Workshop III

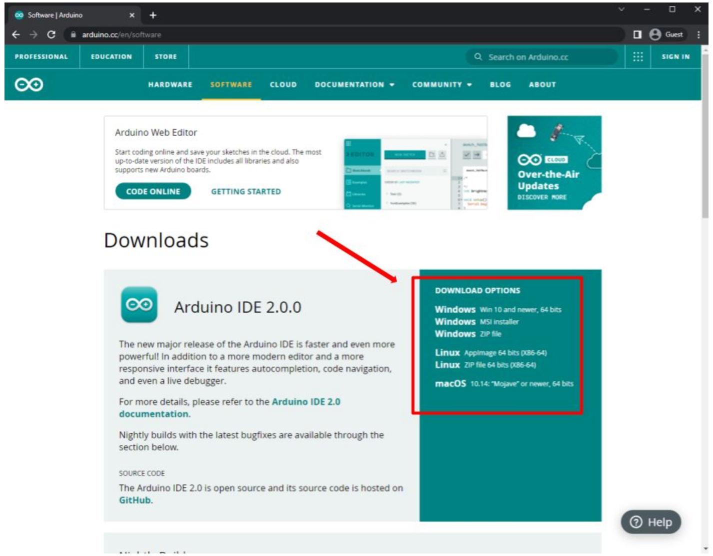
Arduino IDE Setup & Potentiometers

SECTION I

Review: ESP32 Board Setup

Arduino IDE Setup for ESP32

- If you haven't yet, visit the ESP32 Guide on the OPS Website for more information on how to setup Arduino IDE (Watch YouTube video for how-to)



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

Downloads

Arduino IDE 2.0.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the Arduino IDE 2.0 documentation.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

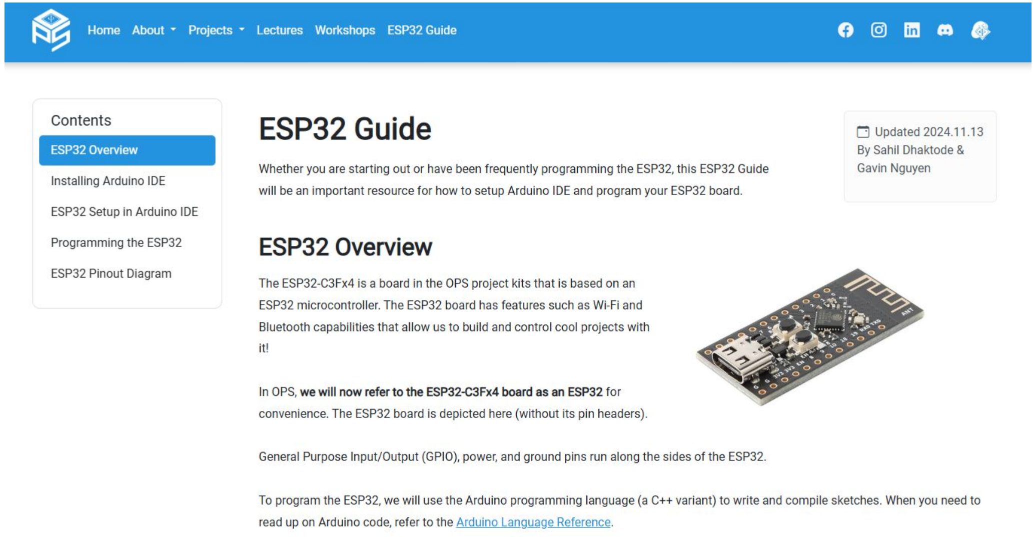
The Arduino IDE 2.0 is open source and its source code is hosted on GitHub.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS 10.14: "Mojave" or newer, 64 bits



Contents

- ESP32 Overview
- Installing Arduino IDE
- ESP32 Setup in Arduino IDE
- Programming the ESP32
- ESP32 Pinout Diagram

ESP32 Guide

Updated 2024.11.13
By Sahil Dhaktode & Gavin Nguyen

Whether you are starting out or have been frequently programming the ESP32, this ESP32 Guide will be an important resource for how to setup Arduino IDE and program your ESP32 board.

ESP32 Overview

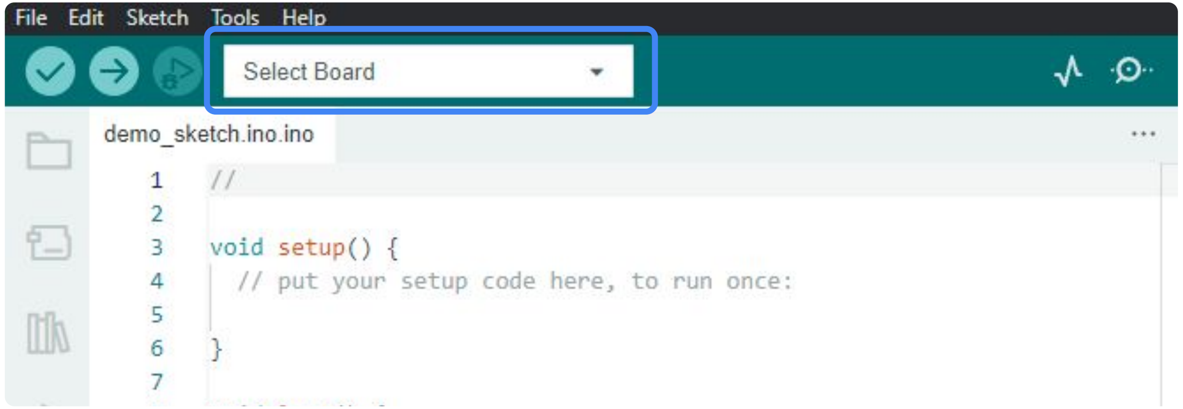
The ESP32-C3Fx4 is a board in the OPS project kits that is based on an ESP32 microcontroller. The ESP32 board has features such as Wi-Fi and Bluetooth capabilities that allow us to build and control cool projects with it!

In OPS, **we will now refer to the ESP32-C3Fx4 board as an ESP32** for convenience. The ESP32 board is depicted here (without its pin headers).

General Purpose Input/Output (GPIO), power, and ground pins run along the sides of the ESP32.

To program the ESP32, we will use the Arduino programming language (a C++ variant) to write and compile sketches. When you need to read up on Arduino code, refer to the Arduino Language Reference.

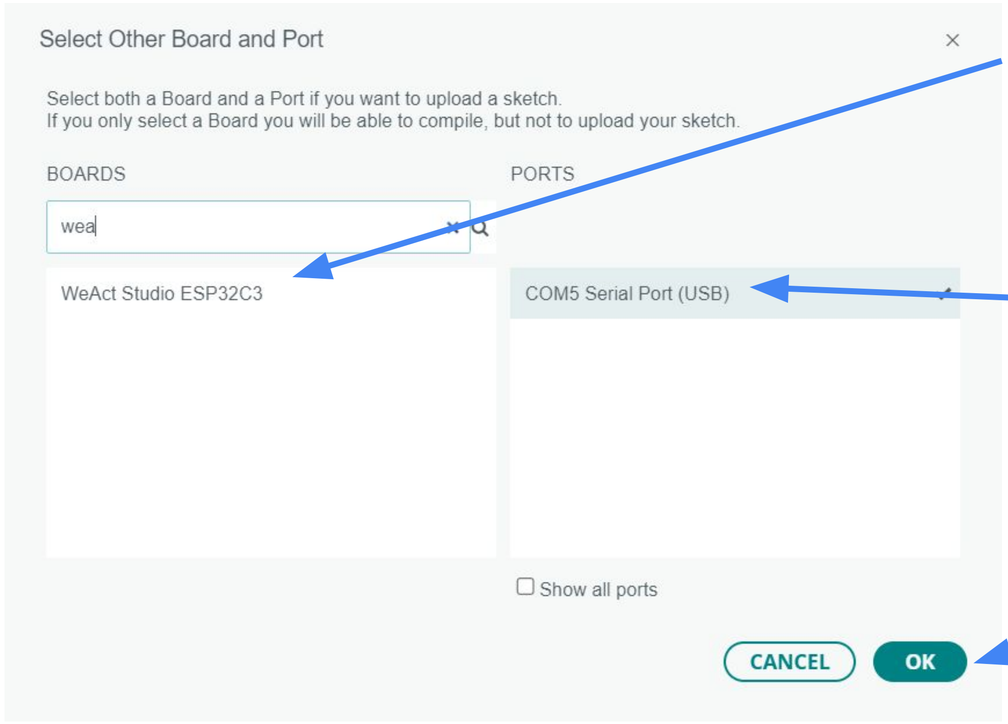
Setting the Target Board



```
//

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

}
```

- Before we can upload a sketch, we must **connect the ESP32 board via the USB-C to USB-A cable to the computer** and configure the IDE to the correct board and USB port
- Open the **Select Board** dropdown and select an available ESP32 board
    - If no option appears, click **Select other board and port…**

Setting the Target Board (Cont'd)


Select Other Board and Port

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

BOARDS

wea

WeAct Studio ESP32C3

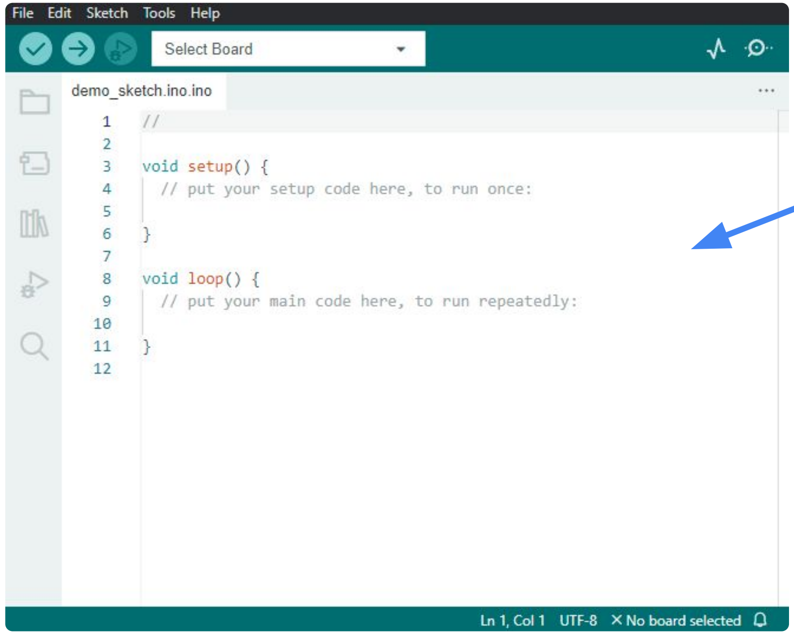
PORTS

COM5 Serial Port (USB)

Show all ports

- In the popup window, search **Boards** and select **WeAct Studio ESP32C3**
- From the available items under **Port**, select the USB port to which the board is currently connected
- Confirm the selections by clicking **OK**

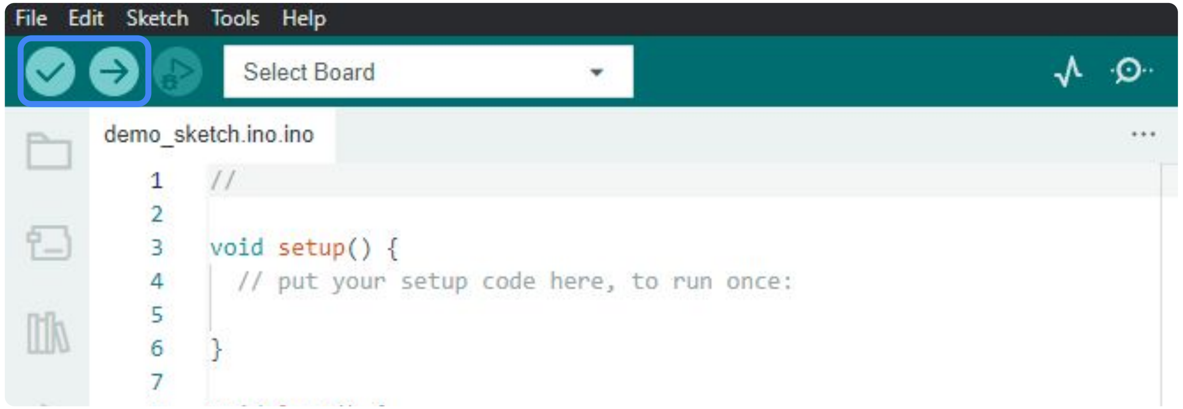
Editing a Sketch


File Edit Sketch Tools Help
Select Board
demo_sketch.ino.ino
Ln 1, Col 1 UTF-8 × No board selected

```
//

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

}

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

}
```

- Sketches can be modified from the **code editor**, which appears just below the menu
- New sketches opened in the editor come with **template code**

Verifying and Uploading a Sketch



```
//

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

}

```

- Before uploading the sketch, select **Verify** (the **checkmark** icon) to compile the sketch
- Once the sketch compiles successfully, select **Upload** (the **right arrow** icon) to upload the compiled sketch to the Arduino board

SECTION II

Review: ESP32 Boot Mode (For Programming)

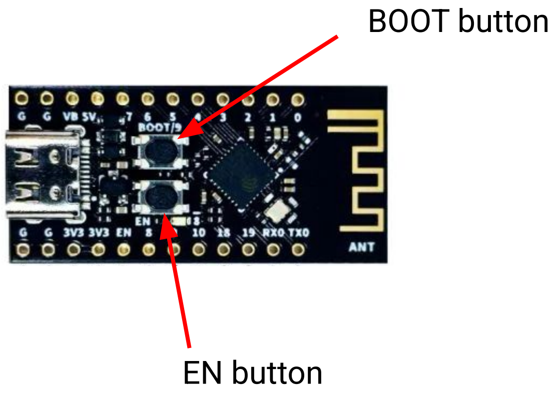
ESP32 Boot Mode

- You will need to enter Boot Mode on the ESP32 whenever you upload code to the ESP32 through Arduino IDE

How to Enter ESP32 Boot Mode:

1. Hold down the **BOOT button** for 2 secs, then hold down the **EN button** as well for 2 secs
2. Let go of the **EN button**, then let go of the **BOOT button**
3. Your board should show up again in the **"Select Board" toolbar**

**BOOT and EN buttons are labeled on the ESP32*



ESP32 Boot Mode Example

OPS

ESP32 Setup and Programming Tutorial

SECTION III

Review: ESP32 Functions

Digital Pin Functions

- `digitalWrite(int pin, int value)`
  - **Sets the voltage** at the output pin to either a `HIGH` (3.3V) or `LOW` (0V) value
  - Analogy - light switch and light bulb:
    - Like toggling a switch on and off
- `digitalRead(int pin)`
  - **Reads the voltage** at the input pin, returning `HIGH` (3.3V) or `LOW` (0V) as an integer (1 or 0)

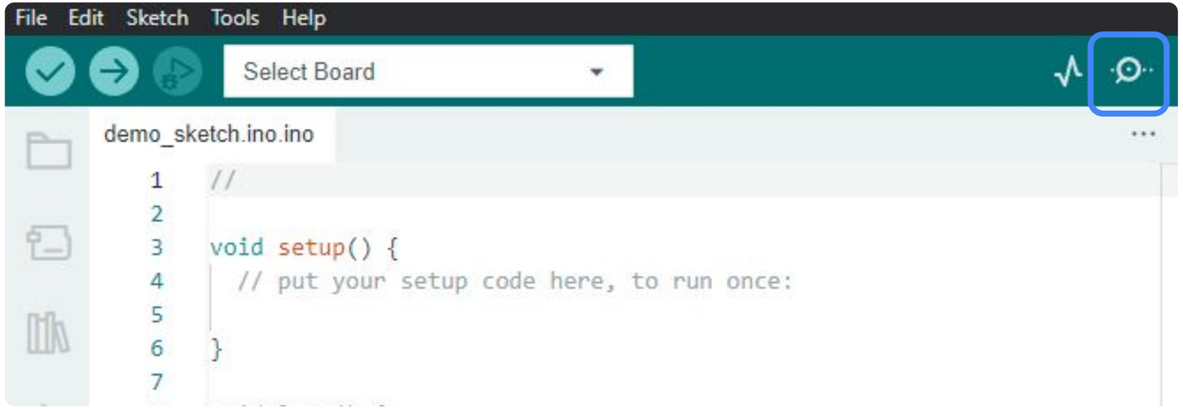
Analog Pin Functions

- `analogWrite(int pin, int value)`
  - **Sets the average voltage** on digital output pin to a value in the **range** `0-255` (0V to 3.3V)
  - Analogy - light dimmer:
    - You use the slide to set the bulb to anywhere *between* MAX brightness or MIN brightness
- `analogRead(int pin)`
  - **Reads the voltage** at the input pin, maps it to a value in the **range** `0-4095` (0V to 3.3V) and returns that value
  - Use the aliases `A0`, `A1`, `A2`… for the pin number

More Basic Functions

- `delay(int ms)`
  - **Pauses the program execution** by **`ms`** milliseconds
- `Serial.print(“Message”)`
  - Sends a string to the computer connected via USB and **displays the string on the Serial Monitor** in the IDE
- `Serial.println(“Message”)`
  - Sends a string to the computer connected via USB and **displays your string on the Serial Monitor** in the IDE, **followed by a newline**

Debugging w/ the Serial Monitor (Cont’d)



```
//

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

}

```

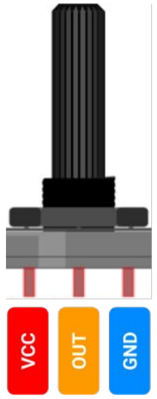
- In the absence of a debugger (the ESP32 is not capable of using one), `Serial.print` is an excellent tool to **help debug programs**
  - Print values to track across parts of your program
    - Unexpected values displayed to the Serial Monitor indicates an error

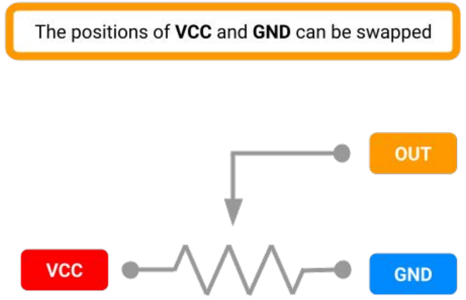
SECTION IV

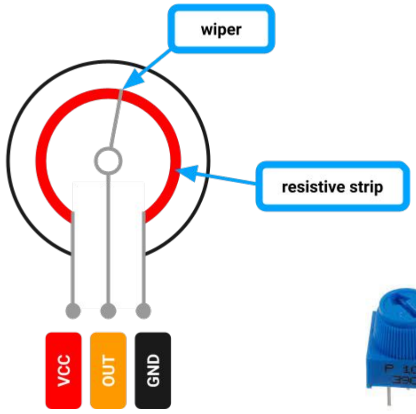
ESP32 Serial Monitor + Potentiometer Exercise

What is a Potentiometer?

- A potentiometer is a *variable* resistor with 3 terminals: VCC, OUT, and GND
- We will use it as a voltage divider to only output a fraction of the supply voltage
    - This output pin voltage varies between the VCC and GND pin voltages based on the dial’s position
- **Disclaimer: Don’t turn the wiper too far past its limit (the knob is fragile and can break easily if turned too far)**

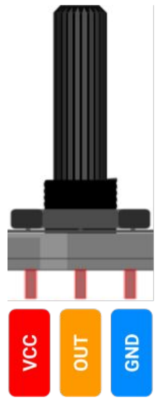


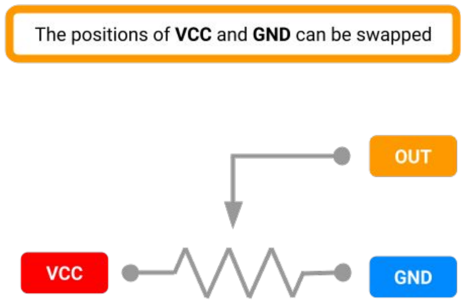


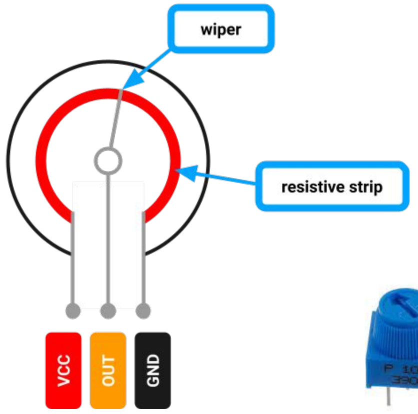


What is a Potentiometer? (cont.)

- A potentiometer has many different applications, such as:
  - Volume control
  - Light dimming
  - Tuning and calibration
- **Trivia questions!**
  - If the wiper is all the way to the left, what is the voltage at the OUT pin?
  - If the wiper is all the way to the left, what would the value in Arduino IDE be between 0-4095?

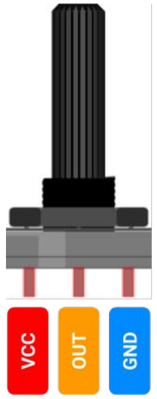


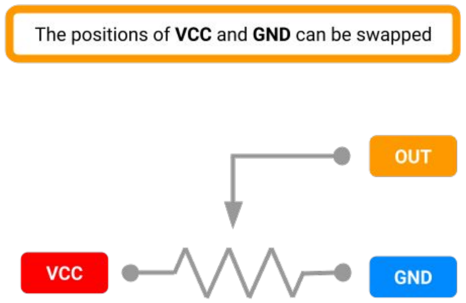


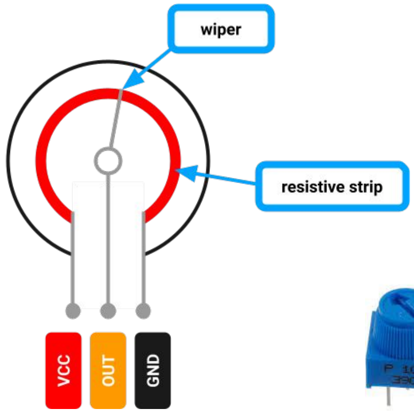


What is a Potentiometer? (cont.)

- A potentiometer has many different applications, such as:
  - Volume control
  - Light dimming
  - Tuning and calibration
- **Trivia questions!**
  - If the wiper is all the way to the left, what is the voltage at the OUT pin?
  - Answer: **3.3V**
  - If the wiper is all the way to the left, what would the value in Arduino IDE be between 0-4095?
  - Answer: **4095**

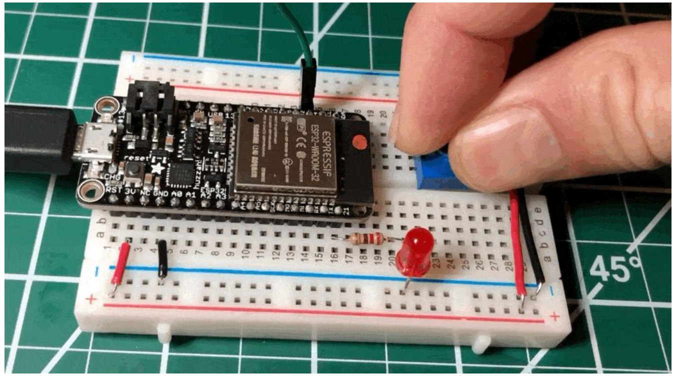






Using Potentiometers with ESP32

- We can’t just use digitalRead() to read values in between 0 - 3.3V off our potentiometer
- Instead, we’ll be using `analogRead(int pin)`
    - The analog pin is wired to the ESP32’s **analog-to-digital converter (ADC)**
        - Translates the analog signal to a discrete digital signal
- Now, let’s look at some code showing this in action!

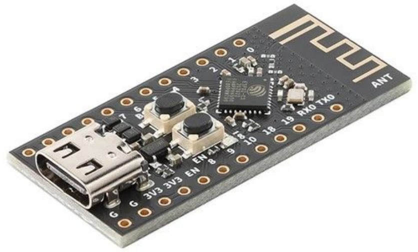
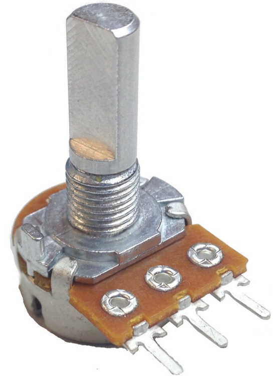
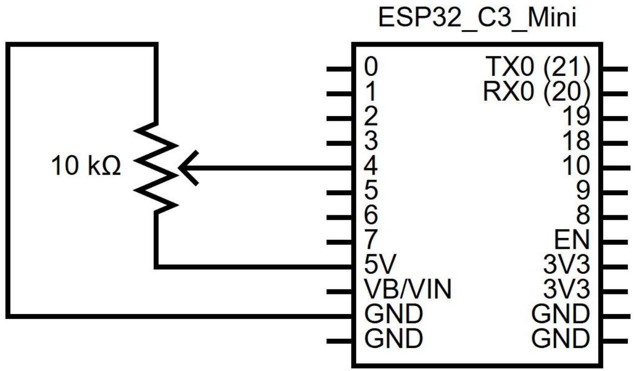


LED Potentiometer Exercise

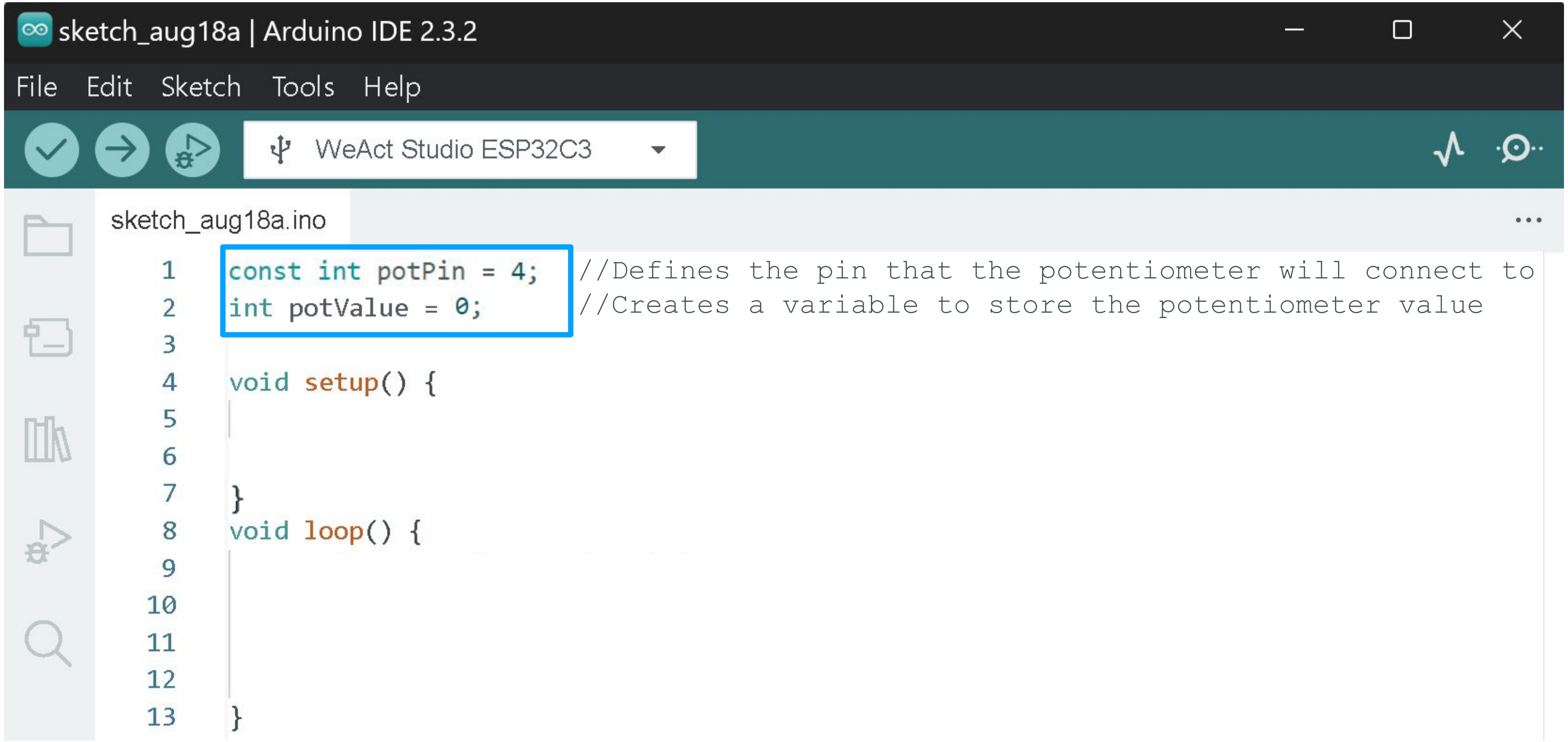
- Now, we’re going to learn how to read values from a potentiometer!
- The goal of this exercise is to print the values read from a potentiometer output pin on the serial monitor
- To start: build the schematic you see on the right

You will need:

- x1 ESP32
- x1 Breadboard
- x1 USB-A to USB-C Cable (For Programming)
- x1 Potentiometer

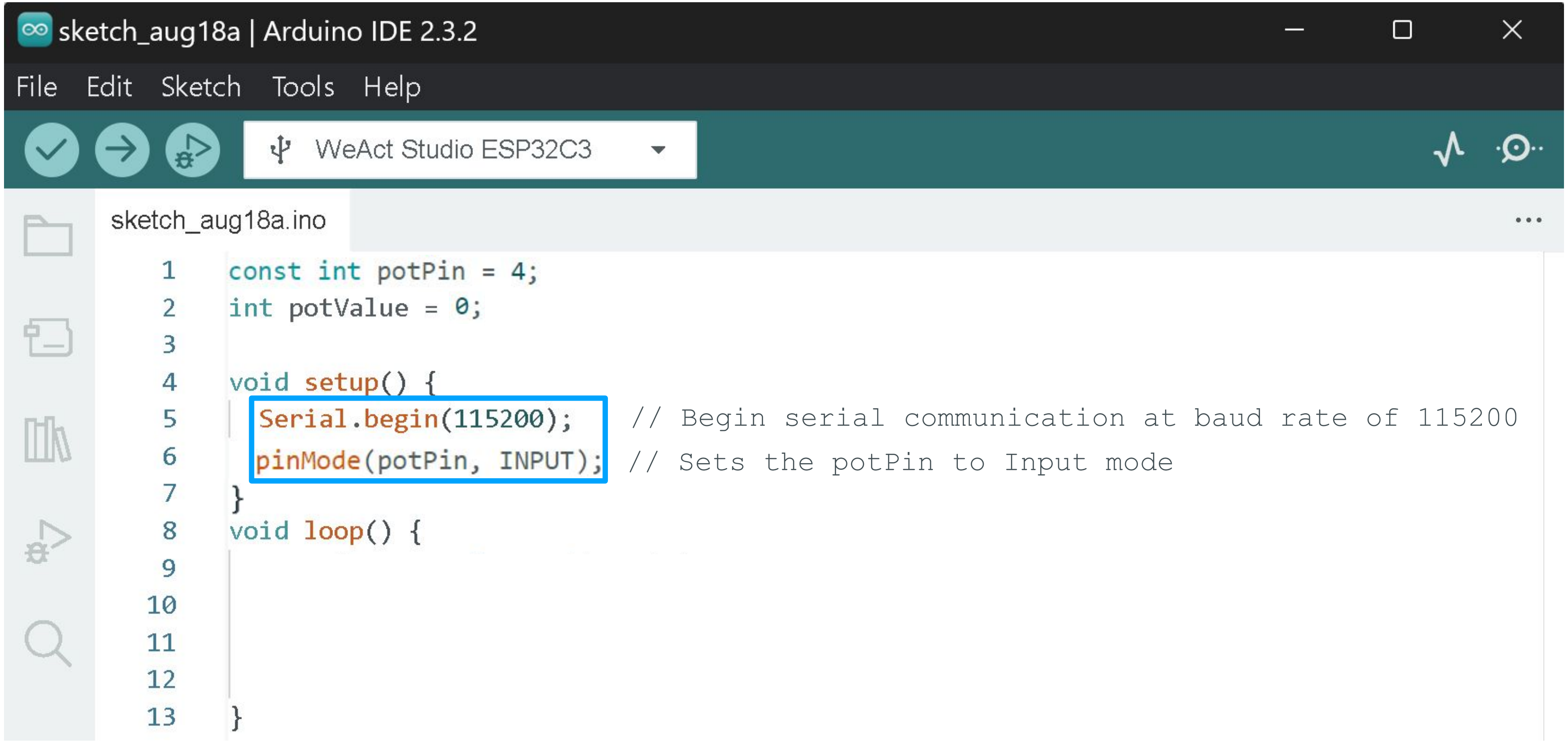




Potentiometer Code Example


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

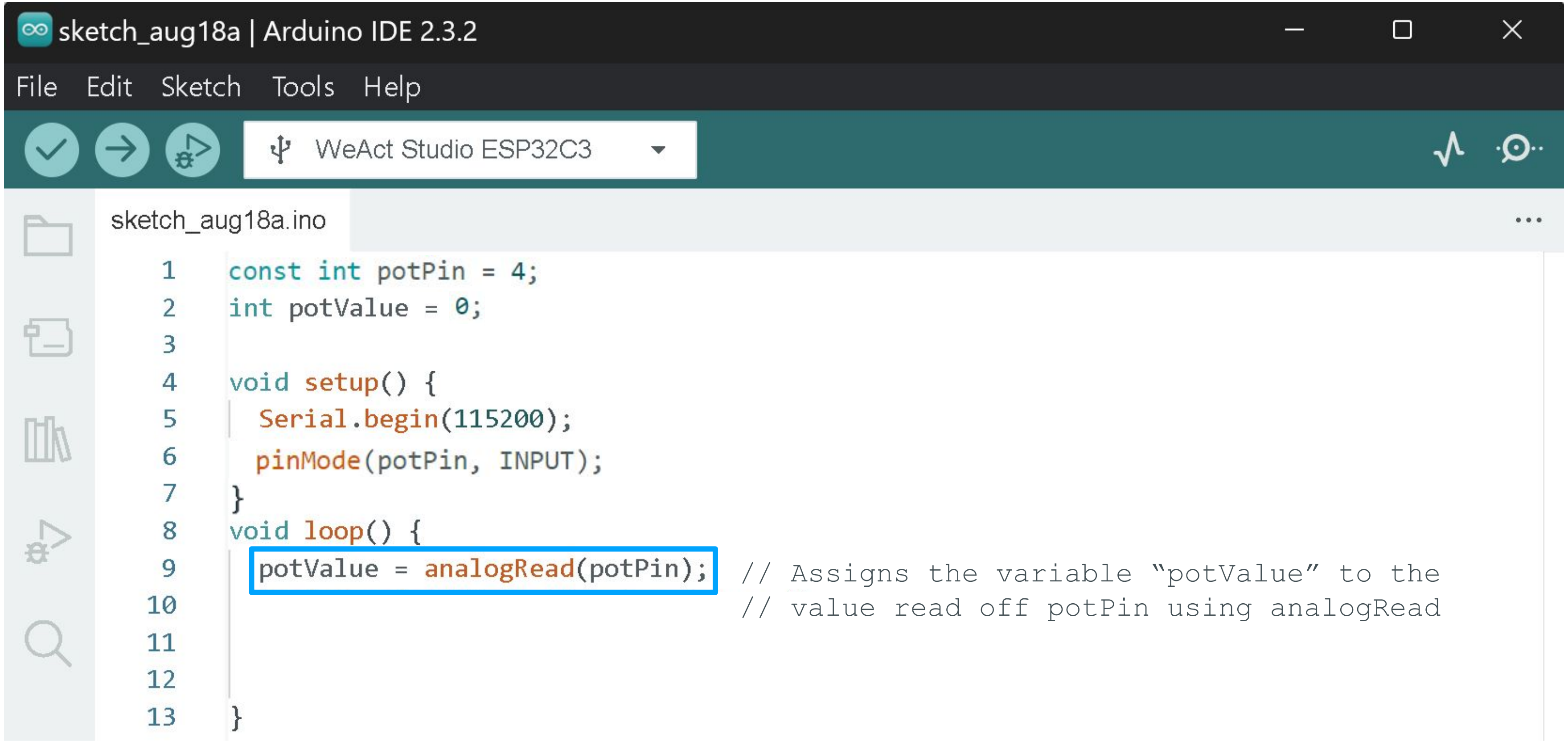
```
const int potPin = 4;  //Defines the pin that the potentiometer will connect to
int potValue = 0;      //Creates a variable to store the potentiometer value

void setup() {



}
void loop() {




}
```

Potentiometer Code Example


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

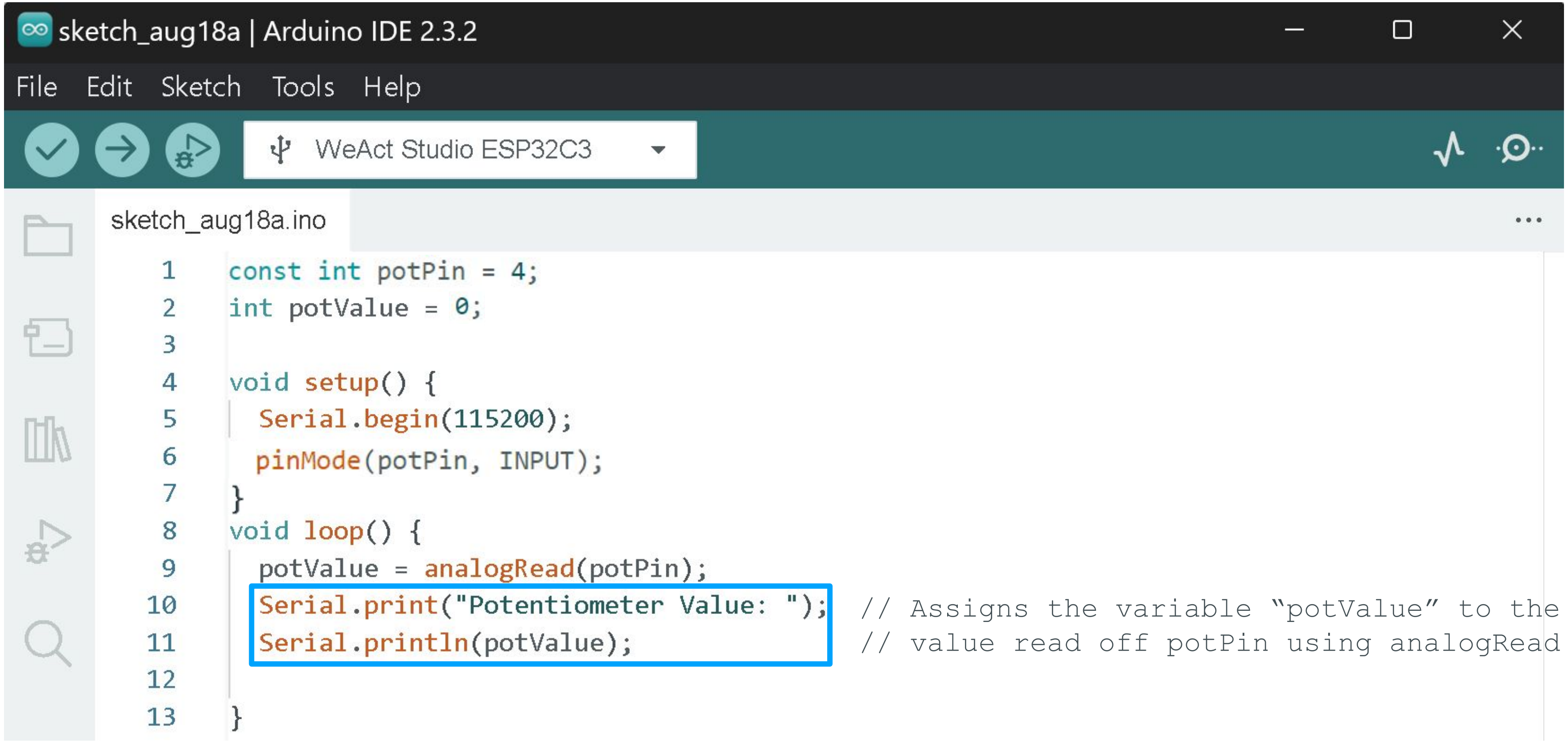
```
const int potPin = 4;
int potValue = 0;

void setup() {
  Serial.begin(115200);    // Begin serial communication at baud rate of 115200
  pinMode(potPin, INPUT);  // Sets the potPin to Input mode
}
void loop() {




}
```

Potentiometer Code Example


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

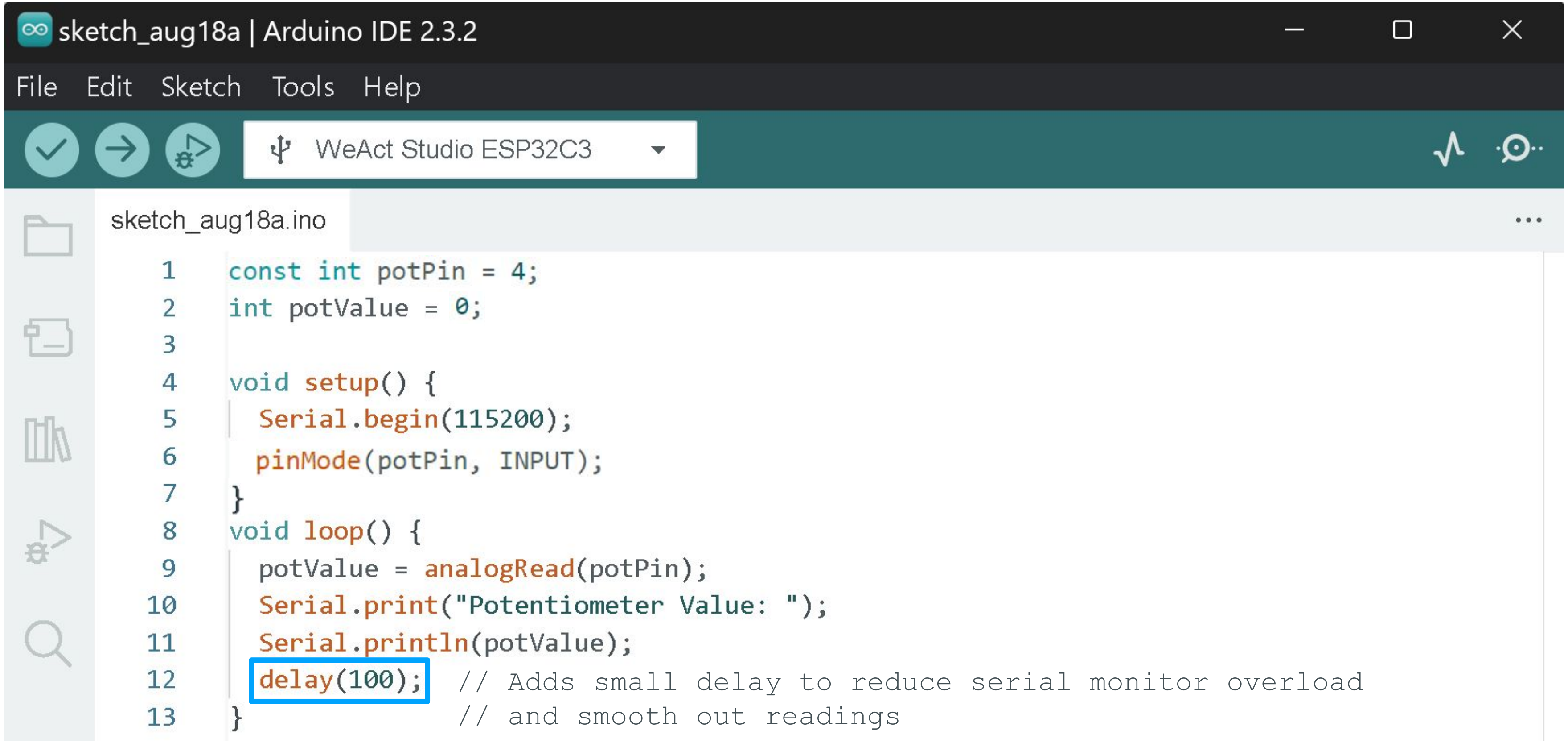
```
const int potPin = 4;
int potValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
}
void loop() {
  potValue = analogRead(potPin);   // Assigns the variable "potValue" to the
                                   // value read off potPin using analogRead



}
```

Potentiometer Code Example



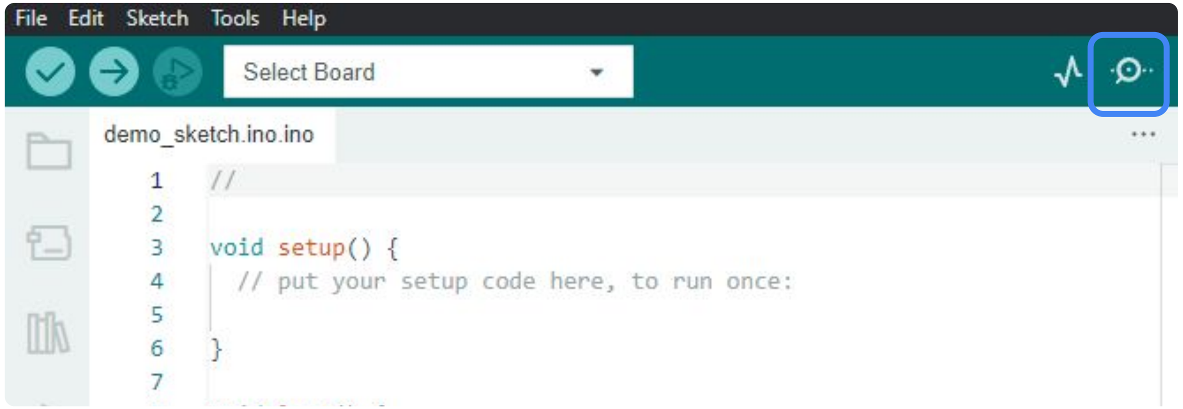
```
const int potPin = 4;
int potValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");   // Assigns the variable "potValue" to the
  Serial.println(potValue);                // value read off potPin using analogRead

}
```

Potentiometer Code Example


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

```
const int potPin = 4;
int potValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);
  delay(100);   // Adds small delay to reduce serial monitor overload
}               // and smooth out readings
```

Using the Serial Monitor



```
//

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

}

```

- While the ESP32 board is connected to the personal computer via USB, select **Serial Monitor** (the **magnifying glass** icon) in the IDE
  - A pane will appear at the bottom of the IDE window which displays all data sent by the ESP32 board using `Serial.print`

SECTION V

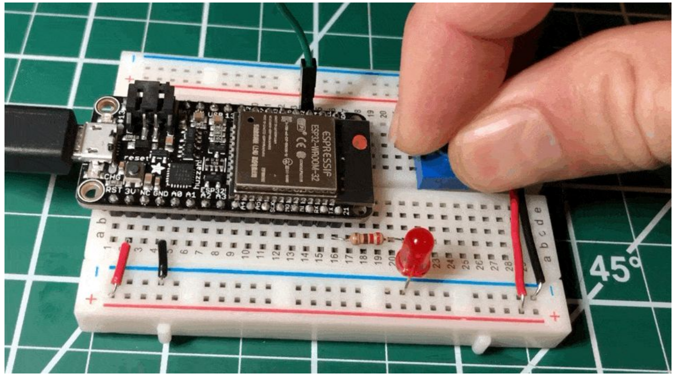
LED Dimmer Exercise

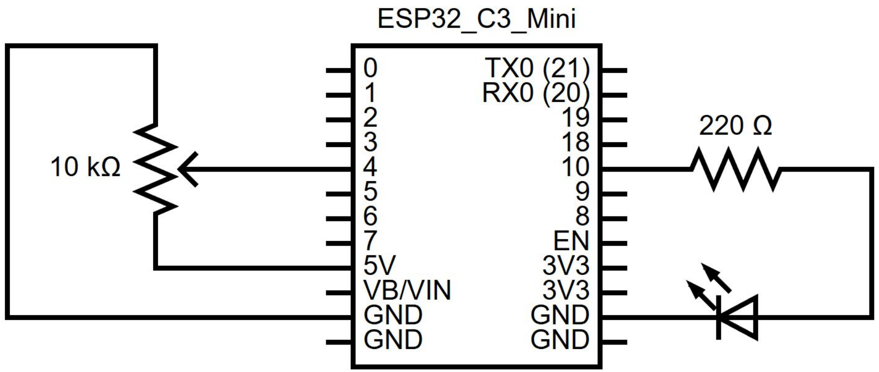
LED Potentiometer Code Exercise

- Now, we're going to control the brightness of an LED with a potentiometer!

- The goal of this exercise is to make the LED grow brighter as you twist the potentiometer **clockwise**

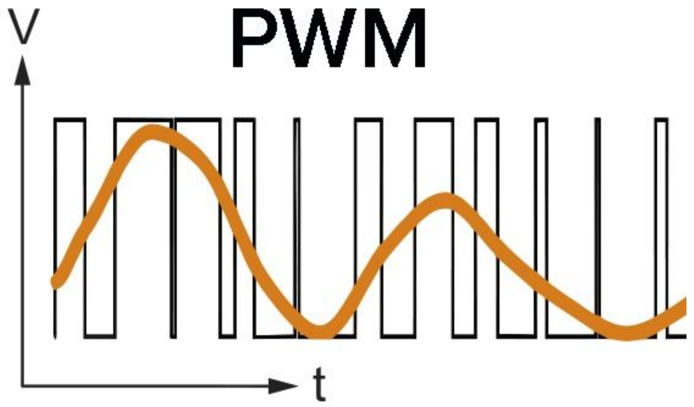
- To start: build the schematic you see on the right





Using Potentiometers with ESP32

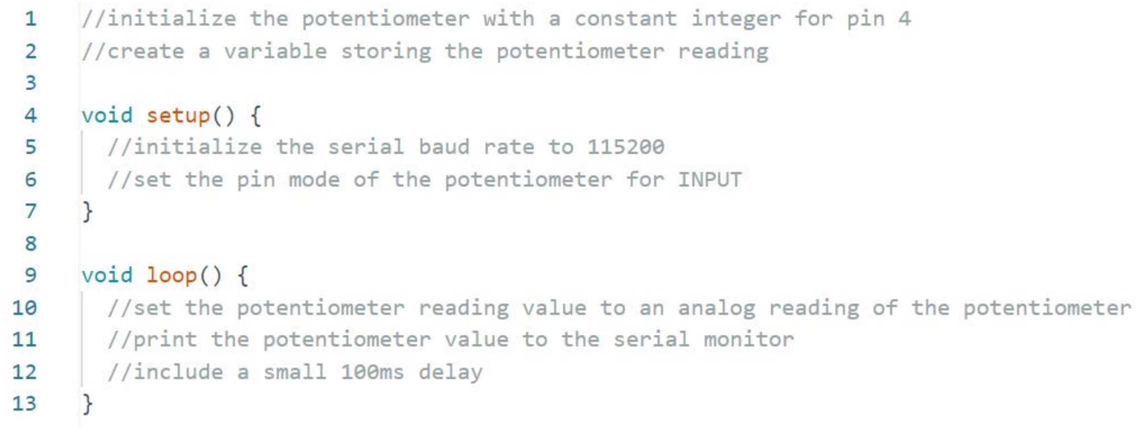
- digitalWrite() can only set a pin's voltage to HIGH or LOW, nothing in between!
- We need to use a new function,
  `analogWrite(int pin, int value)`
  - With **pulse width modulation (PWM) waves**, we can generate an average voltage anywhere between 0V and 3.3V
  - `int pin` - Reference a specific pin to use
  - `int value` - Any value between 0 and 255 (Inputting 0 outputs 0V, and 127 outputs 1.65V, 255 outputs 3.3V, etc.)



LED Potentiometer Code Exercise

- Now that you have built the schematic, it's time to program it!

- Feel free to use this pseudocode outline to guide your programming

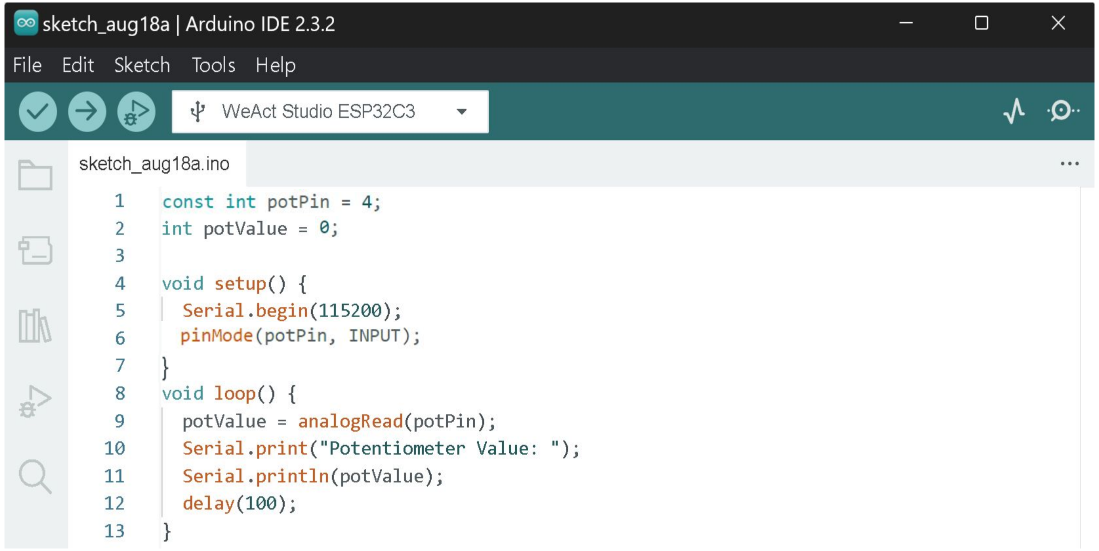


```
//initialize the potentiometer with a constant integer for pin 4
//create a variable storing the potentiometer reading

void setup() {
  //initialize the serial baud rate to 115200
  //set the pin mode of the potentiometer for INPUT
}

void loop() {
  //set the potentiometer reading value to an analog reading of the potentiometer
  //print the potentiometer value to the serial monitor
  //include a small 100ms delay
}
```

LED Potentiometer Code Exercise (cont.)

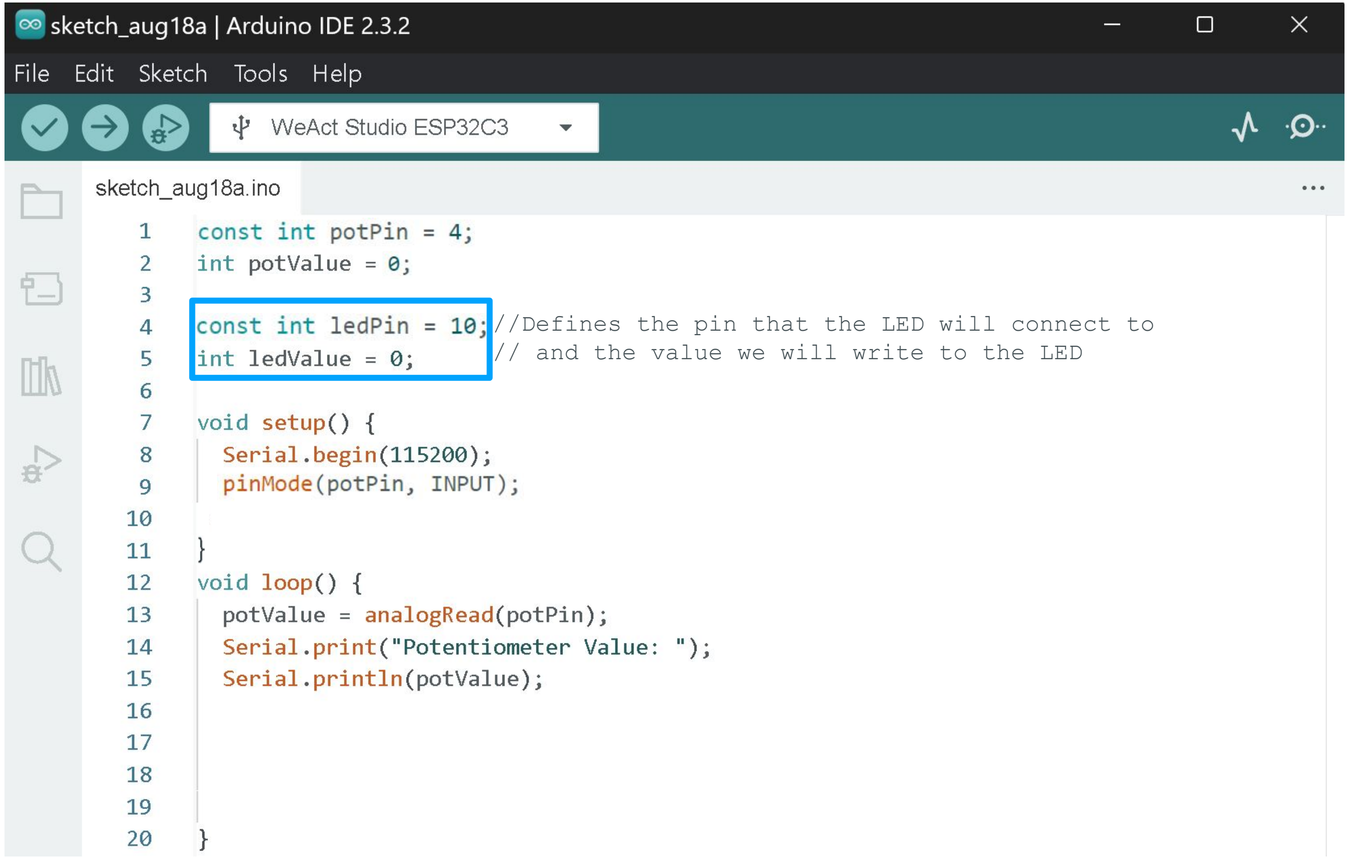
- Now that the schematic is finished, ensure that you have code to read the values of the potentiometer

- Feel free to reference the earlier example code if you need help starting!



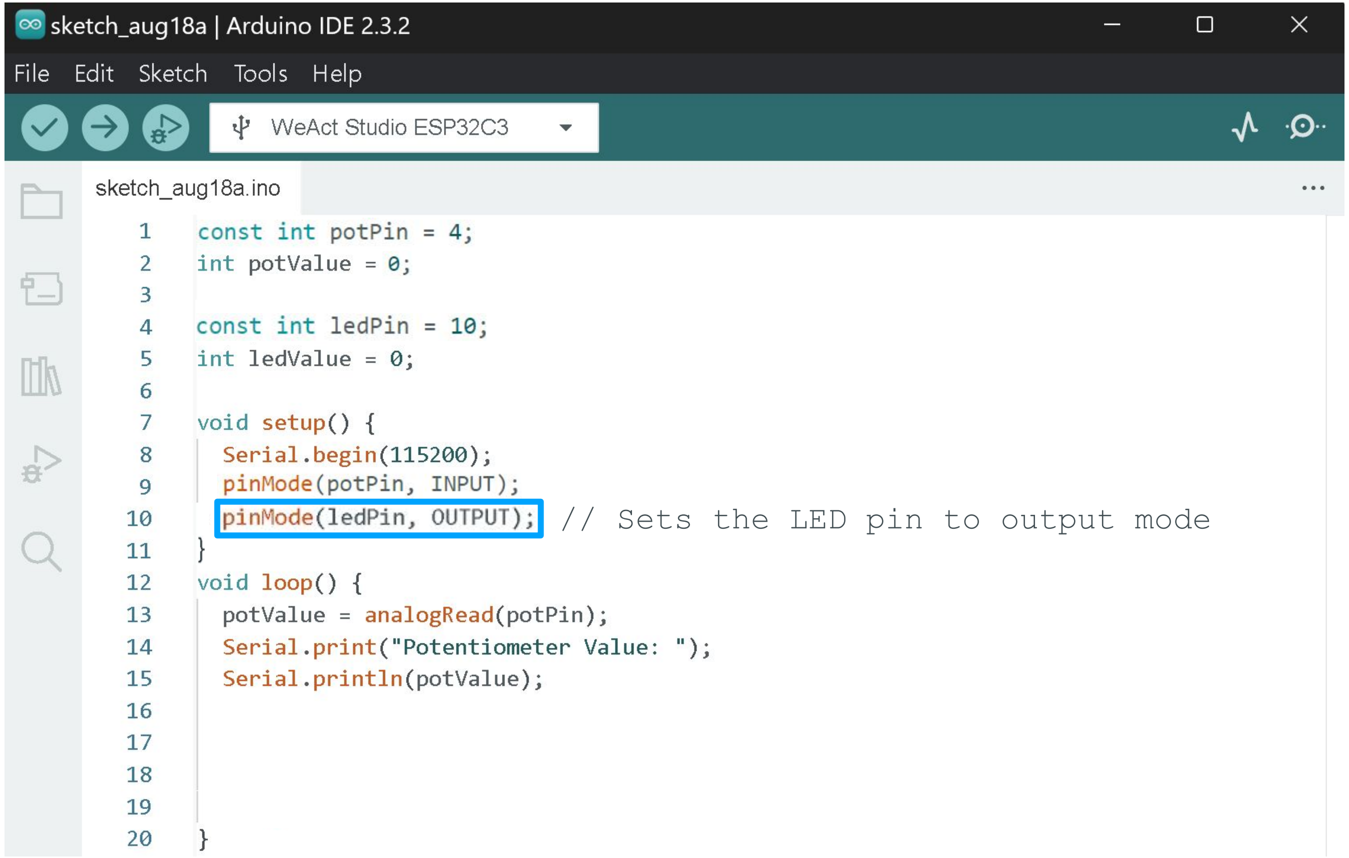
```
const int potPin = 4;
int potValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);
  delay(100);
}
```

LED Potentiometer Code Exercise (cont.)


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

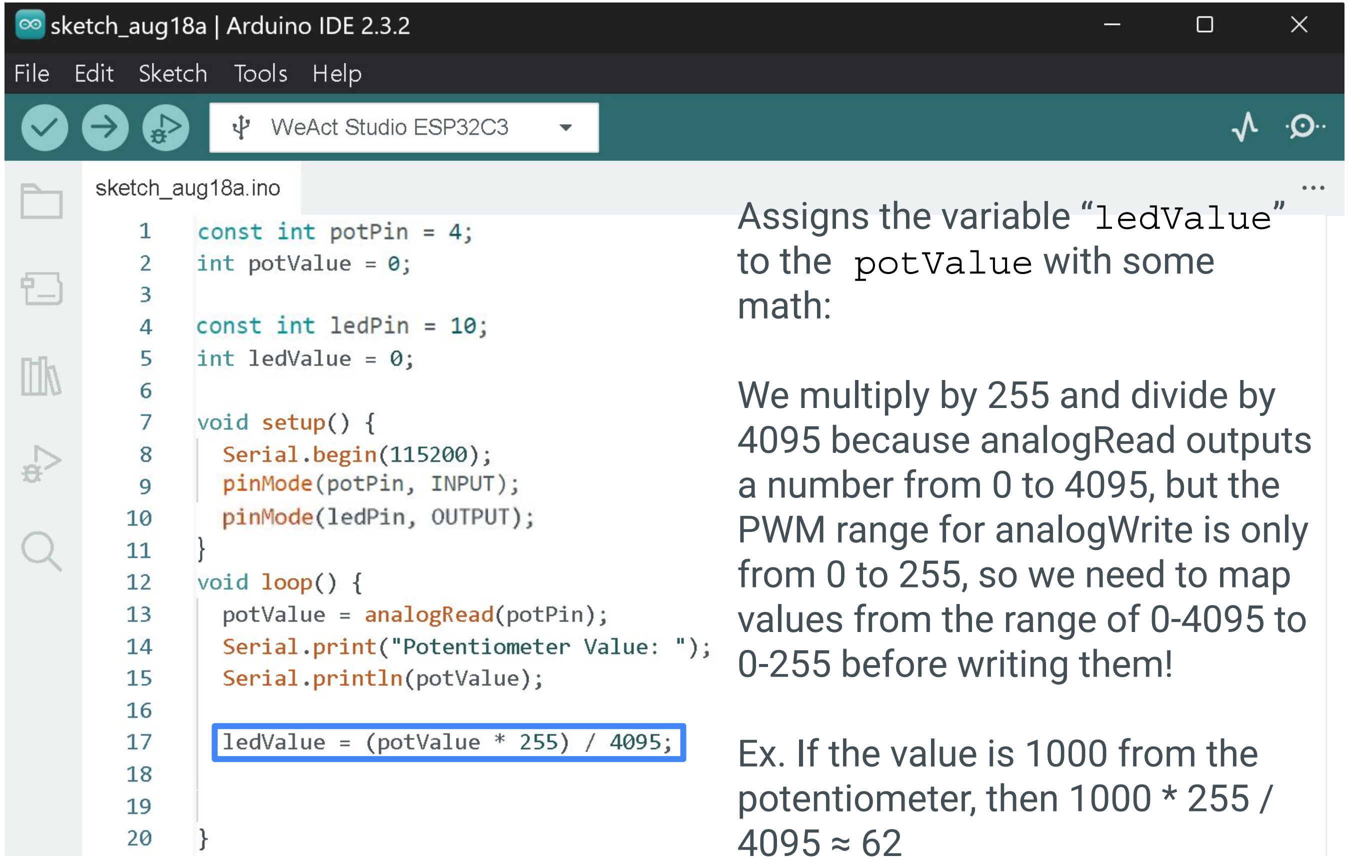
```
const int potPin = 4;
int potValue = 0;

const int ledPin = 10;//Defines the pin that the LED will connect to
int ledValue = 0;     // and the value we will write to the LED

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);

}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);




}
```

LED Potentiometer Code Exercise (cont.)


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

```
const int potPin = 4;
int potValue = 0;

const int ledPin = 10;
int ledValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
  pinMode(ledPin, OUTPUT); // Sets the LED pin to output mode
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);





}
```

LED Potentiometer Code Exercise (cont.)


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

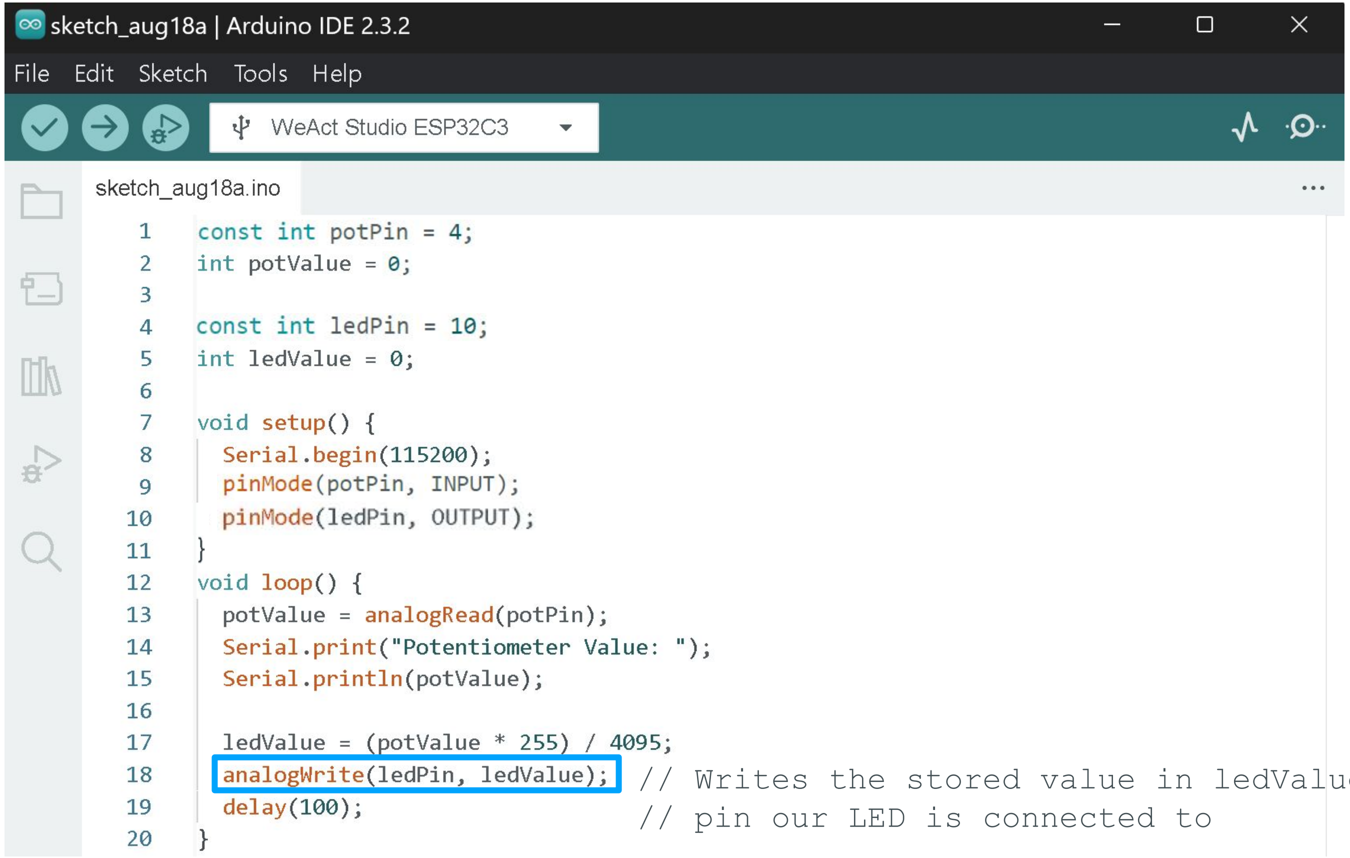
```
const int potPin = 4;
int potValue = 0;

const int ledPin = 10;
int ledValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);

  ledValue = (potValue * 255) / 4095;


}
```

Assigns the variable “`ledValue`” to the `potValue` with some math:

We multiply by 255 and divide by 4095 because analogRead outputs a number from 0 to 4095, but the PWM range for analogWrite is only from 0 to 255, so we need to map values from the range of 0-4095 to 0-255 before writing them!

Ex. If the value is 1000 from the potentiometer, then 1000 * 255 / 4095 ≈ 62

LED Potentiometer Code Example (cont.)


sketch_aug18a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
WeAct Studio ESP32C3
sketch_aug18a.ino

```
const int potPin = 4;
int potValue = 0;

const int ledPin = 10;
int ledValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(potPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  potValue = analogRead(potPin);
  Serial.print("Potentiometer Value: ");
  Serial.println(potValue);

  ledValue = (potValue * 255) / 4095;
  analogWrite(ledPin, ledValue);   // Writes the stored value in ledValue to the
  delay(100);                      // pin our LED is connected to
}
```