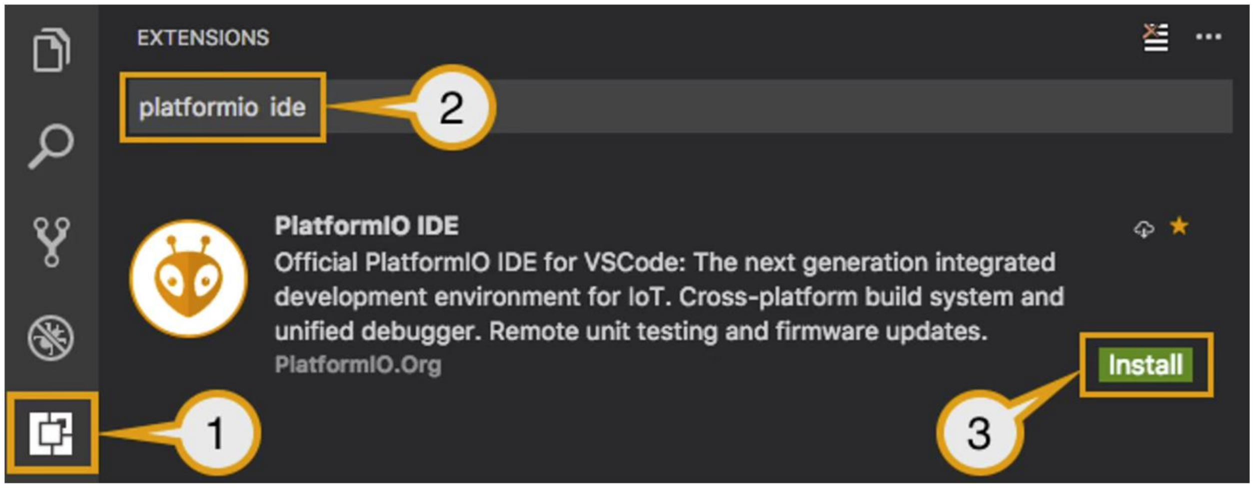
Lab 0 - Intro to the IoT: Design and Applications

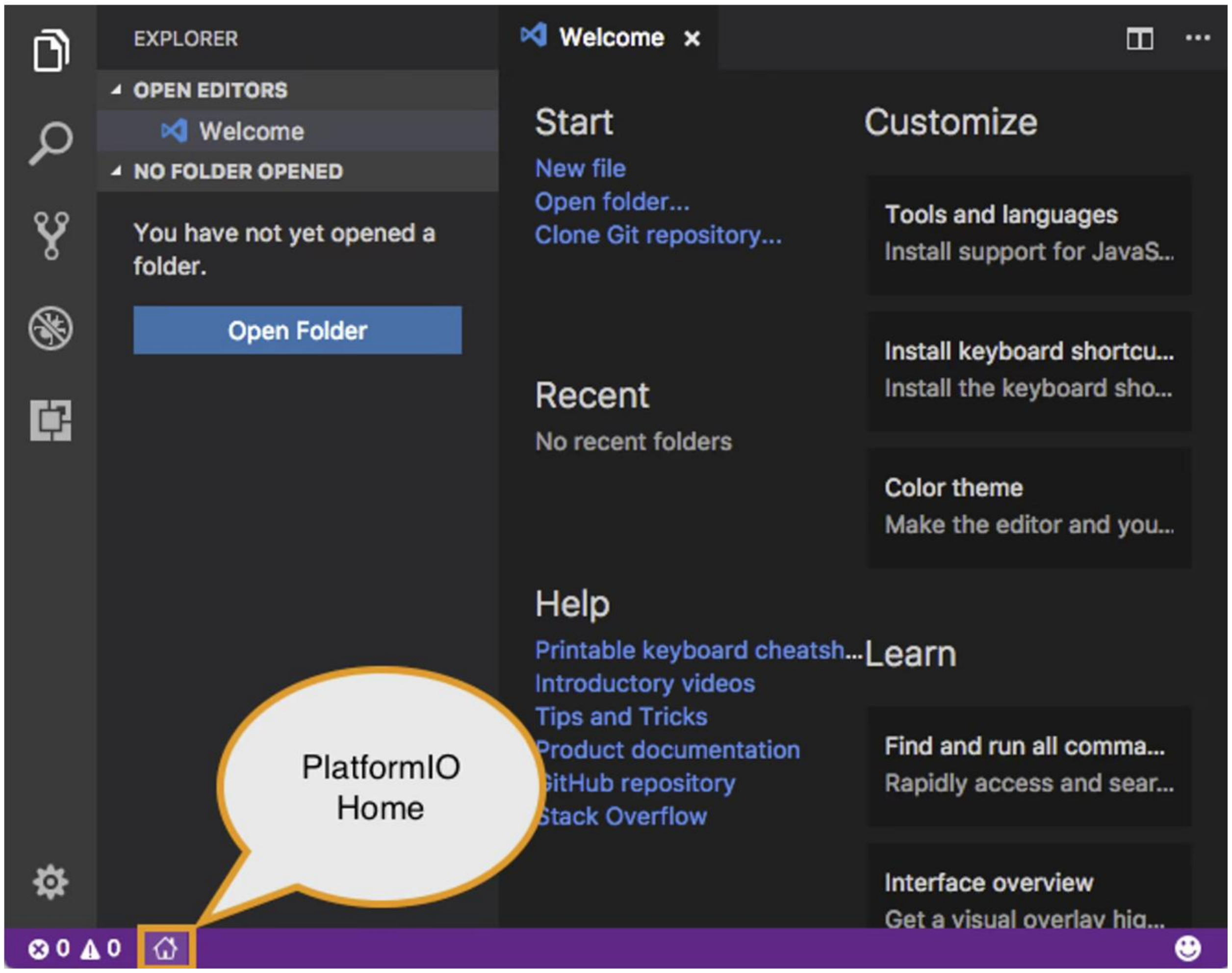
This exercise aims to get a first hands-on experience with the ESP32 board and PlatformIO development environment (IDE) that allows you to quickly build applications involving digital and analog I/O and a debugging console.
`PlatformIO` is a cross-platform, cross-architecture, multiple framework, professional tool for embedded systems engineers and for software developers who write applications for embedded products. We will be running it under Microsoft Visual Studio Code, a free development platform available for Linux, Windows, and Mac OS X. The first thing we need to do is get `Visual Studio Code` installed. Please follow the instructions from here to install both VS Code and PlatformIO extension.

1. *Download and install the official Microsoft Visual Studio Code. PlatformIO IDE is built on top of it*
2. *Open VSCode Package Manager*
3. *Search for the official platformio ide extension*
4. *Install PlatformIO IDE*

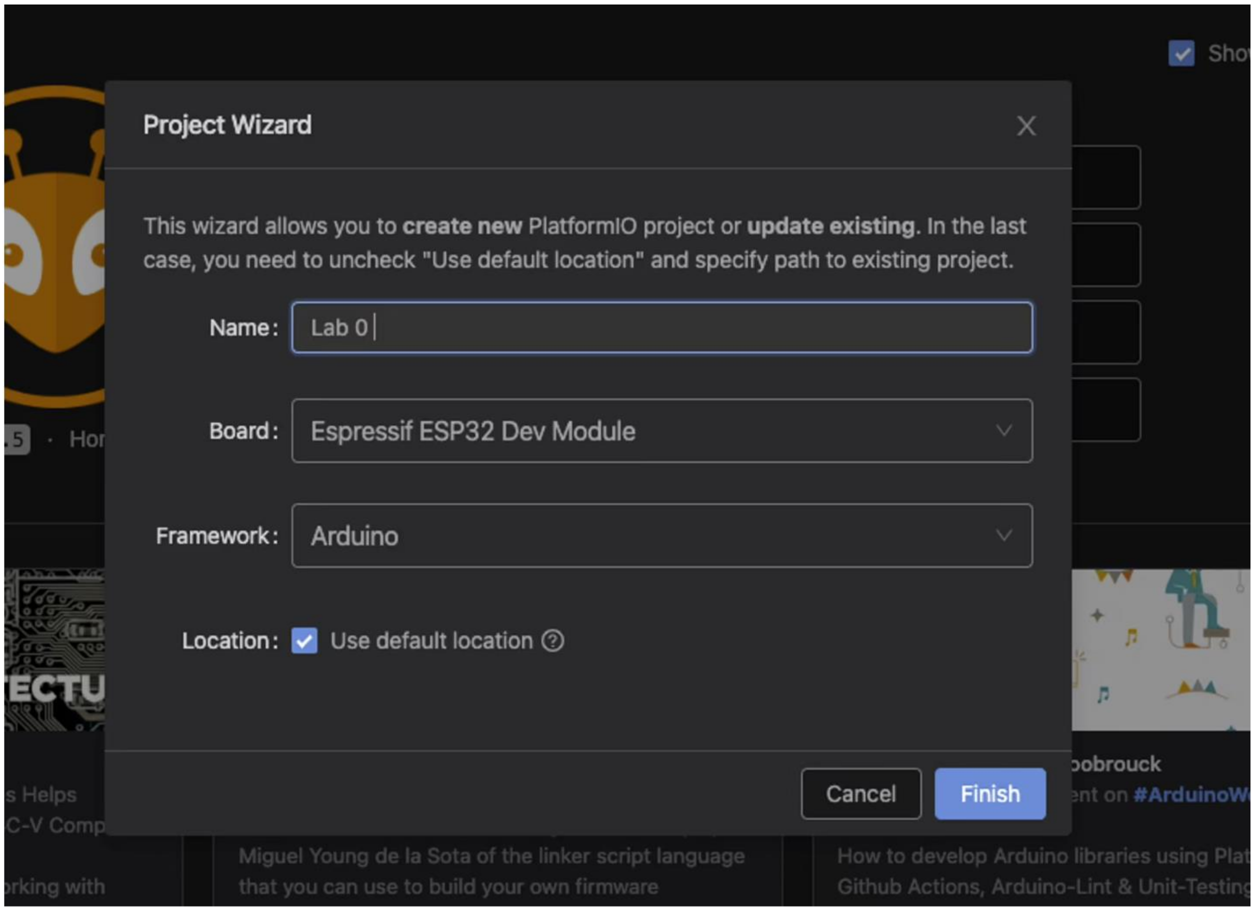

EXTENSIONS
platformio ide
2
PlatformIO IDE
Official PlatformIO IDE for VSCode: The next generation integrated development environment for IoT. Cross-platform build system and unified debugger. Remote unit testing and firmware updates.
PlatformIO.Org
Install
1
3

Setting Up the Project:

1- Click on "PlatformIO Home" button on the bottom PlatformIO Toolbar



2- Click on “New Project”, select a board and create a new PlatformIO Project



Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

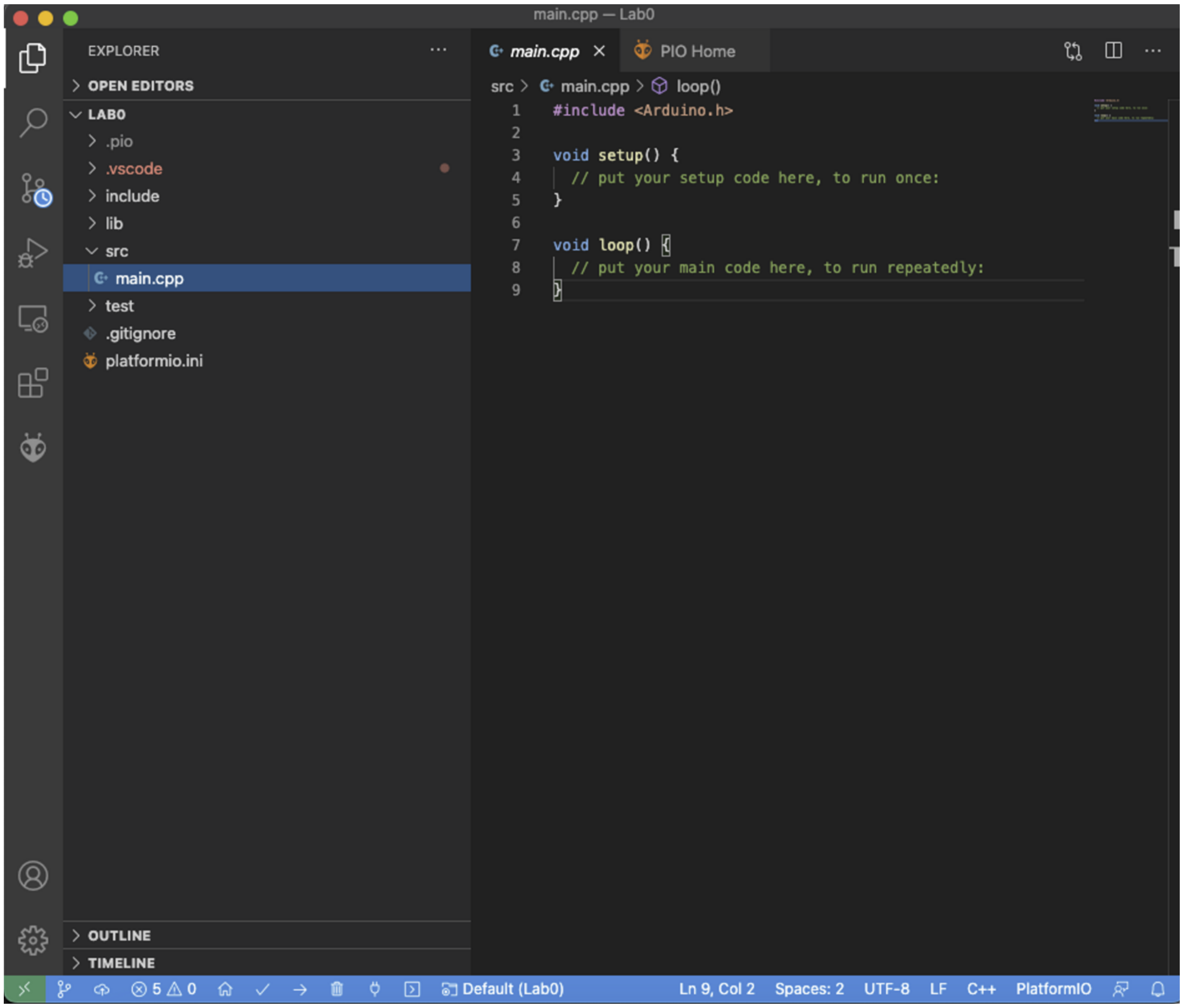
Name: Lab 0

Board: Espressif ESP32 Dev Module

Framework: Arduino

Location: ✓ Use default location

3- Open *main.cpp* file from *src* folder:



```
#include <Arduino.h>

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

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

4- Build your project with ctrl+alt+b hotkey or use the "Build" button on the PlatformIO Toolbar.

Further reading:

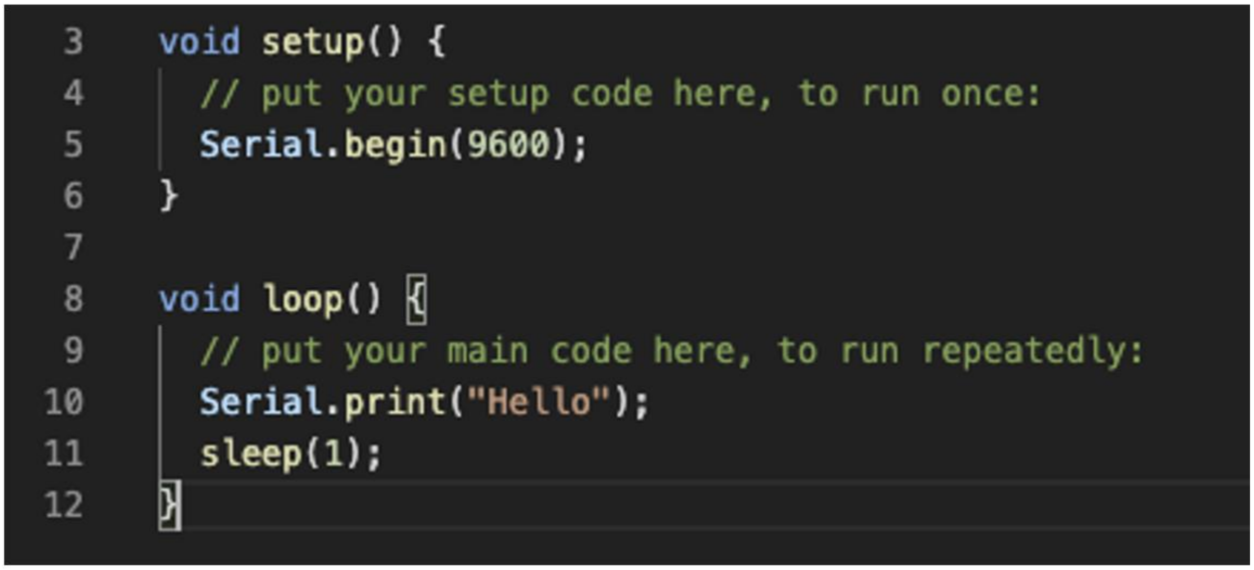
- Tutorials and Examples (step-by-step tutorials with debugging and unit testing)
- Learn more about PlatformIO Toolbar and other commands (Upload, Clean, Serial Monitor) below.

Serial Monitor

PlatformIO provides a built-in serial monitor capable of displaying text sent from the board's serial port (integrated with USB). This monitor is particularly useful when debugging applications, therefore we will use it as our main output during the rest of this demo.

In order to use the board's serial port, an application must first initialize it. This is done within the `setup()` function of the application by using `Serial.begin(baudrate)`, where baudrate is an integer that specifies the baud rate the port must use. Once the serial port has been initialized, the application can send a text to it by using the `Serial.print()` within the `loop()` function. `print()` is a very versatile function that allows printing numerical values, characters, and strings, as shown next:

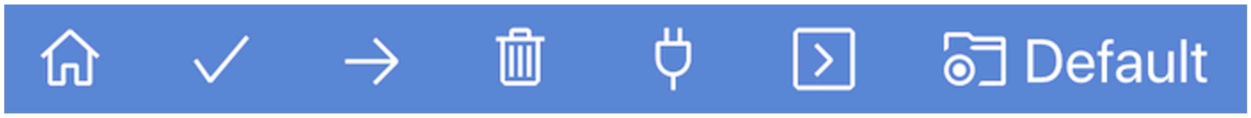
- *Serial.print(78) gives "78"*
- *Serial.print(1.23456) gives "1.23"*
- *Serial.print('N') gives "N"*
- *Serial.print("Hello world.") gives "Hello world."*



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print("Hello");
  sleep(1);
}
```

The first step is to configure the IDE to use the correct USB port for the monitor. Connect the board to the PC using the USB cable, then click on the “Serial Monitor” from the PlatformIO toolbar.



You can customize Serial Port Monitor using Monitor options in “platformio.ini” (Project Configuration File). Further information is provided here.

Deliverable

Your task is to make a light meter gage with a photo-resistor, a servo, and an ESP32 board.

Actuator: Servo

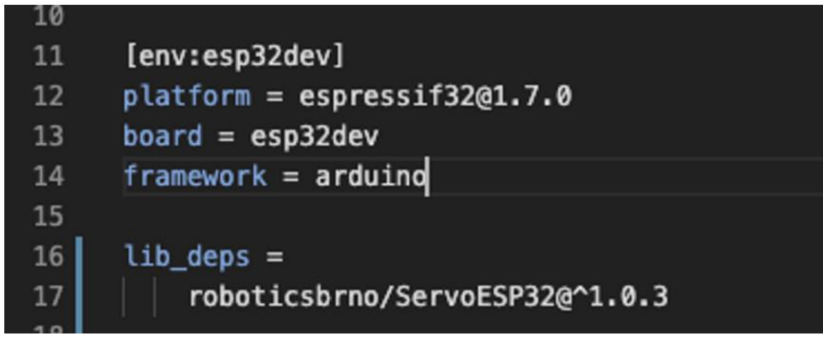
Servo is a motor that you can locate at any position between 0 to 179 degrees. It uses the "Servo.h" library and operates with the write command:

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo
void setup()
{
	myservo.attach(27); // attaches the servo on port 27 to the servo object
}
void loop()
{
	myservo.write(50); // fixes the servo at 50 degrees

}
```

To install the Servo library, follow the below instructions:

1. Open platformio.ini, a project configuration file located at the root of the PlatformIO project.
2. Add the following line to the lib_deps option of *[env:]* section: `roboticsbrno/ServoESP32@^1.0.3`
3. Build a project, PlatformIO will automatically install dependencies.



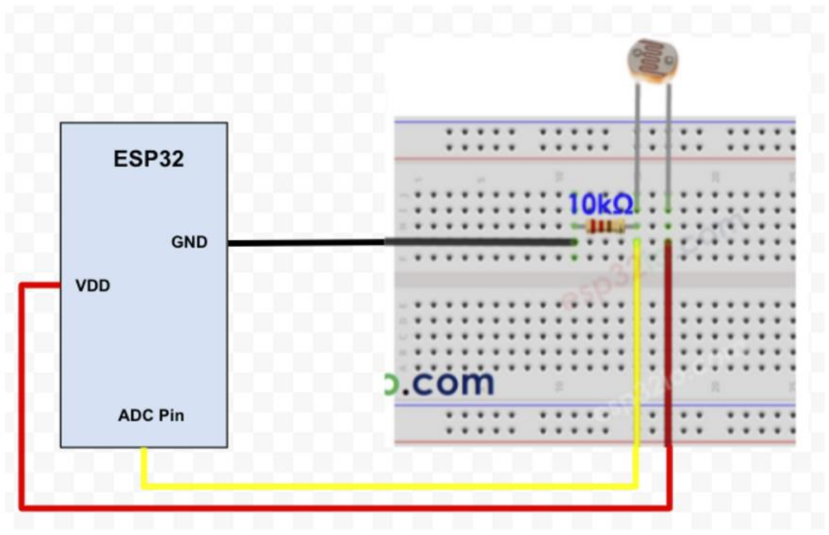
```
[env:esp32dev]
platform = espressif32@1.7.0
board = esp32dev
framework = arduino

lib_deps =
    roboticsbrno/ServoESP32@^1.0.3
```

For further examples, please check here.

Sensor: Photo-resistor

The photocell, a photoresistor or light-dependent resistor (LDR), is a two-terminal, resistive component that increases or decreases its resistance depending on the light it senses. To measure the photocell's resistance with a microcontroller's ADC, we actually have to use it to generate a variable voltage. By combining the photocell with a static resistor, we can create a voltage divider that produces a voltage-dependent on the photocell's resistance. A static resistor value between 1kΩ and 10kΩ should pair well with the photocell. ESP32's analog input pin converts the voltage (between 0 and 3.3v) into integer values, called analog value or ADC value. By connecting an analog input pin of ESP32 to the photoresistor, we can read the analog value by using `analogRead(pinNumber)` function.



Please check here for more information about the Photoresistor and here for another example with ESP32 for a related topic.

Piecing it together

The Light Meter demonstration must include the following phases

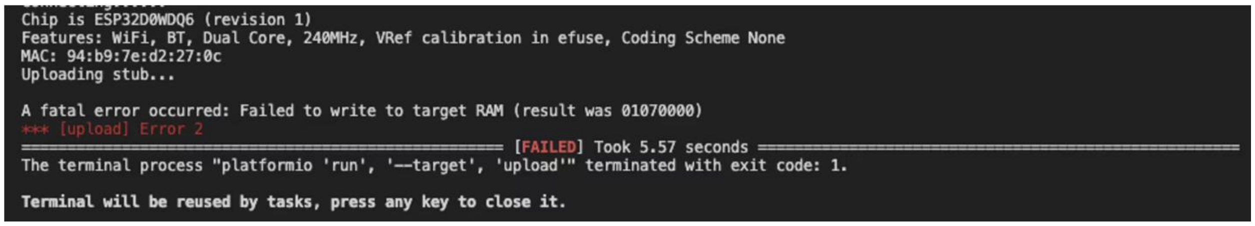
- **Phase 1- Calibration:** You measure the maximum and minimum available light in this phase. This phase takes 10 seconds long and an LED starts blinking. During this phase, you should face the light sensor to the maximum available light around yourself (e.g. your PC display) and also to the minimum light (you can put your finger on the sensor). (HINT: find and store the maximum and minimum values of the light sensor in two variables.)

- **Phase 2- Operation**: after 10 seconds calibration phase the LED stops blinking and the Servo starts moving like a gage according to the amount of light. It should move to 0 degree position for the darkest condition and to 179 degree position in the lightest condition and somewhere in between for the current room light. *(HINT: map the maximum and minimum analog voltage values to 0 and 179, and then move the servo to the mapped value.)*

Troubleshooting

Driver issues

If you compile and upload the code on macOS and get the following results:



```
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: 94:b9:7e:d2:27:0c
Uploading stub...

A fatal error occurred: Failed to write to target RAM (result was 01070000)
*** [upload] Error 2
=================================================== [FAILED] Took 5.57 seconds ===================================================
The terminal process "platformio 'run', '--target', 'upload'" terminated with exit code: 1.

Terminal will be reused by tasks, press any key to close it.
```

Step 1: Install driver

Please install the proper driver from here.

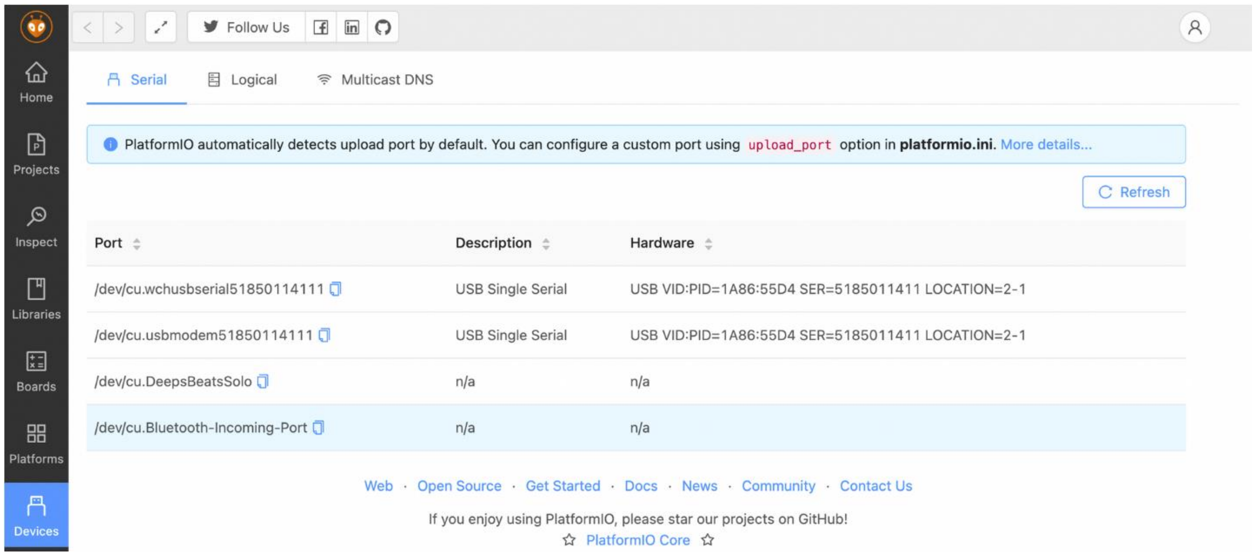
Step 2: Restart laptop.

Step 3: List available devices

Terminal method: Enter the following command in the macOS terminal to list all available ports while the board is connected to your computer.

```
ls  /dev/cu.*
```

GUI Method: You can also find the list of available devices by navigating to `Devices` page on PlatformIO.



PlatformIO automatically detects upload port by default. You can configure a custom port using `upload_port` option in **platformio.ini**. More details...

Port	Description	Hardware
/dev/cu.wchusbserial51850114111	USB Single Serial	USB VID:PID=1A86:55D4 SER=5185011411 LOCATION=2-1
/dev/cu.usbmodem51850114111	USB Single Serial	USB VID:PID=1A86:55D4 SER=5185011411 LOCATION=2-1
/dev/cu.DeepsBeatsSolo	n/a	n/a
/dev/cu.Bluetooth-Incoming-Port	n/a	n/a

Web · Open Source · Get Started · Docs · News · Community · Contact Us

If you enjoy using PlatformIO, please star our projects on GitHub!
☆ PlatformIO Core ☆

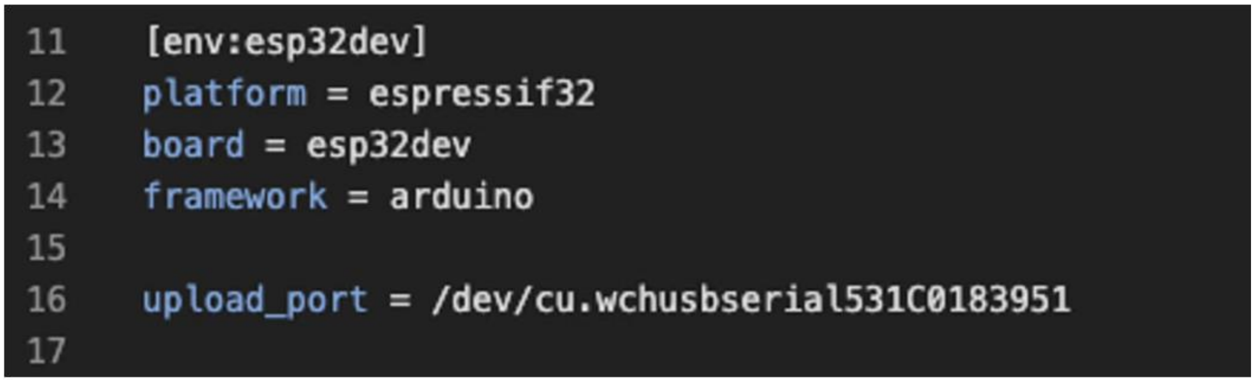
You will see:

1. /dev/cu.usbmodemXXXXXXXX
2. /dev/cu.wchusbserialXXXXXXXX

Make sure to choose the second option!

Step 4: Overwrite upload_port

Finally, write the correct port address in the PlatformIO configuration file as follows:



```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino

upload_port = /dev/cu.wchusbserial531C0183951

```

In the above example, `upload_port` determines the port address.