

Workshop III

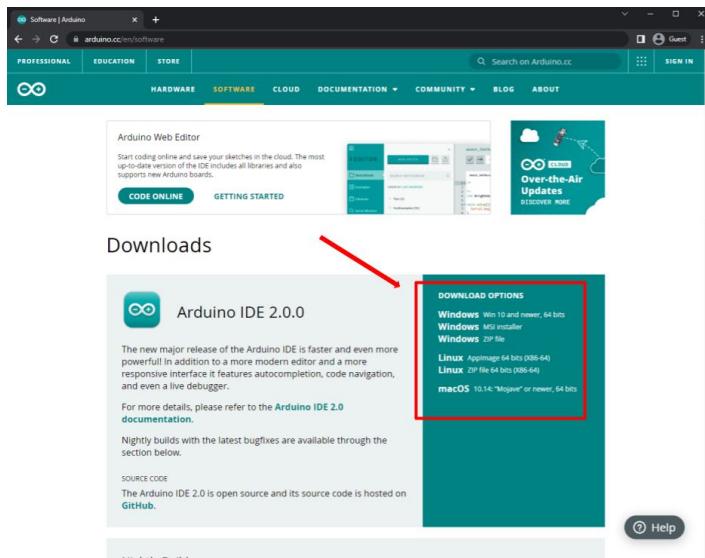
Arduino IDE Setup & Potentiometers

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)



The screenshot shows the OPS website's ESP32 Guide page. The top navigation bar includes Home, About, Projects, Lectures, Workshops, and ESP32 Guide. The main content area has a sidebar with 'Contents' and a 'ESP32 Overview' section highlighted in blue. The main content area contains text about the ESP32 board and its capabilities. A photo of the ESP32-C3Fx4 board is shown on the right. A small box in the bottom right corner indicates the page was updated on November 11, 2024, by Sahil Dhakote & Gavin Nguyen.

Contents

ESP32 Overview

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

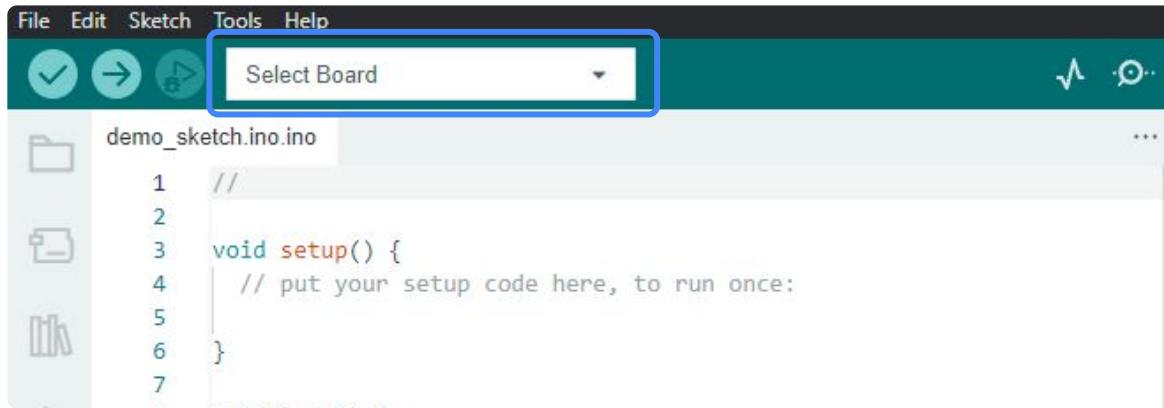
The ESP32-C3Fx4 is a board in the OPS project kits that is based on an ESP32 microcontroller. The ESP32 board has features such as Wi-Fi and Bluetooth capabilities that allow us to build and control cool projects with it!

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.

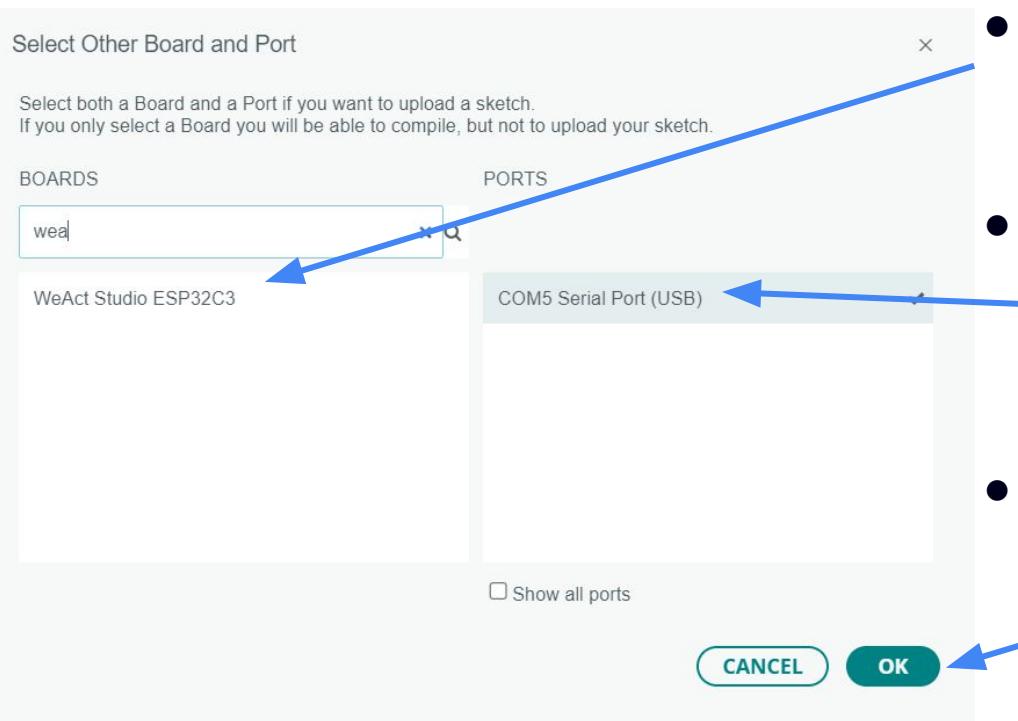
To program the ESP32, we will use the Arduino programming language (a C++ variant) to write and compile sketches. When you need to read up on Arduino code, refer to the [Arduino Language Reference](#).

Setting the Target Board



- 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)



- 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**

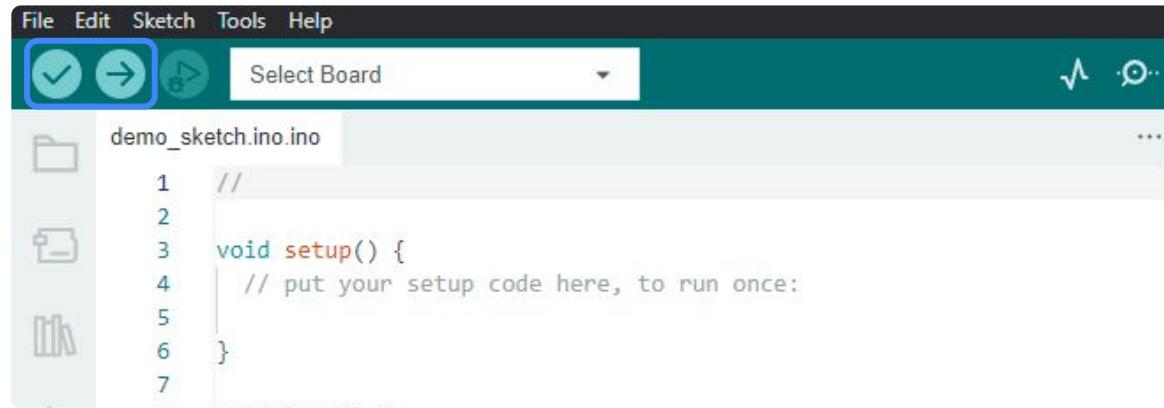
Editing a Sketch

The screenshot shows the Arduino IDE interface. The menu bar includes File, Edit, Sketch, Tools, and Help. A toolbar with various icons is visible above the code area. The code editor window has a title bar "Select Board" and a status bar at the bottom indicating "Ln 1, Col 1" and "UTF-8". The code itself is a template:

```
//  
//  
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



- 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)

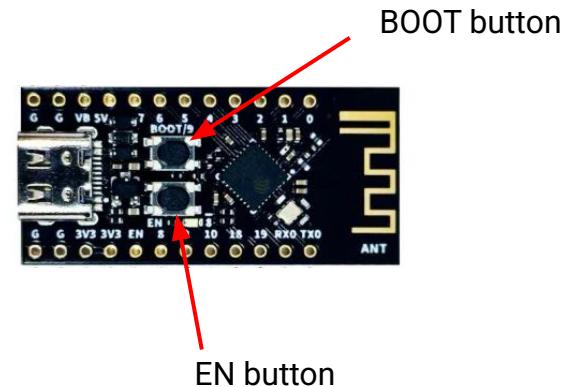
ESP32 Boot Mode

- 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



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

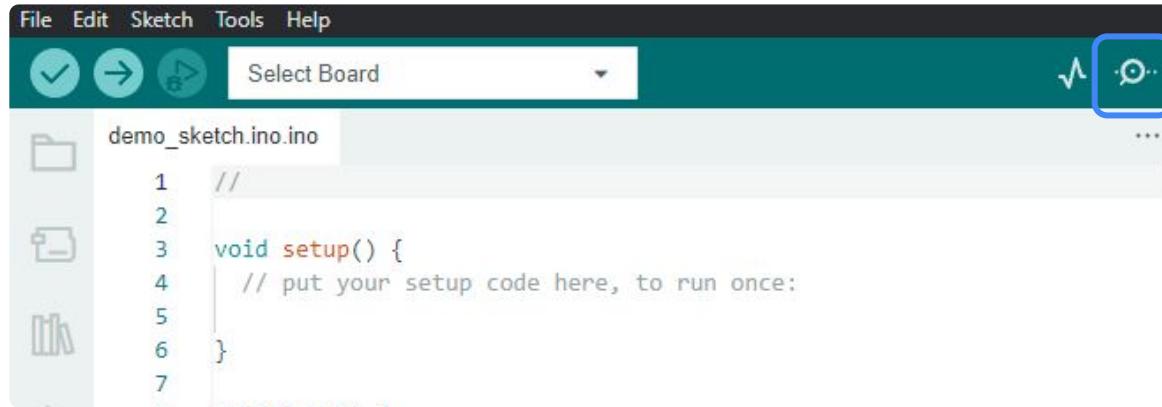


- **analogWrite(int pin, int value)**
 - Sets the average voltage on digital output pin to a value in the range 0–255 (0V to 3.3V)
 - Analogy - light dimmer:
 - You use the slide to set the bulb to anywhere between MAX brightness or MIN brightness
- **analogRead(int pin)**
 - Reads the voltage at the 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)



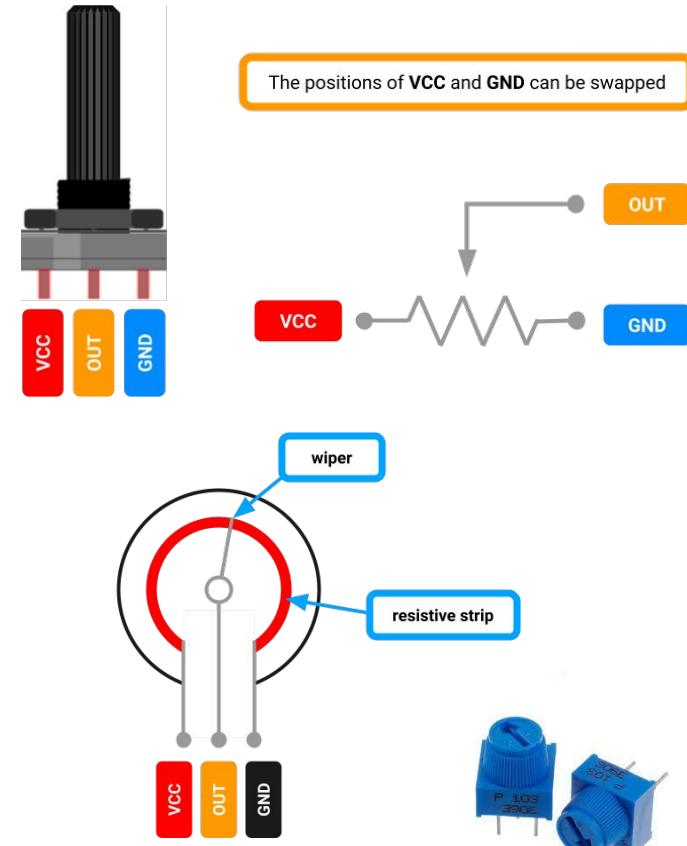
- In the absence of a debugger (the ESP32 is not capable of using one), **Serial.print** is an excellent tool to **help debug programs**
 - Print values to track across parts of your program
 - Unexpected values displayed to the Serial Monitor indicates an error

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)**

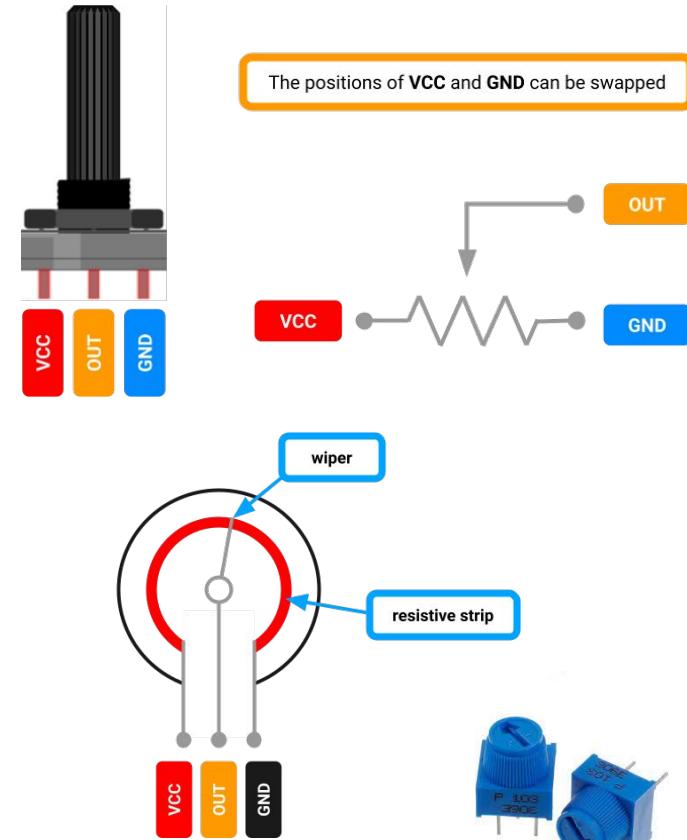


What is a Potentiometer? (cont.)

- A potentiometer has many different applications, such as:
 - Volume control
 - Light dimming
 - Tuning and calibration

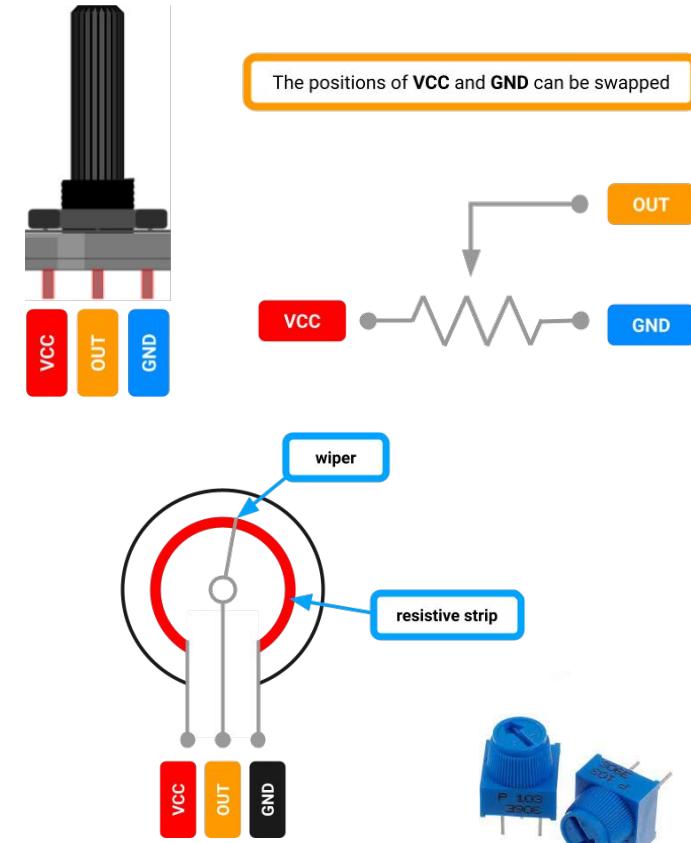
• Trivia questions!

- 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?



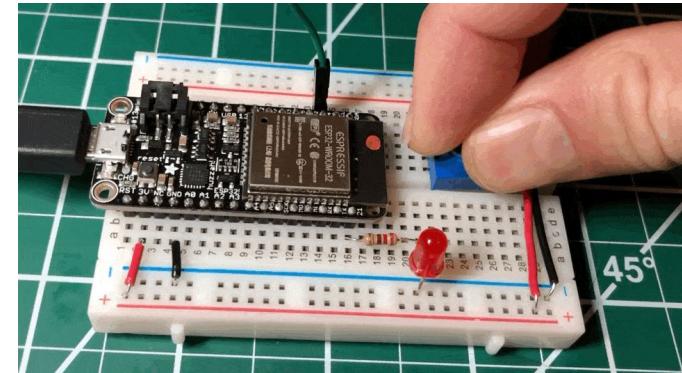
What is a Potentiometer? (cont.)

- 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!

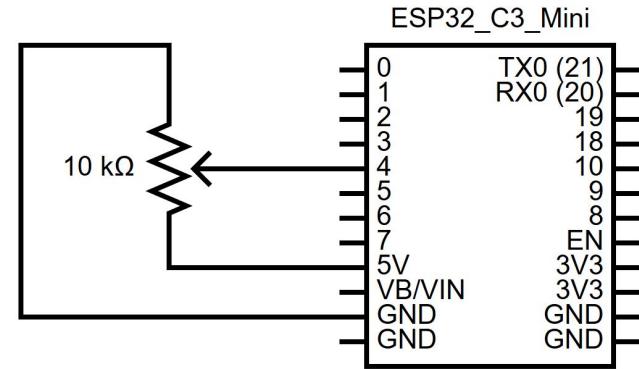


LED Potentiometer Exercise

- 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

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, and Preferences, followed by a dropdown showing "WeAct Studio ESP32C3".
- Code Editor:** Displays the file "sketch_aug18a.ino" with the following code:

```
1 const int potPin = 4; //Defines the pin that the potentiometer will connect to
2 int potValue = 0; //Creates a variable to store the potentiometer value
3
4 void setup() {
5
6
7 }
8 void loop() {
9
10
11
12
13 }
```

The first two lines of code are highlighted with a blue box.
- Sidebar:** On the left, there are icons for File, Sketch, Libraries, and Search.
- Right Panel:** Shows three small circular icons: a checkmark, a right arrow, and a gear.

Potentiometer Code Example

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown for "WeAct Studio ESP32C3".
- Sketch Name:** sketch_aug18a.ino
- Code Area:** Displays the following C++ code for an Arduino sketch.

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 void setup() {
5     Serial.begin(115200); // Begin serial communication at baud rate of 115200
6     pinMode(potPin, INPUT); // Sets the potPin to Input mode
7 }
8 void loop() {
9
10
11
12
13 }
```

The code initializes pin 4 as an input for a potentiometer and sets up serial communication at 115200 baud. The `Serial.begin(115200);` and `pinMode(potPin, INPUT);` lines are highlighted with a blue box.

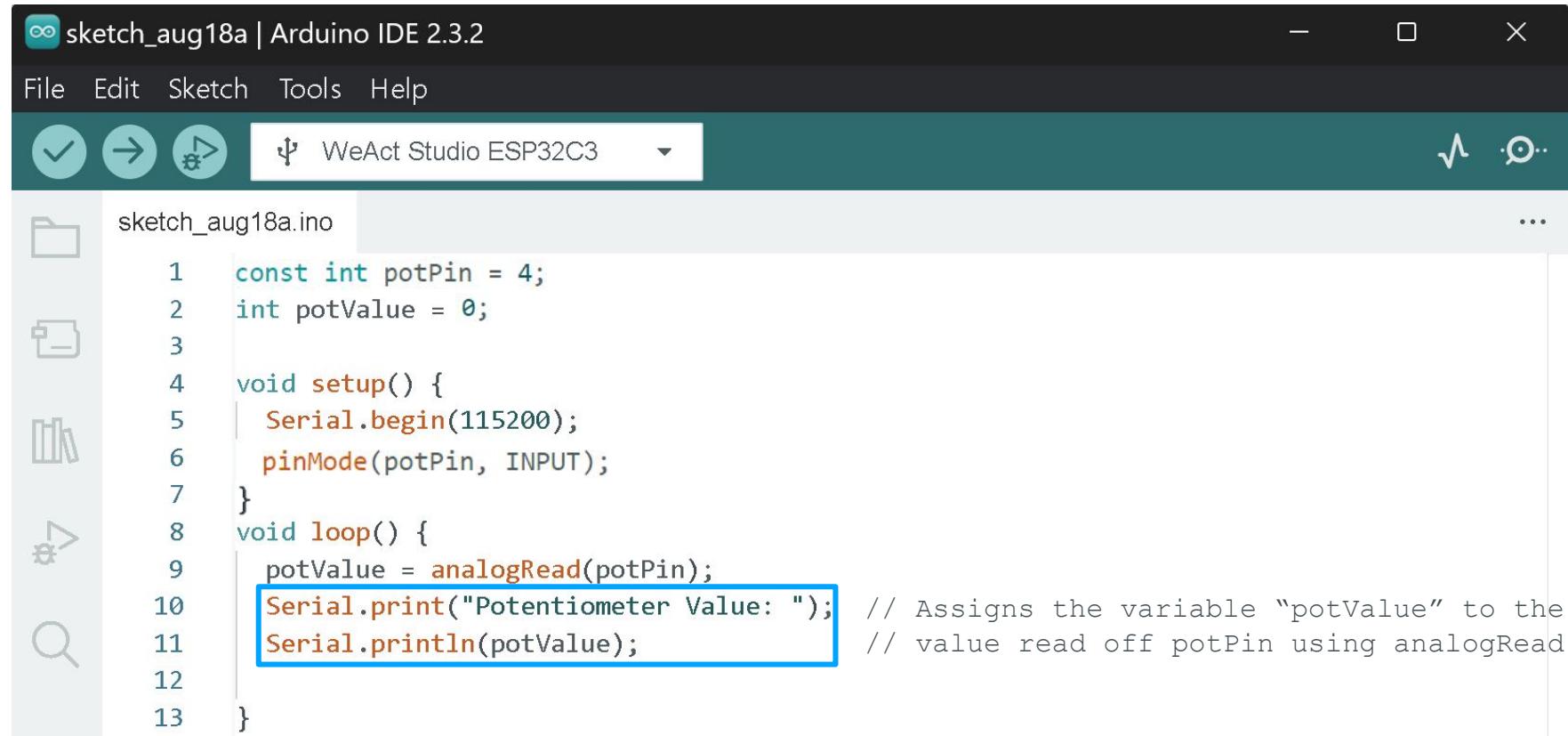
Potentiometer Code Example

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown currently set to "WeAct Studio ESP32C3".
- Sketch Name:** sketch_aug18a.ino
- Code Area:** Displays the following C++ code for reading a potentiometer value.

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 void setup() {
5     Serial.begin(115200);
6     pinMode(potPin, INPUT);
7 }
8 void loop() {
9     potValue = analogRead(potPin); // Assigns the variable "potValue" to the
10                                // value read off potPin using analogRead
11
12
13 }
```

Potentiometer Code Example



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown for "WeAct Studio ESP32C3".
- Sketch Area:** Displays the code for "sketch_aug18a.ino".
- Left Sidebar:** Shows icons for File, Sketch, Board, and Search.
- Code Content:**

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 void setup() {
5     Serial.begin(115200);
6     pinMode(potPin, INPUT);
7 }
8 void loop() {
9     potValue = analogRead(potPin);
10    Serial.print("Potentiometer Value: ");
11    Serial.println(potValue);
12
13 }
```

The lines from 10 to 13 are highlighted with a blue border.

Annotations: A note at the bottom right of the highlighted code block reads: // Assigns the variable "potValue" to the // value read off potPin using analogRead

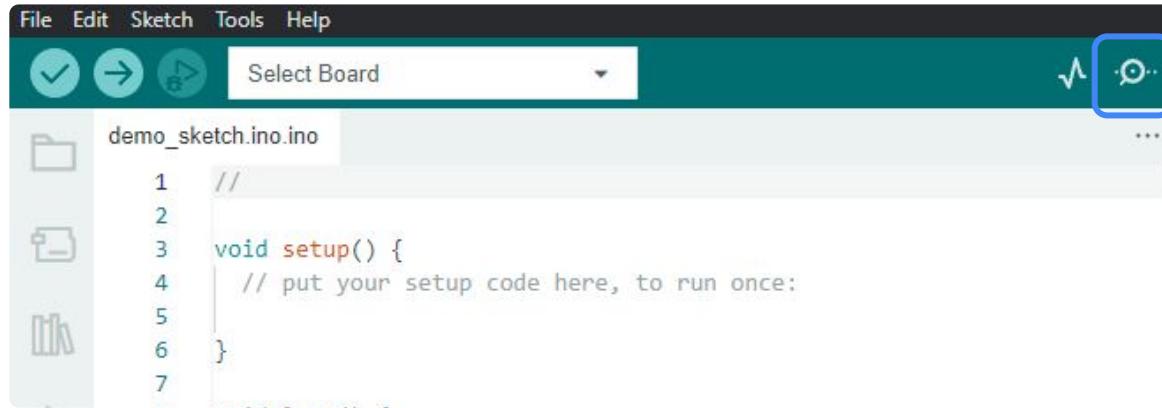
Potentiometer Code Example

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown currently set to "WeAct Studio ESP32C3".
- Sketch Name:** sketch_aug18a.ino
- Code Area:** Displays the following C++ code for reading a potentiometer value and printing it to the Serial Monitor.

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 void setup() {
5     Serial.begin(115200);
6     pinMode(potPin, INPUT);
7 }
8 void loop() {
9     potValue = analogRead(potPin);
10    Serial.print("Potentiometer Value: ");
11    Serial.println(potValue);
12    delay(100); // Adds small delay to reduce serial monitor overload
13 } // and smooth out readings
```

Using the Serial Monitor



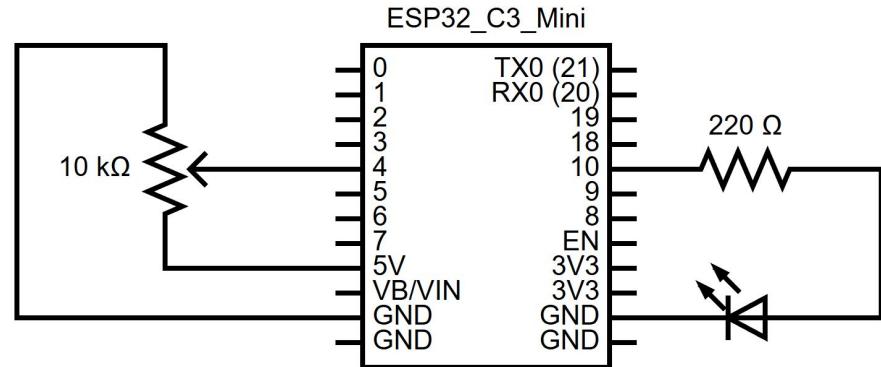
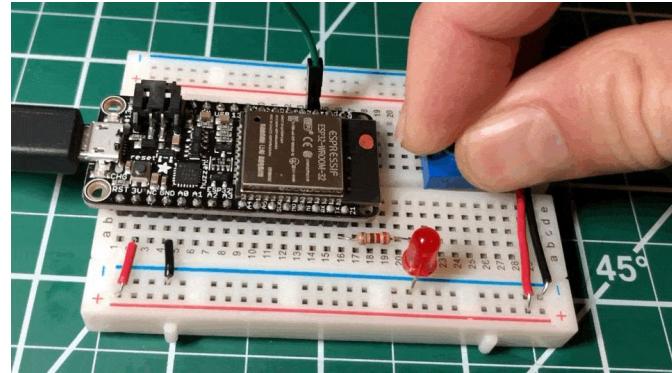
- While the ESP32 board is connected to the personal computer via USB, select **Serial Monitor** (the **magnifying glass** icon) in the IDE
 - A pane will appear at the bottom of the IDE window which displays all data sent by the ESP32 board using **Serial.print**

SECTION V

LED Dimmer Exercise

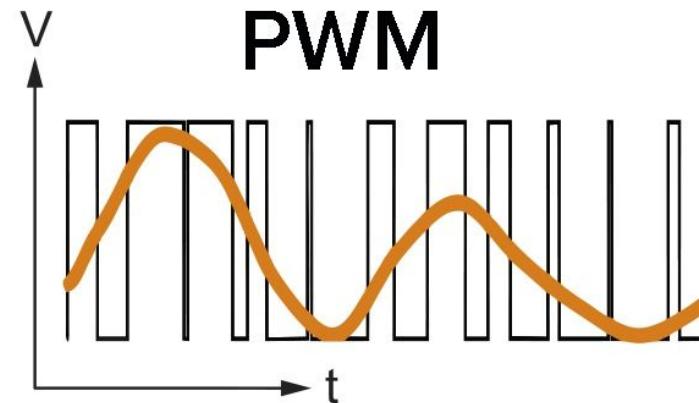
LED Potentiometer Code Exercise

- 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.)



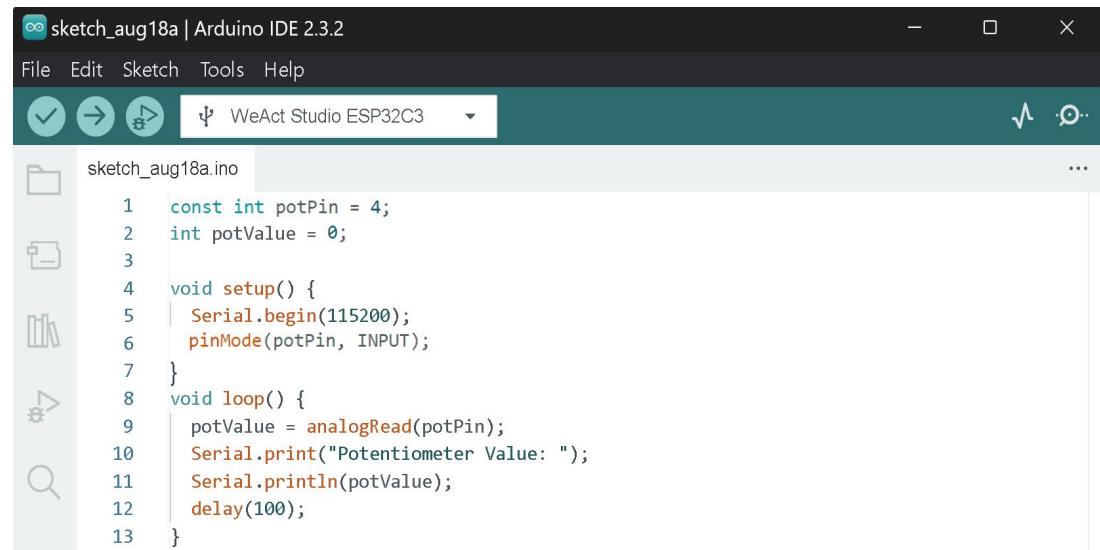
LED Potentiometer Code Exercise

- Now that you have built the schematic, it's time to program it!
- Feel free to use this pseudocode outline to guide your programming

```
1 //initialize the potentiometer with a constant integer for pin 4
2 //create a variable storing the potentiometer reading
3
4 void setup() {
5     //initialize the serial baud rate to 115200
6     //set the pin mode of the potentiometer for INPUT
7 }
8
9 void loop() {
10    //set the potentiometer reading value to an analog reading of the potentiometer
11    //print the potentiometer value to the serial monitor
12    //include a small 100ms delay
13 }
```

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!



The screenshot shows the Arduino IDE interface with the title bar "sketch_aug18a | Arduino IDE 2.3.2". The toolbar includes standard icons for file operations and a dropdown for the board: "WeAct Studio ESP32C3". The code editor displays the following sketch:

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 void setup() {
5     Serial.begin(115200);
6     pinMode(potPin, INPUT);
7 }
8 void loop() {
9     potValue = analogRead(potPin);
10    Serial.print("Potentiometer Value: ");
11    Serial.println(potValue);
12    delay(100);
13 }
```

LED Potentiometer Code Exercise (cont.)

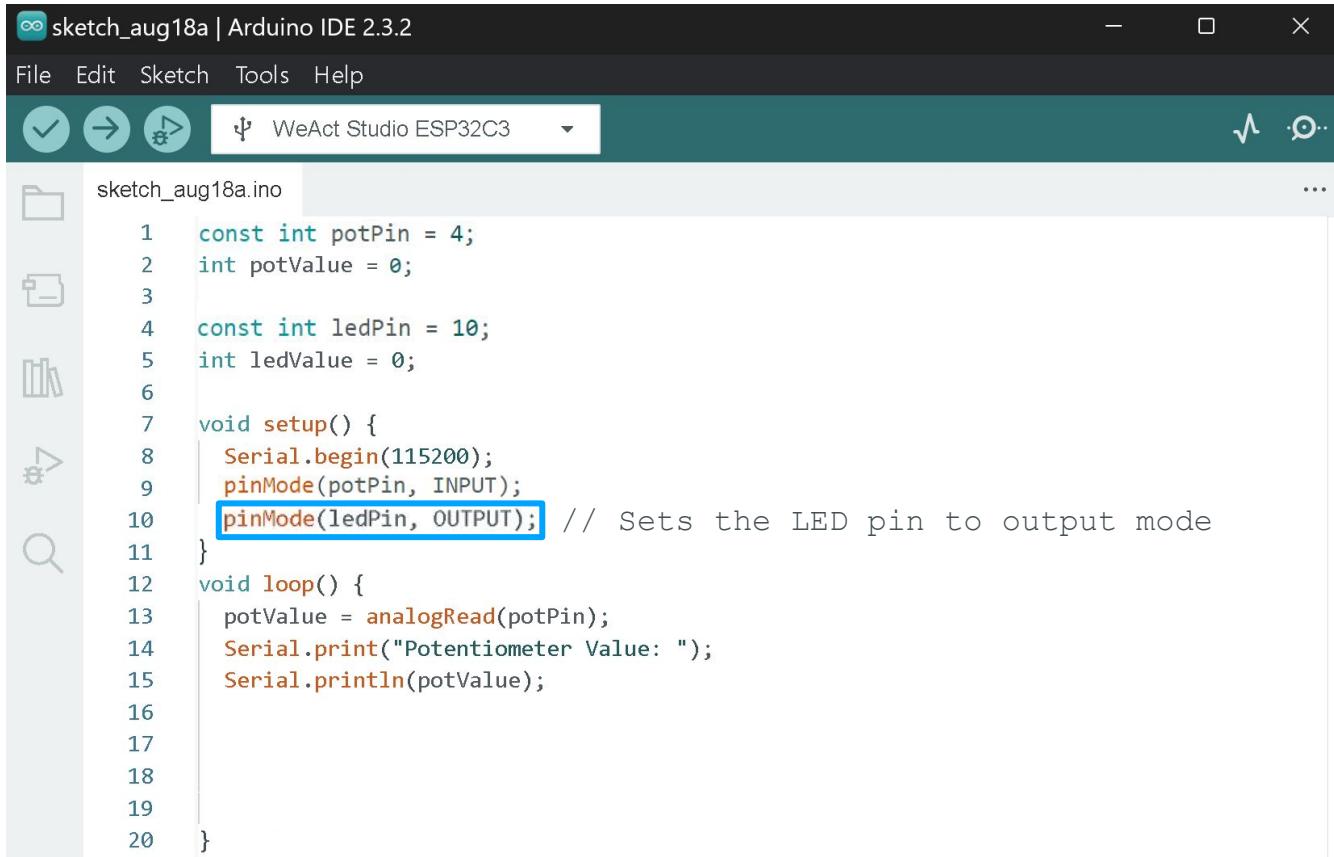
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown for "WeAct Studio ESP32C3".
- Code Editor:** Displays the `sketch_aug18a.ino` file content.
- Code Content:**

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 const int ledPin = 10; //Defines the pin that the LED will connect to
5 int ledValue = 0; // and the value we will write to the LED
6
7 void setup() {
8     Serial.begin(115200);
9     pinMode(potPin, INPUT);
10
11 }
12 void loop() {
13     potValue = analogRead(potPin);
14     Serial.print("Potentiometer Value: ");
15     Serial.println(potValue);
16
17
18
19
20 }
```
- Sidebar:** On the left side, there are icons for File, Sketch, Tools, Help, and a magnifying glass for search.

The line `const int ledPin = 10;` is highlighted with a blue rectangular selection.

LED Potentiometer Code Exercise (cont.)



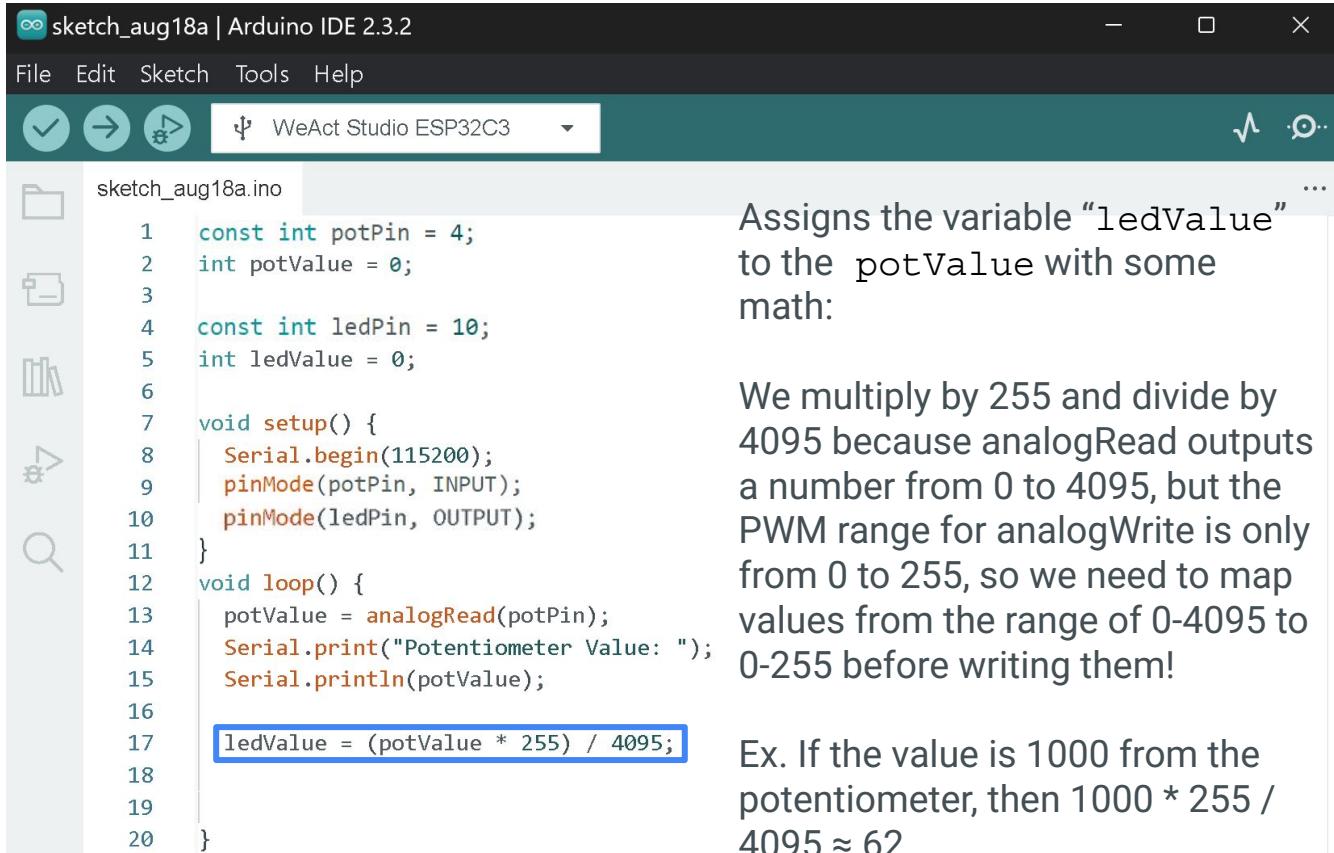
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Toolbar:** File, Edit, Sketch, Tools, Help
- Header Bar:** WeAct Studio ESP32C3
- Sketch List:** sketch_aug18a.ino
- Code Editor:** Displays the following C++ code for an Arduino sketch:

```
1 const int potPin = 4;
2 int potValue = 0;
3
4 const int ledPin = 10;
5 int ledValue = 0;
6
7 void setup() {
8     Serial.begin(115200);
9     pinMode(potPin, INPUT);
10    pinMode(ledPin, OUTPUT); // Sets the LED pin to output mode
11 }
12 void loop() {
13     potValue = analogRead(potPin);
14     Serial.print("Potentiometer Value: ");
15     Serial.println(potValue);
16
17
18
19
20 }
```

The line `pinMode(ledPin, OUTPUT);` is highlighted with a blue rectangle.

LED Potentiometer Code Exercise (cont.)

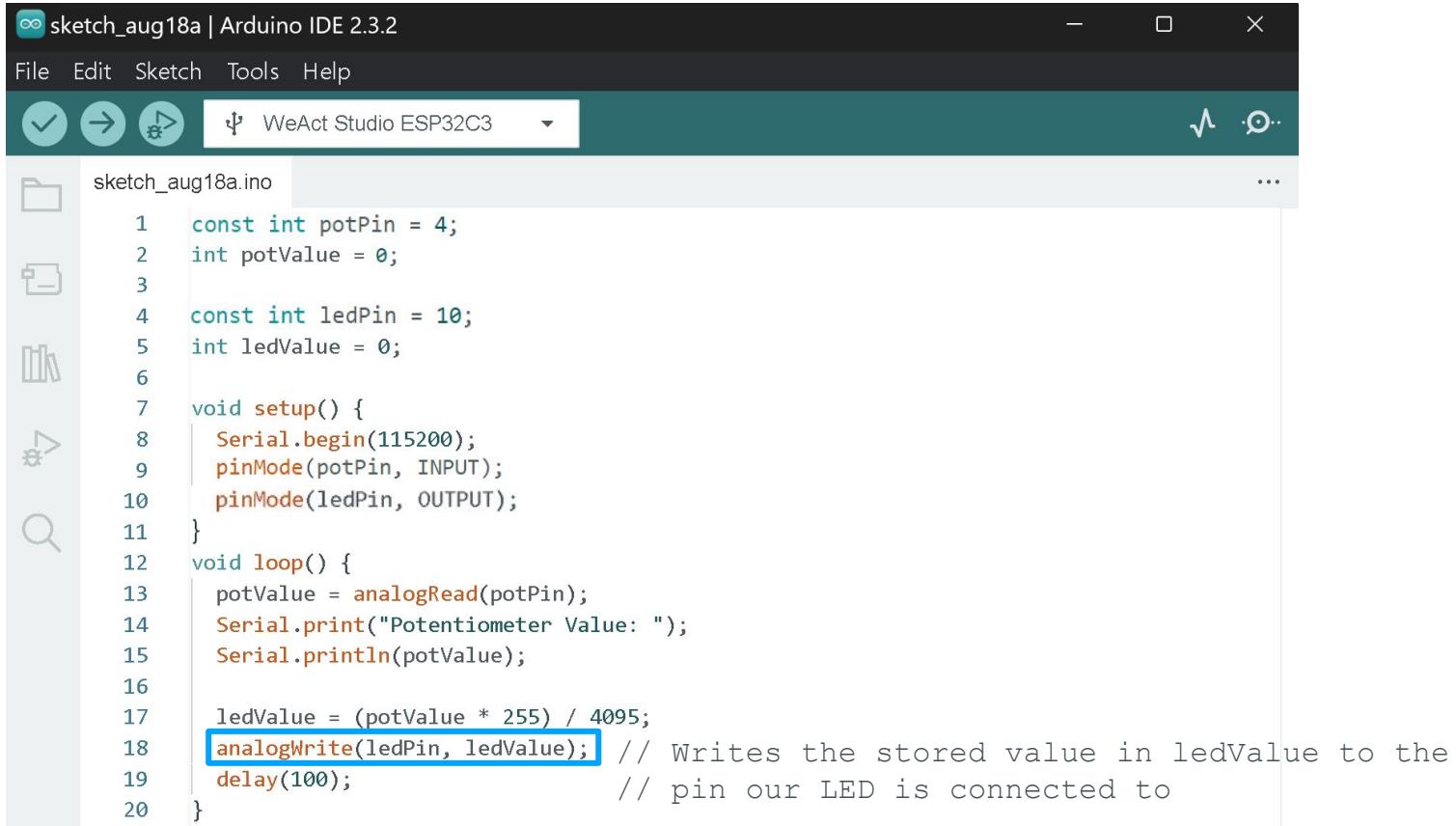


The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_aug18a | Arduino IDE 2.3.2
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Stop, and a dropdown for "WeAct Studio ESP32C3".
- Sketch Area:** Displays the code for "sketch_aug18a.ino".
- Code Content:**

```
1  const int potPin = 4;
2  int potValue = 0;
3
4  const int ledPin = 10;
5  int ledValue = 0;
6
7  void setup() {
8    Serial.begin(115200);
9    pinMode(potPin, INPUT);
10   pinMode(ledPin, OUTPUT);
11 }
12 void loop() {
13   potValue = analogRead(potPin);
14   Serial.print("Potentiometer Value: ");
15   Serial.println(potValue);
16
17   ledValue = (potValue * 255) / 4095;
18
19 }
```
- Annotations:**
 - A callout box points to the line `ledValue = (potValue * 255) / 4095;` with the text: "Assigns the variable “ledValue” to the `potValue` with some math:"
 - A callout box points to the line `ledValue = (potValue * 255) / 4095;` with the text: "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!"
 - A callout box points to the line `ledValue = (potValue * 255) / 4095;` with the text: "Ex. If the value is 1000 from the potentiometer, then $1000 * 255 / 4095 \approx 62$ "

LED Potentiometer Code Example (cont.)



The screenshot shows the WeAct Studio ESP32C3 IDE interface. The title bar reads "sketch_aug18a | Arduino IDE 2.3.2". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar features icons for save, run, and refresh. The central workspace displays the code for "sketch_aug18a.ino". The code uses const variables for potPin (4) and ledPin (10), initializes potValue and ledValue to 0, and sets up the pins. The setup() function begins serial communication at 115200 bps and configures the potentiometer pin as INPUT and the LED pin as OUTPUT. The loop() function reads the potentiometer value, prints it to the Serial monitor, calculates the LED brightness (ledValue), writes it to the LED pin via analogWrite, and adds a 100ms delay. The code is color-coded for readability.

```
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
                                 // pin our LED is connected to
    delay(100);                // pin our LED is connected to
}
```

FAIR USE DISCLAIMER

Copyright Disclaimer under section 107 of the Copyright Act 1976, allowance is made for “fair use” for purposes such as criticism, comment, news reporting, teaching, scholarship, education and research.

Fair use is a use permitted by copyright statute that might otherwise be infringing.

Non-profit, educational or personal use tips the balance in favor of fair use.

CC BY-NC-SA 4.0

This work by the Institute of Electrical and Electronics Engineers, UC Irvine Branch, is licensed under CC BY-NC-SA 4.0