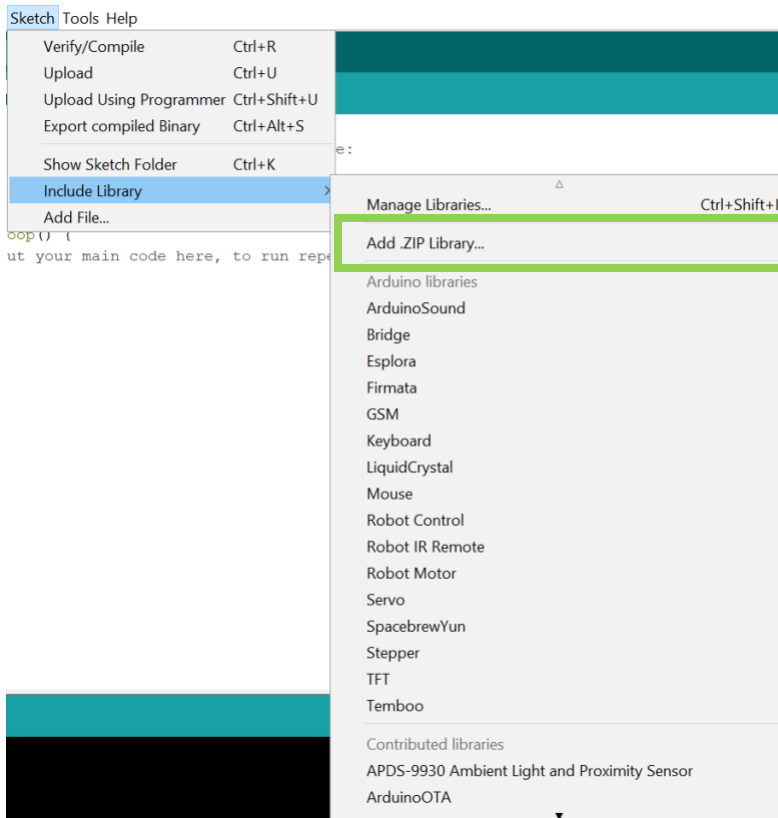
1. Click [here](https://github.com/maarten-pennings/ENS210) to open the GitHub page for the ENS210 library.
Here is the URL: <https://github.com/maarten-pennings/ENS210>

2. Click the green “Code” button, then in the dropdown, click “Download ZIP”



The screenshot shows the GitHub interface for the repository 'maarten-pennings / ENS210'. The 'Code' button is highlighted in green, and its dropdown menu is open, showing options like 'Clone', 'Open with GitHub Desktop', and 'Download ZIP'. The 'Download ZIP' option is highlighted with a green border. The repository page shows a file tree with folders like 'examples' and 'src', and files like 'LICENSE', 'README.md', 'keywords.txt', 'library.properties', 'wire-esp.jpg', and 'wire-nanov3.jpg'. The 'About' section on the right provides details about the library, including its description and tags like 'graph', 'websocket', 'arduino-ide', 'driver', 'arduino-library', 'temperature-sensor', 'thingspeak', 'humidity-sensor', 'i2c-interface', and 'ens210'.

3. Next, open the Arduino IDE and click Sketch > Include Library > Add .ZIP Library...



The screenshot shows the Arduino IDE menu structure. The 'Sketch' menu is open, and the 'Include Library' option is selected. The 'Include Library' submenu is also open, and the 'Add .ZIP Library...' option is highlighted with a green border. The 'Add .ZIP Library...' option is highlighted with a green border. The 'Add .ZIP Library...' option is highlighted with a green border. The 'Add .ZIP Library...' option is highlighted with a green border.

4. Navigate to the downloaded ZIP file titled “ENS210-master” and select it, then click Open.

5. Include the following ENS210 library and create an ENS210 object.

```
#include <Wire.h> // I2C library
#include "ens210.h" // ENS210 library

// object that will be used to get temperature values
ENS210 ens210;
```

6. Input the following setup() function:

```
void setup() {
  // Enable serial
  Serial.begin(115200);

  // Enable I2C
  Wire.begin();

  // Enable temperature sensor
  ens210.begin();
}
```

7. Input the following loop() function:

```
void loop() {
  // creates variables for temperature and humidity and measures new values
  int t_data, t_status, h_data, h_status;
  ens210.measure(&t_data, &t_status, &h_data, &h_status);
  Serial.print("Temp: ");
  // converts temperature to Fahrenheit
  Serial.print( ens210.toFahrenheit(t_data, 10) / 10.0 );
  Serial.print(" F");
  Serial.println();
  //waits 2 seconds
  delay(2000);
}
```

This library reads temperature and humidity values at the same time, but we do not use the humidity values in this example.

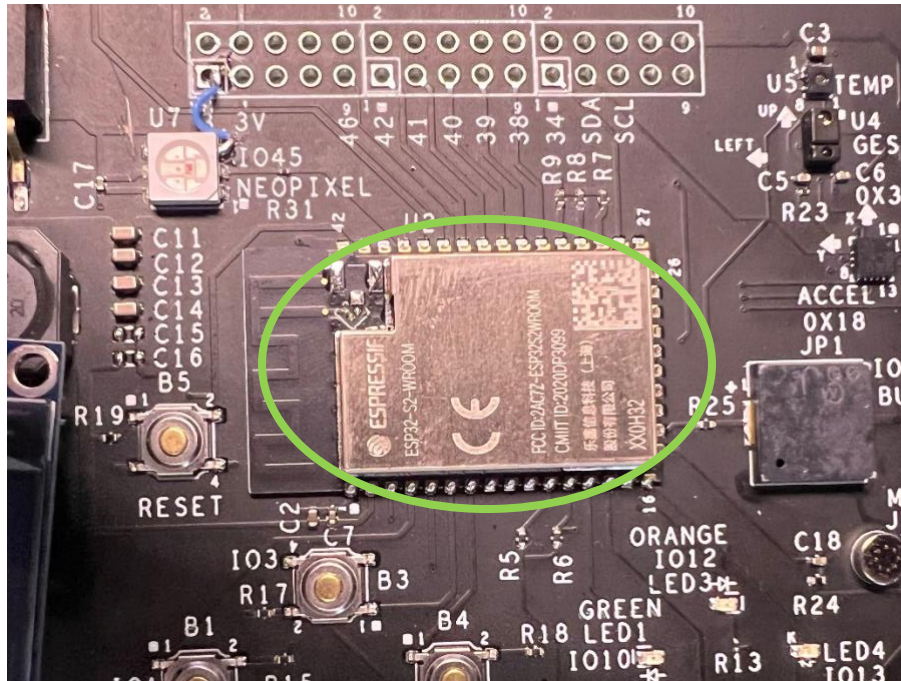
Result

After you compile and upload the sketch, open your serial monitor and you should be able to see the current temperature! Every 2 seconds, the board will update the temperature and tells you if it has increased or decreased! Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you try to light up the red LED when the temperature is over 80° Fahrenheit and blue LED when the temperature is under 40° Fahrenheit?
- Can you print the humidity out to the serial and turn on the LEDs accordingly?

Lesson B – 6: EEPROM



OVERVIEW

In this lesson we will:

- Learn what is EEPROM
- Write a simple program to write your name and age to EEPROM and then read it/print to Serial

Teacher Guide

In this lesson, the students will be interacting with EEPROM. This is located inside of the CPU and is a small hard drive location. However, this device has a limited lifetime (do not worry it is very large and can be written to about 100,000 times) so make sure that students do not run this program lots of times to prevent hardware damage.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

EEPROM stands for electrically erasable programmable read-only memory. This type of memory can be easily erasable and reprogrammed using pulses of **voltage**. In the Lenovo Educational Board, EEPROM is stored in the CPU. EEPROM is typically a small amount of data and does not forget content if power supply is lost. The data that is stored, can be erased byte-by-byte and uses electricity for deletion.

This type of technology has limitless applications as it is used in smart cards and remote keyless systems. Since these systems are relatively cheap (compared to other systems like RAM), we integrated this small programmable memory into the Lenovo Educational Board.

Procedure

In this lesson, we will be writing and reading variables from the EEPROM.

1. First, create a blank sketch and save it as EEPROM.
2. Include the EEPROM library and create EEPROM objects:

```
#include "EEPROM.h"

// Create eeprom objects with name and size
EEPROMClass NAMES("eeprom0");
EEPROMClass HEIGHT("eeprom1");
EEPROMClass AGE("eeprom2");
```

3. Input the following **setup()** function to initialize EEPROM and declare variables:

```
void setup() {
  // begin serial
  Serial.begin(115200);
  // wait 1 second for initialization
  delay(1000);
  // begin EEPROM classes with space in memory
  NAMES.begin(0x500);
  HEIGHT.begin(0x200);
  AGE.begin(0x100);

  // variables to be written to and read from EEPROM
  const char* name = "John Smith"; // change to your name
  char rname[32];
  double height = 5.8; // change to your height (feet.inch or meters)
  uint32_t age = 47; // change to your age
```

4. Write the variables into the EEPROM inside the `setup()` function:

```
// Write: Variables ---> EEPROM stores
// use put method to store in EEPROM
NAMES.writeString(0, name);
HEIGHT.put(0, height);
AGE.put(0, age);
Serial.println("Writing to EEPROM:");
Serial.print("name: "); Serial.println(name);
Serial.print("height: "); Serial.println(height);
Serial.print("age: "); Serial.println(age);
Serial.println("-----\n");
```

5. Clear the variables inside the `setup()` function:

```
// Clear variables
rname[0] = '\0';
height = 0;
age = 0;
Serial.println("Clearing variables");
Serial.print("name: "); Serial.println(rname);
Serial.print("height: "); Serial.println(height);
Serial.print("age: "); Serial.println(age);
Serial.println("-----\n");
```

6. Read the variables from the EEPROM inside the `setup()` function:

```
// Read: Variables <--- EEPROM stores
// use get method to read from EEPROM
NAMES.get(0, rname);
HEIGHT.get(0, height);
AGE.get(0, age);
Serial.println("Reading from EEPROM:");
Serial.print("name: "); Serial.println(rname);
Serial.print("height: "); Serial.println(height);
Serial.print("age: "); Serial.println(age);

Serial.println("Done!");
}
```

7. Input the following `loop()` function:

```
void loop() {
  delay(0xFFFFFFFF); // max delay value to prevent looping
}
```

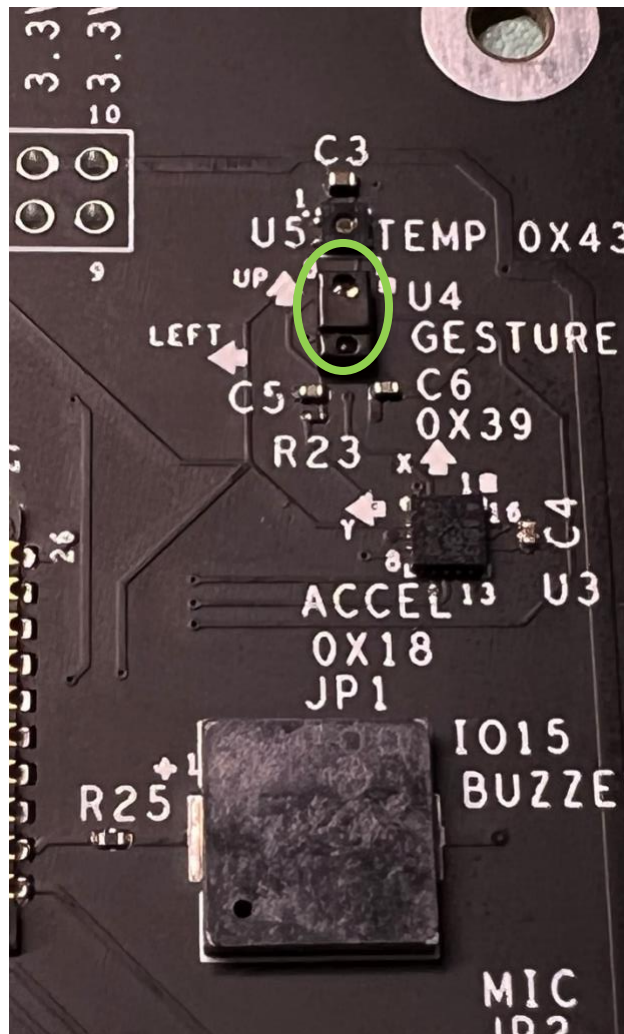
Result

After you compile and upload the sketch, open the Serial to see the variables written and read to and from the EEPROM. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you write and read other values to and from the EEPROM besides name, height, and age?

Lesson B – 7: Color Sensor



OVERVIEW

In this lesson we will:

- Learn how a color sensor detects colors
- Write a simple program to get the RGB values of a color using the color sensor

Teacher Guide

In this lesson, the students will be interacting with the color sensor on the device. In order to read the color of an object, it should be bright and well lit. Also, hold the object around 6-12 inches from the sensor. Additionally, if color readings are wrong, try looking up the color on Google and point the sensor at the screen. The higher the value of a color, the more of it that is in the object. This lesson uses an object from a library to get color readings from the device.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

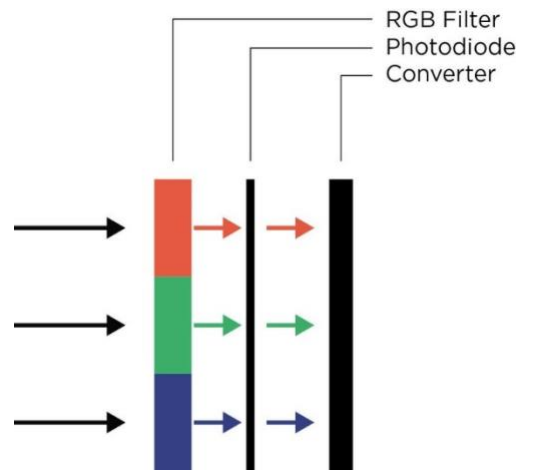
Did you define the PIN number correctly in your program?

Background Info

Do you recall having to mix two colors together to create another color in art class? For example, you can mix an equal amount of blue and yellow together to create green! In the electronic world, we use the colors Red, Green, and Blue to create a useful array of colors for your digital camera, television, and computer screens. This concept is known as the RGB value. Each value ranges from 0 to 255. Let's say you want to purple to display, then your RGB value would look something like this: (128, 0, 128). The first value represents the intensity of the color red, the second is green, and the third is blue.

Well, what if you have no idea what the RGB for a specific color that you need? There are a couple of ways you can find the RGB value for a specific color. You can search online for an RGB Color Codes Chart and select the color that is closest to your preference. But if this is not good enough, you can use a Color Sensor to get the exact RGB value.

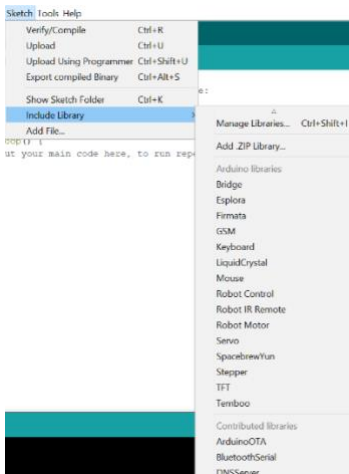
A color sensor detects the color of a surface. It casts white light onto an object and records the reflected color, as well as the intensity of the reflection. Then the photodiode converts the amount of light to current through the red, green and blue filter. The converter then converts the current into voltage so that the Arduino board can read.



Procedure

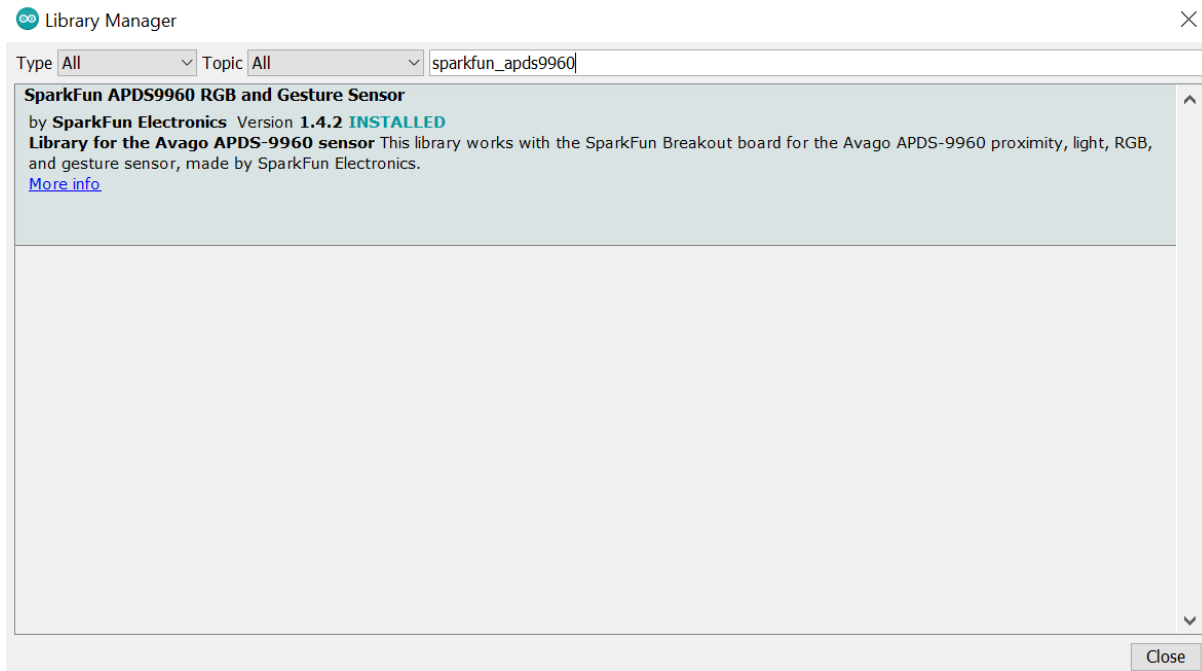
In this lesson, we will be able to put an object up to the board and it will be able to tell what color of red, green, and blue it primarily consists of. Also, if it has an equal amount of two colors it will tell us.

1. Install the SparkFun_APDS9960 library by clicking Sketch > Include Library > Manage Libraries.



2. In the resulting dialog box, type **SparkFun_APDS9960** into the search bar.

3. Navigate to **SparkFun_APDS9960** and click **Install** then **Close**.



4. Import the **SparkFun_APDS9960** library, create **SparkFun_APDS9960** object, and declare variables:

```
// include the following libraries
#include <Wire.h>
#include <SparkFun_APDS9960.h>

// APDS object to read color data from sensor
SparkFun_APDS9960 apds = SparkFun_APDS9960();
// Global variables to keep track of color data
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;
```

5. Input the following **setup()** function.

```
void setup() {
  // Initialize Serial port
  Serial.begin(9600);
  Serial.println();

  // Initialize APDS
  apds.init();
  apds.enableLightSensor(false);
  // Wait for initialization and calibration to finish
  delay(500);
}
```

This initializes the APDS object as well as setting up the color sensor. It also sets the baud rate of the Serial to 9600 which is the rate information is transferred along a communication channel.

6. Input the following `loop()` function.

```
void loop() {
  // Read the light levels (red, green, blue)
  // if data is available
  if ( !apds.readRedLight(red_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ) {
    Serial.println("Error reading light values");
  } else {
    // print red, green, blue values to serial
    Serial.print(" Red: ");
    Serial.print(red_light);
    Serial.print(" Green: ");
    Serial.print(green_light);
    Serial.print(" Blue: ");
    Serial.println(blue_light);
  }

  // Wait 1 second between readings
  delay(1000);
}
```

This section of the code waits every 1 second before getting new color data from the sensor and printing the R,G,B values to the Serial.

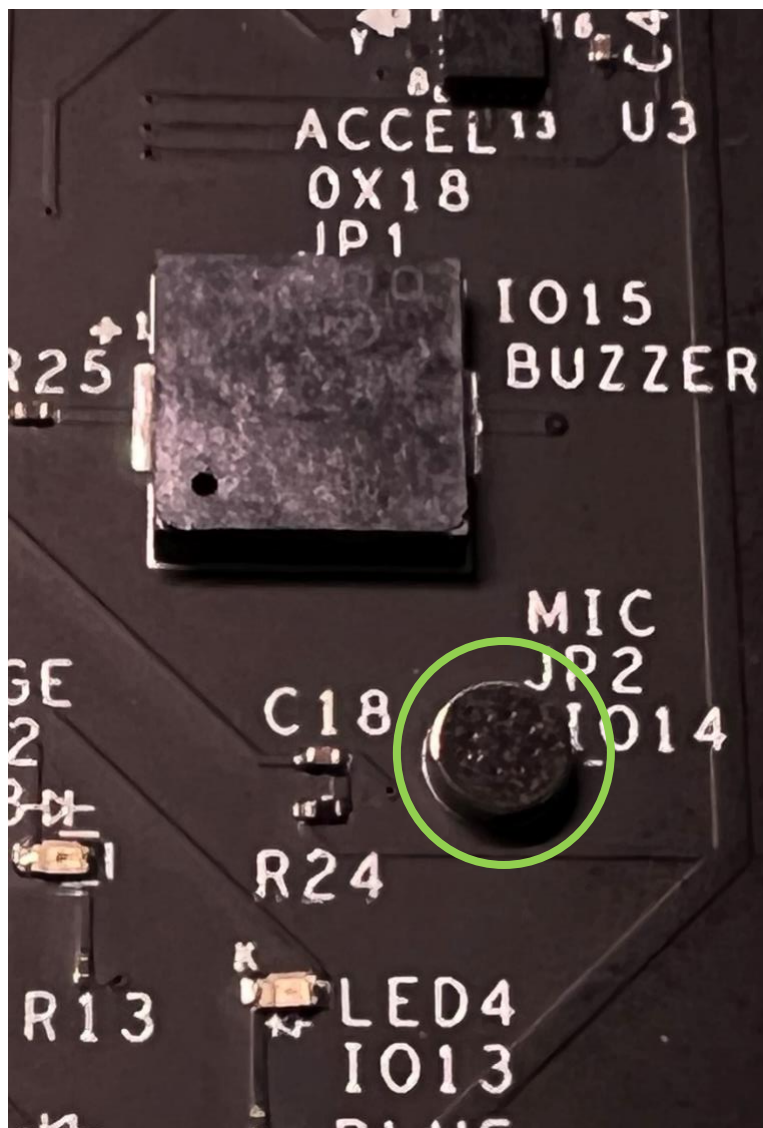
Result

After you compile and upload the sketch, open the Serial Monitor and place various objects in front of the color sensor and you will see what color the object is primarily of. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you make a sketch to turn the Neopixel LED to the exact color that the RGB sensor detects?

Lesson B – 8: What Was That Sound?



OVERVIEW

In this lesson we will:

- Learn how a microphone works
- Write a simple program to detect noise with the Microphone

Teacher Guide

In this lesson, the students will be interacting with the microphone. This device is a simple setup where values are read from the microphone, then accordingly, a LED is turned on. While certain LEDs may not turn on due to different values not being met, please tell students that if the value is low (green LED on) to blow gently on the microphone. However, if the LED values are high (blue LED on), tell students to be quiet and remove loud noises from the room to see other LEDs illuminate.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Have you ever had to speak through a microphone because your voice too small? With a microphone, you can amplify your voice through a speaker so everyone can hear you! A microphone is an acoustic to electric transducer or sensor that detects sound signals and converts them into an electrical signal. A microphone sound sensor detects sound and gives a measurement of how loud a noise is.

What can we do with a Microphone?

In this lesson, we will go over how to connect a microphone to an Arduino so that the Arduino can detect whether there is sound in the environment or not.

This circuit is only capable of detecting whether there is sound in the environment or not. It won't be able to record the words that you say, with the ability of being able to play it back. Or do things such as record music or play it back.

The microphone simply serves to detect whether there is sound. The Arduino also has the capability of measuring the level of the sound, so that only sounds above a certain threshold can a certain circuit action.

An application of using a microphone with an Arduino is if you want to build some type of sound alarm circuit, in which sound triggers a circuit event such as flashing sirens or turning on a buzzer. For example, if you have a room that should be very quiet and someone breaks into it, making a lot of noise, a microphone placed in an Arduino can easily trigger another electronic component. Another application could be a clap-on-off circuit, in which clapping can turn the circuit on or off. Basically, any circuit that needs to be triggered by sound can be done with a microphone as input into an Arduino.



Procedure

The Lenovo Educational board is equipped with a microphone, so you can hear noise around you. For this lesson, we will make different LEDs light up for different volumes of noise.

1. Create the following constants and variables above the `setup()` function.

```
const int micPin = 10; // microphone pin number
int micVal = 0; // value read from the microphone

const int GreenledPin = 15; // green LED pin
const int YellowledPin = 16; // yellow LED pin
const int OrangedPin = 17; // orange LED pin
const int BlueledPin = 18; // blue LED pin
int ledState = HIGH;
```

Here we declared the pin values for the LEDs and the microphone. In addition, we declared two variables: one to read values from the microphone, and the other to turn the LEDs on.

2. Input the following `setup()` function to set the `pinModes` for the LEDs.

```
void setup() {
  // set the LEDs to output
  pinMode(GreenledPin, OUTPUT);
  pinMode(YellowledPin, OUTPUT);
  pinMode(OrangedPin, OUTPUT);
  pinMode(BlueledPin, OUTPUT);
}
```

3. Input the following `loop()` function.

```
void loop() {
  // read the current microphone value
  micVal = analogRead(micPin);
  if (micVal < 100) // if value is less than 100 turn on the green LED
  {
    digitalWrite(GreenledPin, ledState);
    digitalWrite(YellowledPin, LOW);
    digitalWrite(OrangedPin, LOW);
    digitalWrite(BlueledPin, LOW);
  }
  else if (micVal >= 100 && micVal < 500)
  // if value is less than 500 and greater than or equal to 100 turn on the yellow LED
  {
    digitalWrite(YellowledPin, ledState);
    digitalWrite(GreenledPin, LOW);
    digitalWrite(OrangedPin, LOW);
    digitalWrite(BlueledPin, LOW);
  }
  else if (micVal >= 500 && micVal < 1000)
  // if value is less than 1000 and greater than or equal to 500 turn on the orange LED
  {
    digitalWrite(OrangedPin, ledState);
    digitalWrite(GreenledPin, LOW);
    digitalWrite(YellowledPin, LOW);
    digitalWrite(BlueledPin, LOW);
  }
  else // if none of the above conditions are met (>= 1000) then turn on the blue LED
  {
    digitalWrite(BlueledPin, ledState);
    digitalWrite(GreenledPin, LOW);
    digitalWrite(YellowledPin, LOW);
    digitalWrite(OrangedPin, LOW);
  }
  // wait .3 seconds between readings
  delay(300);
}
```

In this section, we read the value from the microphone and turn on different colored LEDs accordingly.

Result

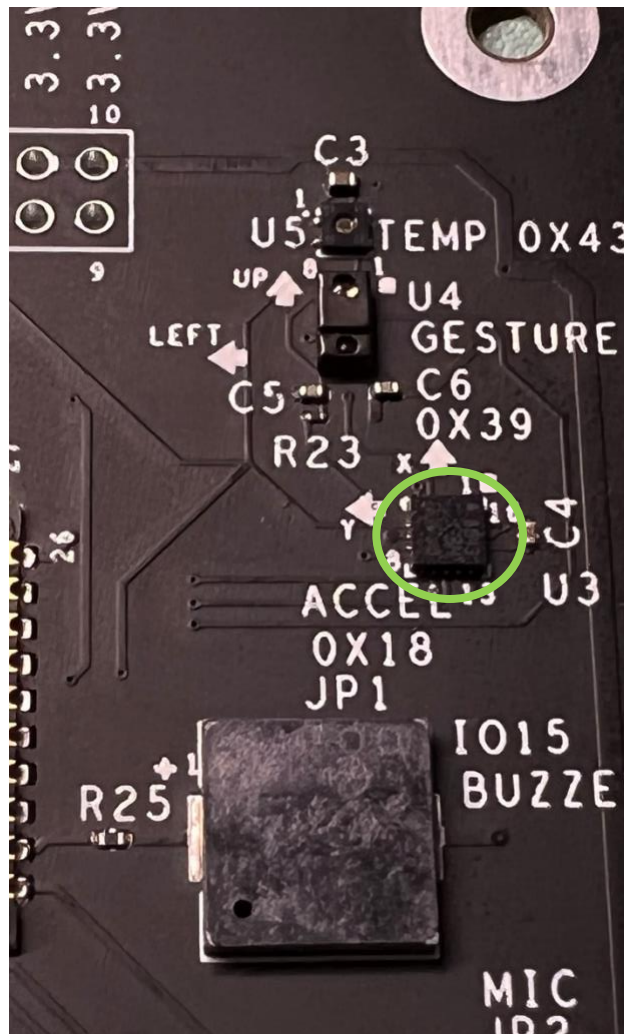
After you compile and upload the sketch, you should see different colored LEDs light up according to the value that is read from the microphone. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Note: For Rev 2 of the Lenovo Educational Board, your microphone may misread values leading to unexpected LEDs lighting up

Now It's Your Turn!

- Can you make the Buzzer sound and the Neopixel flash red when the microphone value is greater than 1000?

Lesson B – 9: Which Way Is Up?



OVERVIEW

In this lesson we will:

- Learn how an accelerometer works
- Write a simple program to get the position and orientation of the Lenovo Educational board

Teacher Guide

In this lesson, the students will be interacting with the accelerometer. This device measures orientations of objects as well as forces (i.e., the force of hitting the floor if the device is dropped). Please tell students to not attempt to drop the board or hit it with a force as this will damage the board and this lesson focuses on the orientation of the board. Make sure that students set the baud rate to 115200 to read the different orientations from the Serial. Additionally, we recommend starting the board on a flat surface before rotating it to see the different orientations more clearly.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

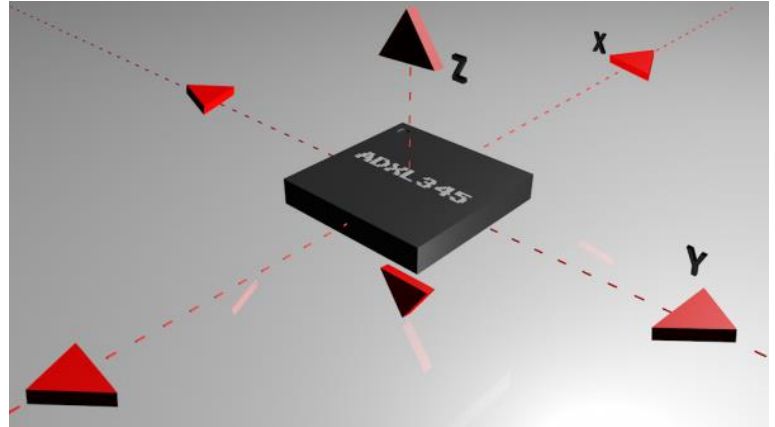
Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s^2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to $9.8 m/s^2$, but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications.



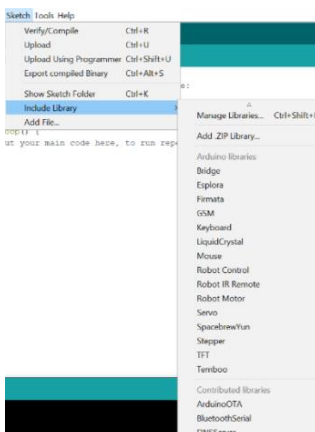
How are accelerometers being used?

Accelerometers have multiple applications in industry and science. Highly sensitive accelerometers are components of inertial navigation systems for aircraft and missiles. Accelerometers are used to detect and monitor vibration in rotating machinery. Accelerometers are used in tablet computers and digital cameras so that images on screens are always displayed upright. Accelerometers are used in drones for flight stabilization. Coordinated accelerometers can be used to measure differences in proper acceleration, particularly gravity, over their separation in space

Procedure

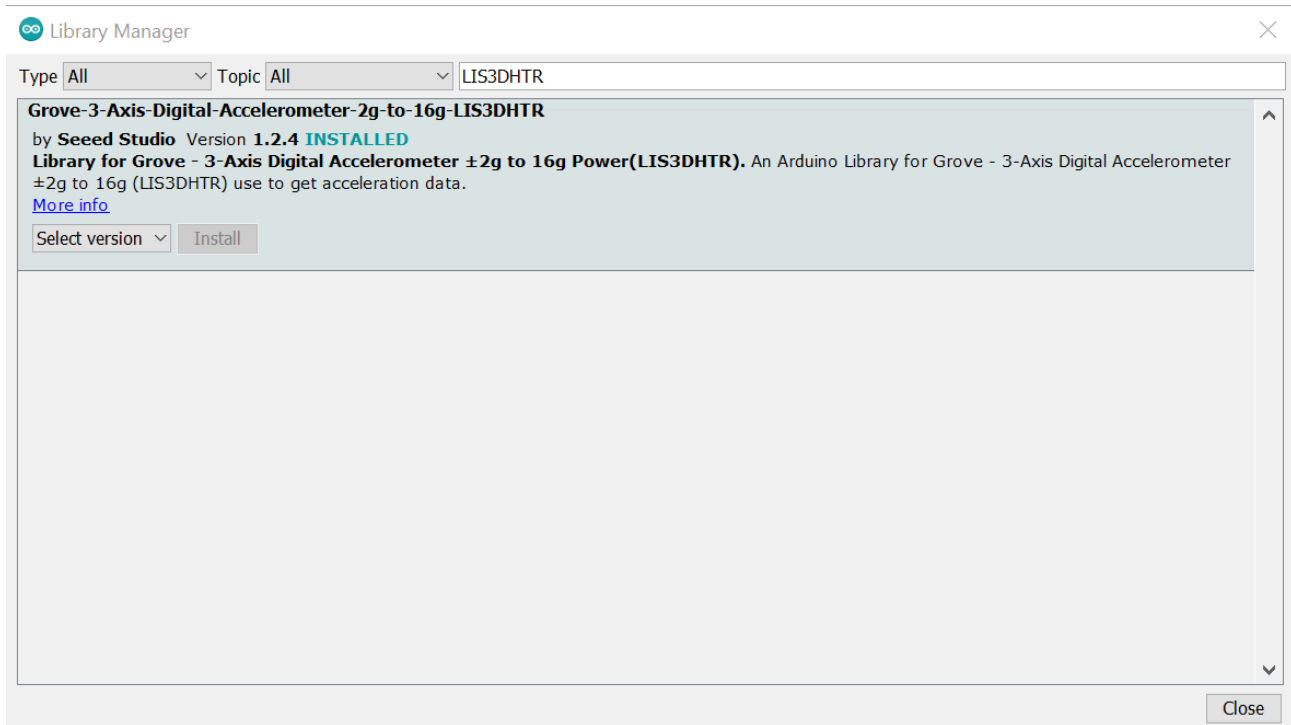
For this lesson, we will work with the Accelerometer on the board to print the orientation i.e., the direction that the board is tilted, to the Serial Monitor.

1. Install the LIS3DHTR library by clicking Sketch > Include Library > Manage Libraries.



2. In the resulting dialog box, type **LIS3DHTR** into the search bar.

3. Navigate to LIS3DHTR and click Install then Close.



4. Input the following library statement and object declaration above the **setup()** function.

```
#include <LIS3DHTR.h> // include the libraries for reading the accelerometer
#include <Wire.h>

LIS3DHTR<TwoWire> LIS; //create object
#define WIRE Wire
```

5. Create the following variables above the **setup()** function.

```
float x = 0; // create x coordinate
float y = 0; // create y coordinate
float z = 0; // create z coordinate
```

6. Input the following **setup()** function.

```

void setup() {
  Serial.begin(115200); // set baud rate to 115200

  // turn on the accelerometer
  pinMode(2, OUTPUT);
  digitalWrite(2, HIGH);
  // set up accelerometer
  Wire.begin(8, 9);
  LIS.begin(Wire);
  delay(100);
  LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ);
  LIS.setHighSolution(true);
}

```

This section sets up and turns on the Accelerometer as well as setting the baud rate for the Serial.

7. Input the following `loop()` function to get the x, y, and z values.

```

void loop() {
  // find the current x, y, and z orientation
  x = LIS.getAccelerationX();
  y = LIS.getAccelerationY();
  z = LIS.getAccelerationZ();
}

```

8. Input the following into the `loop()` function to read the orientation:

```

// using x, y, and z values read the orientation
if (x >= .40){
  Serial.println("Lenovo Educational Board is held rotated forward.");
}
if (x <= -.40){
  Serial.println("Lenovo Educational Board is held rotated backward.");
}

if (y >= .40){
  Serial.println("Lenovo Educational Board is held slanted to the right.");
}

if (y <= -.40){
  Serial.println("Lenovo Educational Board is held slanted to the left.");
}

if (z >= .80){
  Serial.println("Lenovo Educational Board is held flat.");
}

if (z <= -.80){
  Serial.println("Lenovo Educational Board is held upside down.");
}
}

```

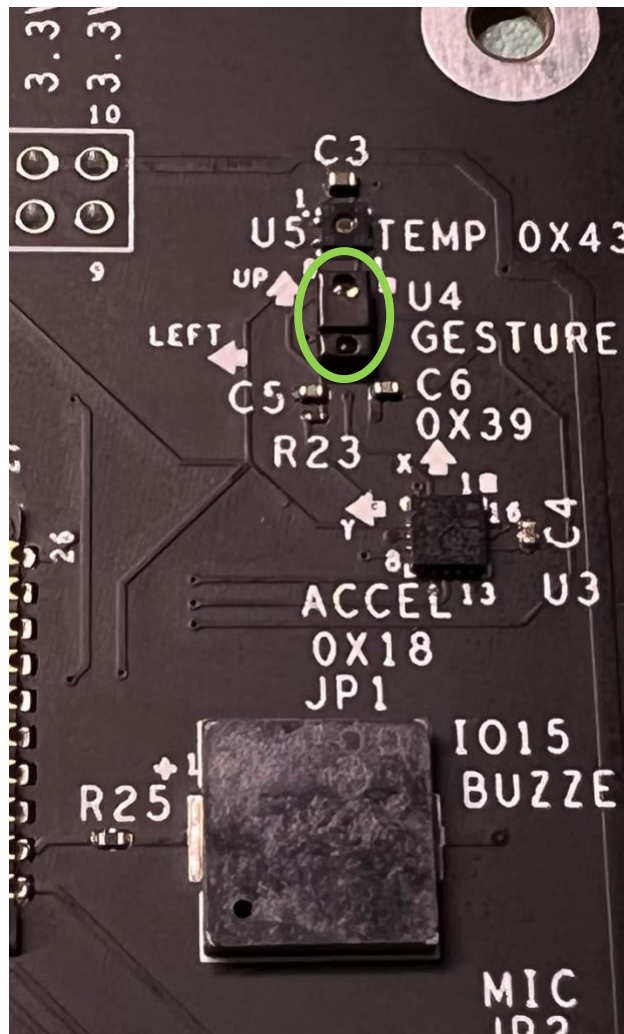
Result

After you compile and upload the sketch, you should open the Serial Monitor and see the different directions that the board is tilted as you move it around. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you make different LEDs turn on based on the orientation of the board or make the Neopixel turn red when the board is upside down?

Lesson B – 10: Detect Brightness Levels



OVERVIEW

In this lesson we will:

- Learn how the ambient light sensor works
- Write a simple program to get the current light level

Teacher Guide

In this lesson, the students will be interacting with ambient light which is how much light that there is in a given place. Please make sure that students have the Serial open with a baud rate of 9600 to see output. Additionally, you can turn off the lights in the classroom or tell students to cover the sensor with their hand to prevent light from getting to it to allow students to see the “It is dark here.” output. Likewise, make sure that their sensor faces light to see the “It is light here.” output as the sensor being in the shadows can affect output.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

What is an ambient light sensor?

An **ambient light sensor** is a photodetector that is used to sense the amount of ambient light present. It provides measurements of ambient light intensity which match the human eye's response to light under a variety of lighting conditions. Ambient light sensors are typically used to mimic the response of the human eye in some regard, or at least be applied to an electronic component that directly relates to the human experience under ambient lighting conditions. They are commonly used to adjust display brightness of a certain device based on the brightness of the outside environment. Apart from other photovoltaic sensors, they do not have an infrared or another light transmission element that is used to enhance elements outside the sensing device. They rely strictly on the light provided by the environment in which they reside, as do eye under the same conditions.



Within the light sensor, there is a sensing element that is sensitive to light in some manner. Usually, it is a type of diode or photoresistor which has a response to the light input that can be measured and transmitted effectively. This response is most commonly a voltage change across the discrete element that a controller can detect.

Where are light sensors being used?

An ambient light sensor is a component in smartphones, notebooks, other mobile devices, automotive displays and LCD TVs. It is a photodetector that is used to sense the amount of ambient light present, and appropriately dim the device's screen to match it. This avoids having the screen be too bright when the user's pupils are adapted for vision in a dark room, or too dim when the device is used outdoors in the daytime. In cars, ambient light sensors help in automatically turning on the headlights when outdoor lighting conditions become dark.

Procedure

In this lesson, we will learn about the Ambient Light Sensor Module and also how to interface an Ambient Light Sensor with Arduino to obtain the ambient light data and display it through the serial monitor.

1. Include the following libraries and create the SparkFun_APDS9960 object.

```
/ include the following libraries
#include <Wire.h>
#include <SparkFun_APDS9960.h>

// APDS object
SparkFun_APDS9960 apds = SparkFun_APDS9960();
// Global variable to read ambient light values
uint16_t ambient_light = 0;
```

2. Input the following setup() function.

```
void setup() {
  // Initialize Serial port
  Serial.begin(9600);
  Serial.println();

  // Initialize APDS
  apds.init();
  apds.enableLightSensor(false);
  // Wait for initialization and calibration to finish
  delay(500);
}
```

3. Input the following loop() function to read the ambient_light values.

```
void loop() {
  // Read the light levels (ambient)
  if (!apds.readAmbientLight(ambient_light)) {
    Serial.println("Error reading light values");
  } else {
    // print if it is light or dark
    if (ambient_light < 100)
    {
      Serial.println("It is dark here.");
    }
    else
    {
      Serial.println("It is light here.");
    }
  }
  // Wait 1 second before next reading
  delay(1000);
}
```

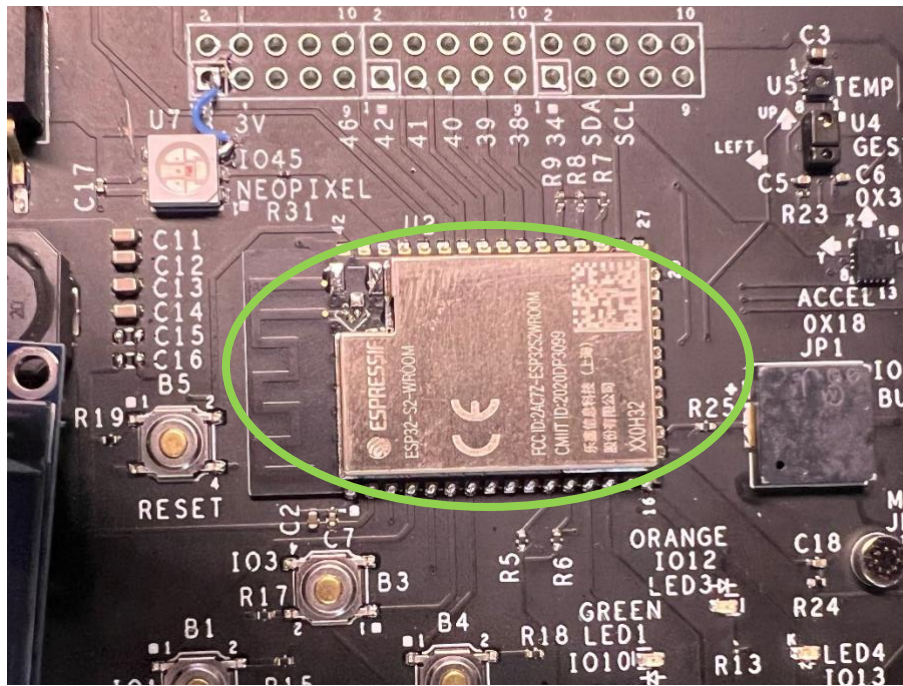
Result

You should now know how to read from the light sensor! You can take that information and create more complex programs with it. Open the Serial Monitor information page seen in the Table of Contents for additional information.

Now It's Your Turn!

- Can you make a nightlight, an LED that only turns on when it is sufficiently dark, out of this? (Hint: this is an intermediate lesson, just start thinking about it or try it on your own first!)

Lesson B – 11: WiFi Scan



OVERVIEW

In this lesson we will:

- Learn how to scan for nearby Wi-Fi networks
- Learn about different types of Wi-Fi

Teacher Guide

In this lesson, the students will be interacting with Wi-Fi on their device. This program is very short but explores some new libraries and functionalities that we have not explored yet. The Wifi.h library for this lesson does not need to be downloaded as it is built into the Arduino IDE. Additionally, we recommend having a WiFi network and password that the students can join in preparation for this lesson.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. Board Connectivity

Is your board connected to your computer?

2. COM Port

Is the IDE configuration set to the right COM port?

3. PIN number

Did you define the PIN number correctly in your program?

Background Info

How does WIFI work?

Wi-Fi is the name of a wireless networking technology that uses radio waves to provide high-speed network and Internet connections. Wi-Fi allows computers, smartphones, or other devices to connect to the Internet or communicate with one another wirelessly within a particular area. Wi-Fi networks have no physical wired connection between sender and receiver by using **radio frequency** (RF) technology -- a frequency within the electromagnetic spectrum associated with radio wave propagation. When an RF current is supplied to an antenna, an electromagnetic field is created that then is able to propagate through space.



Figure 1 - Aruba Access Point

Wireless network requires an access point (AP) to connect to multiple devices. The primary job of an access point is to broadcast a wireless signal that computers can detect and "tune" into. In order to connect to an access point and join a wireless network, computers and devices must be equipped with wireless network adapters.

What is WIFI station mode?

Wi-Fi station mode is one of the modes for a Wi-Fi device. For this lesson, we will use this mode to scan for nearby networks. This mode is often considered the normal Wi-Fi mode and oftentimes is method for a smartphone to connect to Wi-Fi. Additionally, station mode allows you to join a network that already exists whereas other Wi-Fi methods like access point (AP) which we will explore in later lessons might act as a hub for Wi-Fi connections.

Procedure

In this lesson, we will learn to use the built in Wi-Fi of the Lenovo Educational Board to scan for nearby networks that we could connect to.

1. Include the WiFi Library.

```
#include "WiFi.h"
```

2. Input the following `setup()` function.

```
void setup()
{
  Serial.begin(115200); // Set baud rate to 115200
  // Set WiFi to station mode
  // and disconnect from an access point if it was previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  Serial.println("Setup done");
}
```

This section sets the Wi-Fi mode to station and disconnects from any networks that the board was connected to.

3. Input the following loop() function.

```
void loop()
{
  Serial.println("scan start");
  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  if (n == 0) {
    Serial.println("no networks found");
  } else {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i) {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      // SSID denotes the WiFi name
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      // RSSI denotes the strength of the network 0 is near/-100 is far
      Serial.print(")");
      Serial.println();
      delay(10);
    }
  }
  Serial.println("");
  // Wait 5 seconds before scanning again
  delay(5000);
}
```

Inside of this function, we scan for networks and print the name of the Wi-Fi network or the SSID. In addition, we print the network strength or RSSI inside of parenthesis. This scales from 0 being very close and -100 being very far.

Result

Now, you should know how to look for nearby Wi-Fi networks using the Serial monitor on your Arduino IDE. Open the Serial Monitor information page seen in the Table of Contents for additional information.

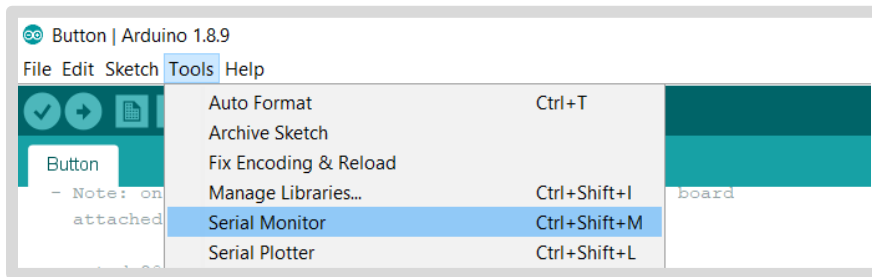
Now It's Your Turn!

- Can you test out the other Wi-Fi modes on the device to see what they do?
- Can you find your IP address to connect to a web browser? (we will do this in a later lesson)

Serial Monitor Information

How to open the Serial Monitor

1. Choose Tools > Serial Monitor.
2. Change the speed to **115200** baud from the bottom right corner the serial monitor.

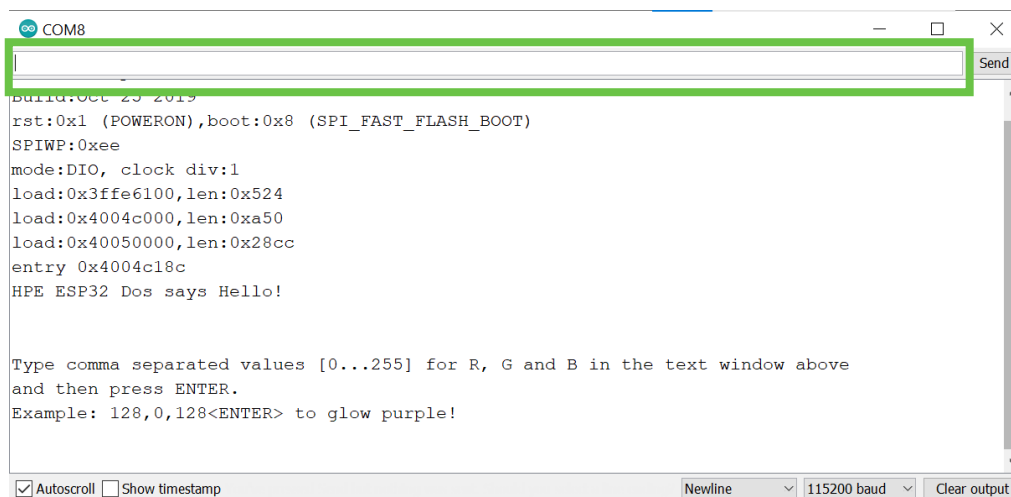


OR...

Click the magnify glass in the top right corner



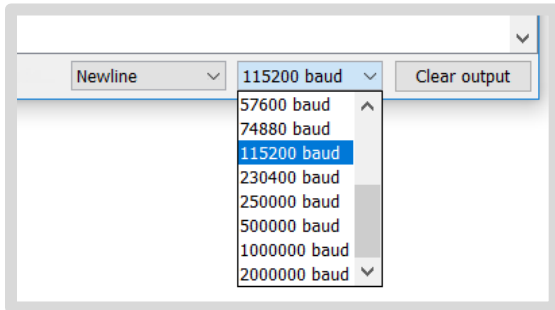
How input text in the Serial Monitor



Type text into the above textbox that is surrounded by a green rectangle.

How to change the baud rate

1. Open the Serial Monitor (see above)
2. Choose the appropriate baud rate from the drop down (we will use either 9600 or 115200 for all lessons)



Glossary of Terms

To help those unfamiliar with Arduino/coding related terms, here are some defined words that may help you throughout the course: (there might be more sprinkled throughout the lessons!)

Software Terminology

Integrated Development Environment (IDE) – a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, a compiler, build automation tools, and a debugger. The Arduino IDE allows you to write your program, compile it for your microcontroller, upload your program to the board, and interface with the microcontroller's debug serial port.

Program – a set of instructions that are written in such a way that the computer can understand and execute them.

Programming Language – the unique set of words and formatting that is used to create a program that the computer will execute. Examples include Python, C, C++, Java, etc.

Sketch – a special term used by the Arduino IDE to mean a program. Sketches are written using the C/C++ programming language.

Open Source – a software program in which the original source code is made freely for programmers to modify and reuse in their own programs.

Library – a collection of source code or precompiled modules and resources for a program to use. They consist of pre-written code, classes, values, configuration data, etc. Arduino sketches (programs) will make use of many Libraries (code written by others) to create a fully functional program.

Advanced Terminology

Firmware — a piece of permanent software that is programmed into a read-only memory (ROM).

Class — can be described as a template or a blueprint from which objects are created. It includes a set of properties and/or methods that is shared by a number of objects. An example of a class can be an animal.

Object — an instance of a class. Objects have states/attributes and behaviors. An example of an object can be a dog.

Hardware Terminology

Development board – A circuit board that contains a microcontroller and the interfaces to upload, run, and debug programs.

Microcontroller – A miniature computer all in one little package. It contains all the necessary computer components (processor, memory, and I/O devices) needed to run and interact with a program.

I/O - stands for Input/Output. Devices connected to a microcontroller are either input devices, output devices, or both. An LED is an example of an output device, whereas a button is an input device. Serial ports that both transmit and receive data are both input and output devices.

I/O pins – On a microcontroller, these are the physical pins that devices can be connected to. Microcontroller's typically have a unique number or label associated with every I/O pin they provide.

Breadboard – a solderless device for temporary prototyping with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate.

Printed Circuit Board (PCB) – a base board for physical supporting and wiring the surface-mounted and socketed components in most electronics.

Printed Circuit Assembly (PCA)/Printed Circuit Board Assembly (PCBA) – is the process of soldering or assembly of electronic components to a PCB.

Schematic – a blueprint/diagram that maps out all the elements on an electrical circuit board. It shows how devices are connected together.