SYMBOLA

ESP32 Hello World Blinky

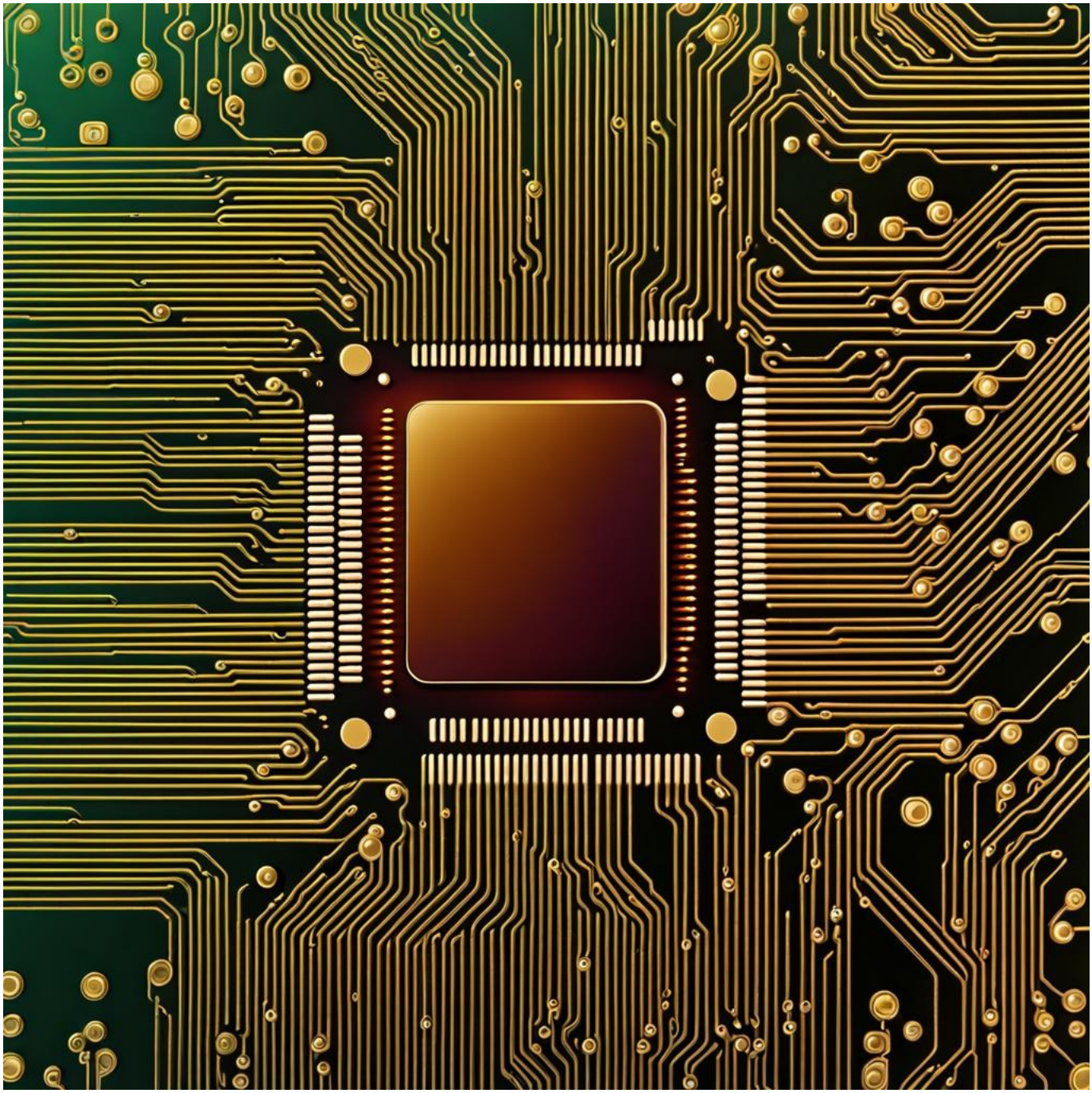


Table of Contents

Disclaimer

This guide is provided for informational purposes only. Every effort has been made to ensure the accuracy and completeness of this guide, but it is provided “as is” without warranty of any kind, express or implied. The author shall not be held liable for any direct, indirect, incidental, or consequential damages or losses arising from the use of this guide.

The procedures and software described in this guide are subject to change and may not be up-to-date. Users are advised to exercise caution and consider their specific circumstances when following the instructions.

This guide may contain links to external websites. The author is not responsible for the content or accuracy of any external site.

Please use this guide responsibly and at your own risk.

Contact the Author

If you've spotted an error or simply wish to make contact, feel free to leave a message at:

https://symbola.co.uk/contact/

Your feedback and inquiries are always welcome!

Preface

The ESP32 is a versatile and popular microcontroller for IoT projects. This guide will show you how to set up your development environment with Visual Studio Code and PlatformIO and create a basic project that prints a message to the serial port and blinks an LED.

Hardware / Software Requirements

To follow this guide effectively, you will need the following hardware and software:

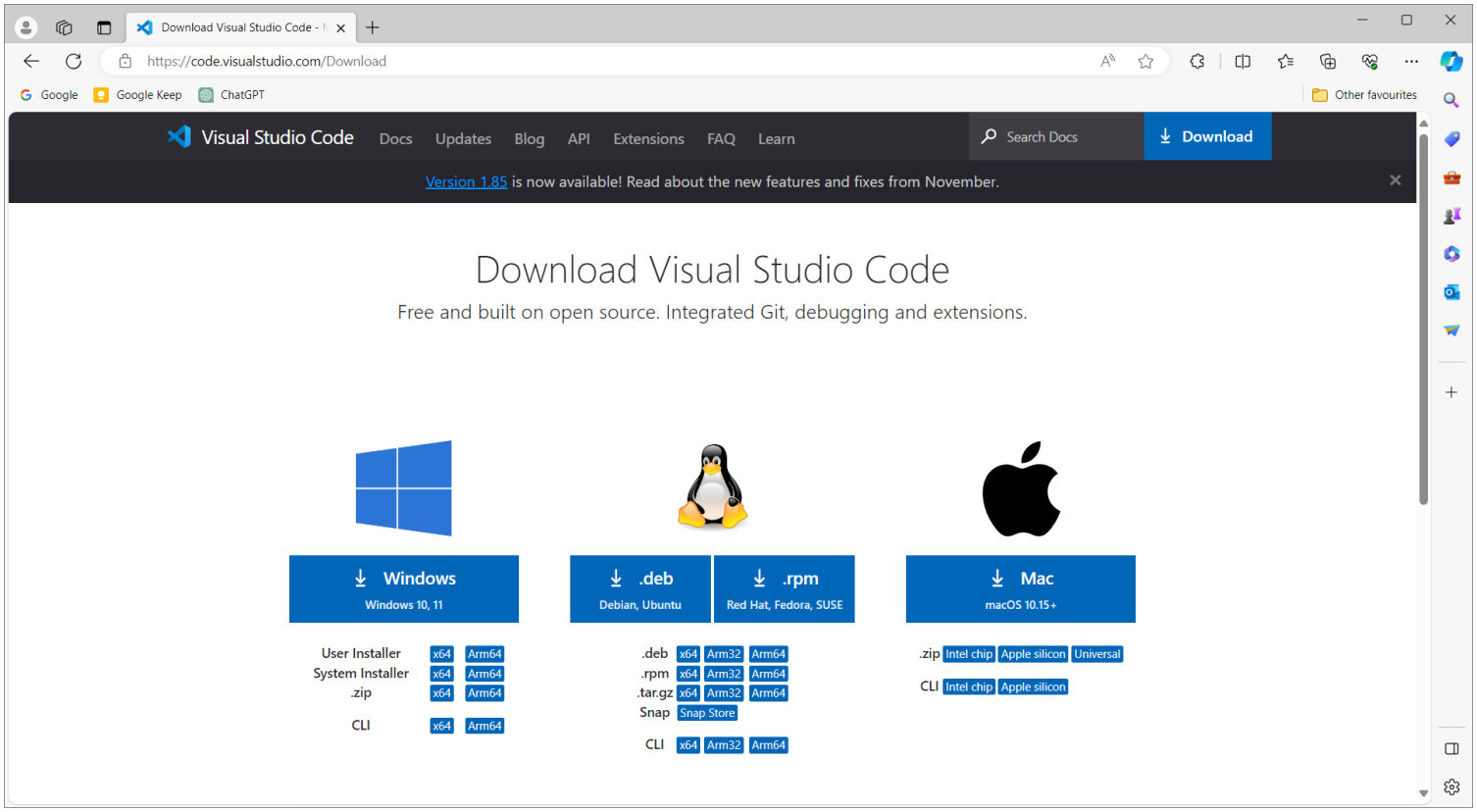
1. Windows PC:
   - Operating System: Windows 10 or Windows 11. (Note: This guide uses Windows 11 Pro 64-bit for demonstrations).
2. ESP32 Development Board:
   - Model: ESP32 with 38 Pins. A commonly used model that can be found on AliExpress.
   - https://www.aliexpress.com/item/32959541446.html (ESP-32 38Pin).
3. Micro USB Cable:
   - Used for connecting the ESP32 board to your computer for programming and power supply.
4. LED:
   - A standard light-emitting diode for use in the "Hello World" blinking example.
5. Resistor:
   - A 150-200 Ohm resistor to limit current to the LED, preventing potential damage.
6. Breadboard:
   - A solderless breadboard for assembling the circuit without soldering.
7. Jumper Wires:
   - Also known as linker wires, these are used for making connections on the breadboard between the ESP32, LED, resistor, and other components.

While this guide specifically uses Windows 10/11 for demonstrations, it's worth noting that the ESP32 can also be programmed using other operating systems like macOS and Linux. However, the steps for setting up the development environment and programming the ESP32 may differ in these systems and are not covered in this guide.

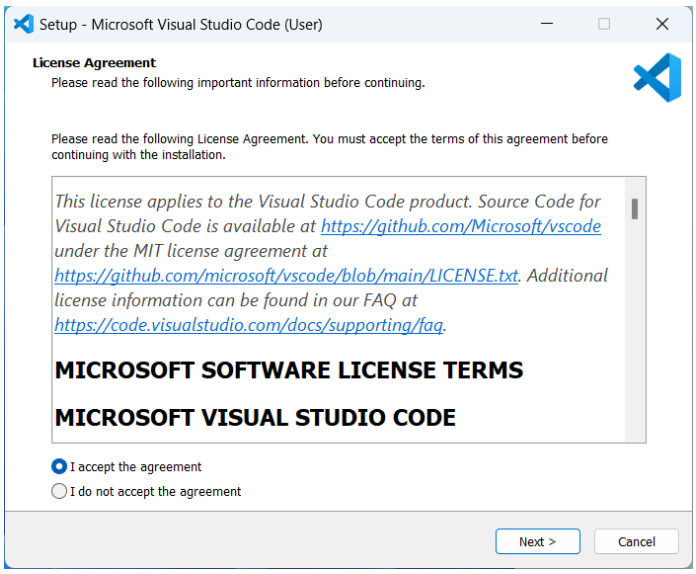
Installing Visual Studio Code

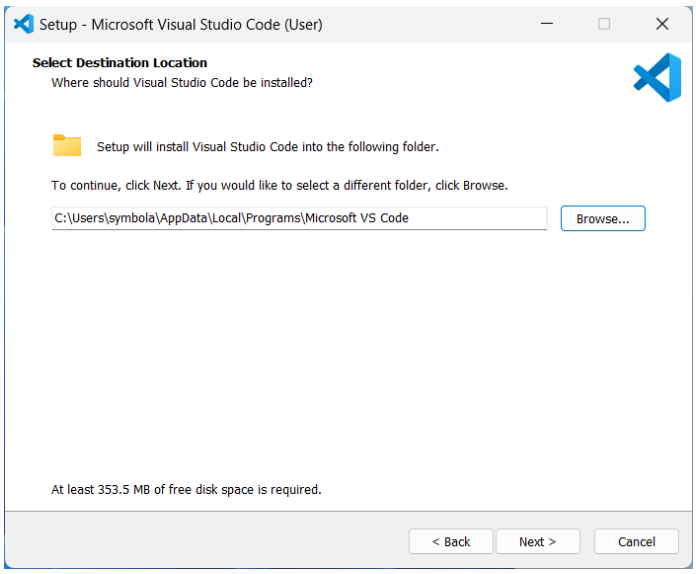
Visual Studio Code (VS Code)

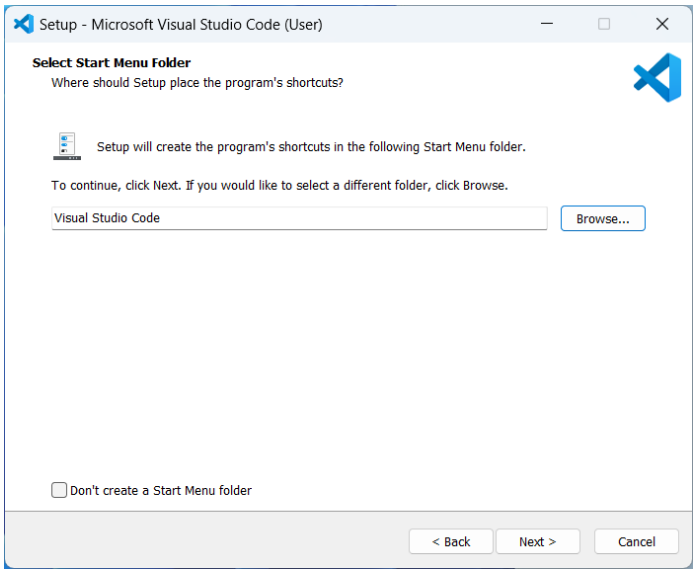
1. Download the Installer:
   - Visit code.visualstudio.com/download.
   - Select the appropriate installer for your operating system (User Installer x64 used in this guide).

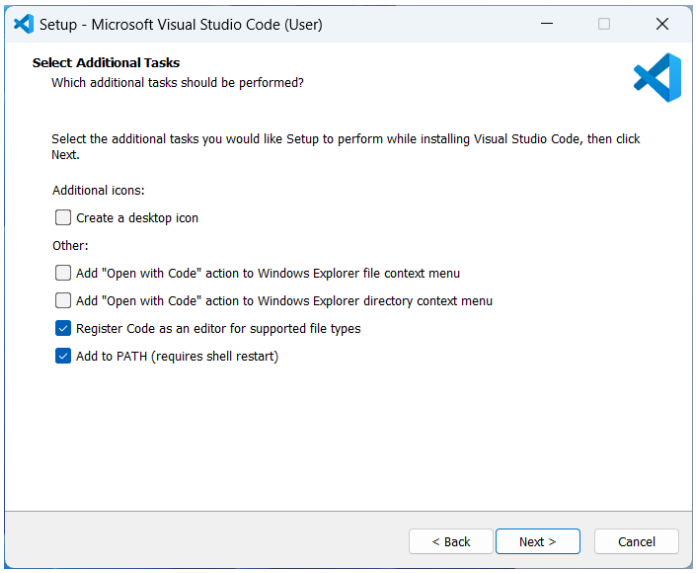


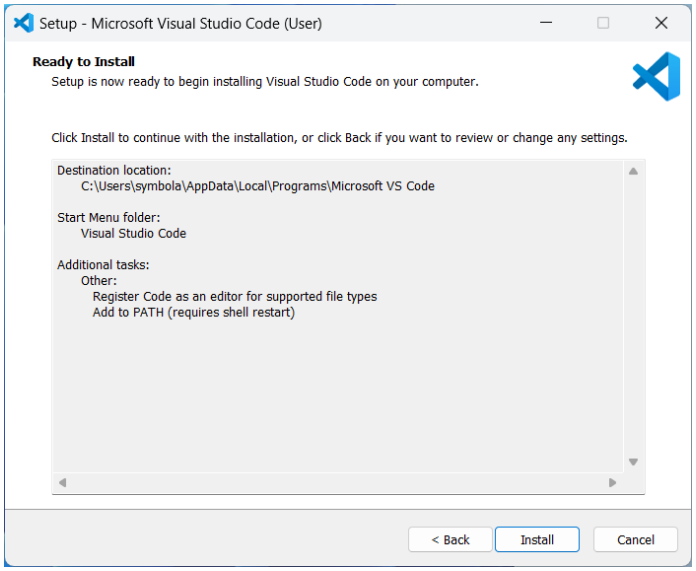
Once the download is complete, run the installer. Follow the on-screen instructions to complete the installation process.

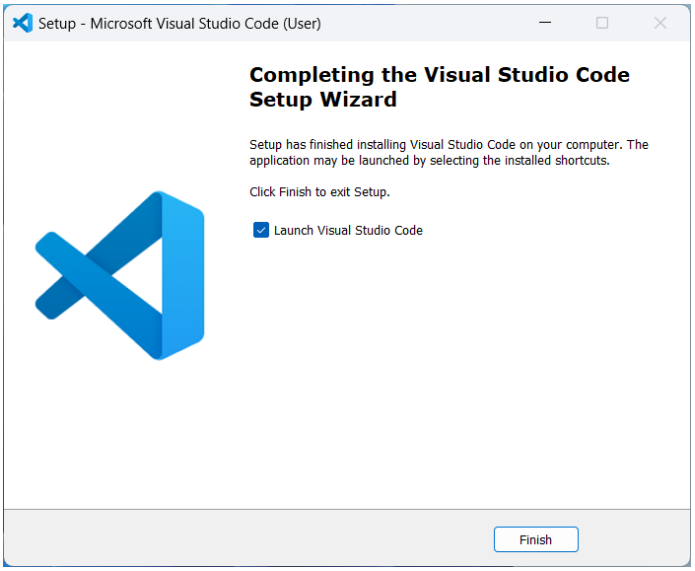






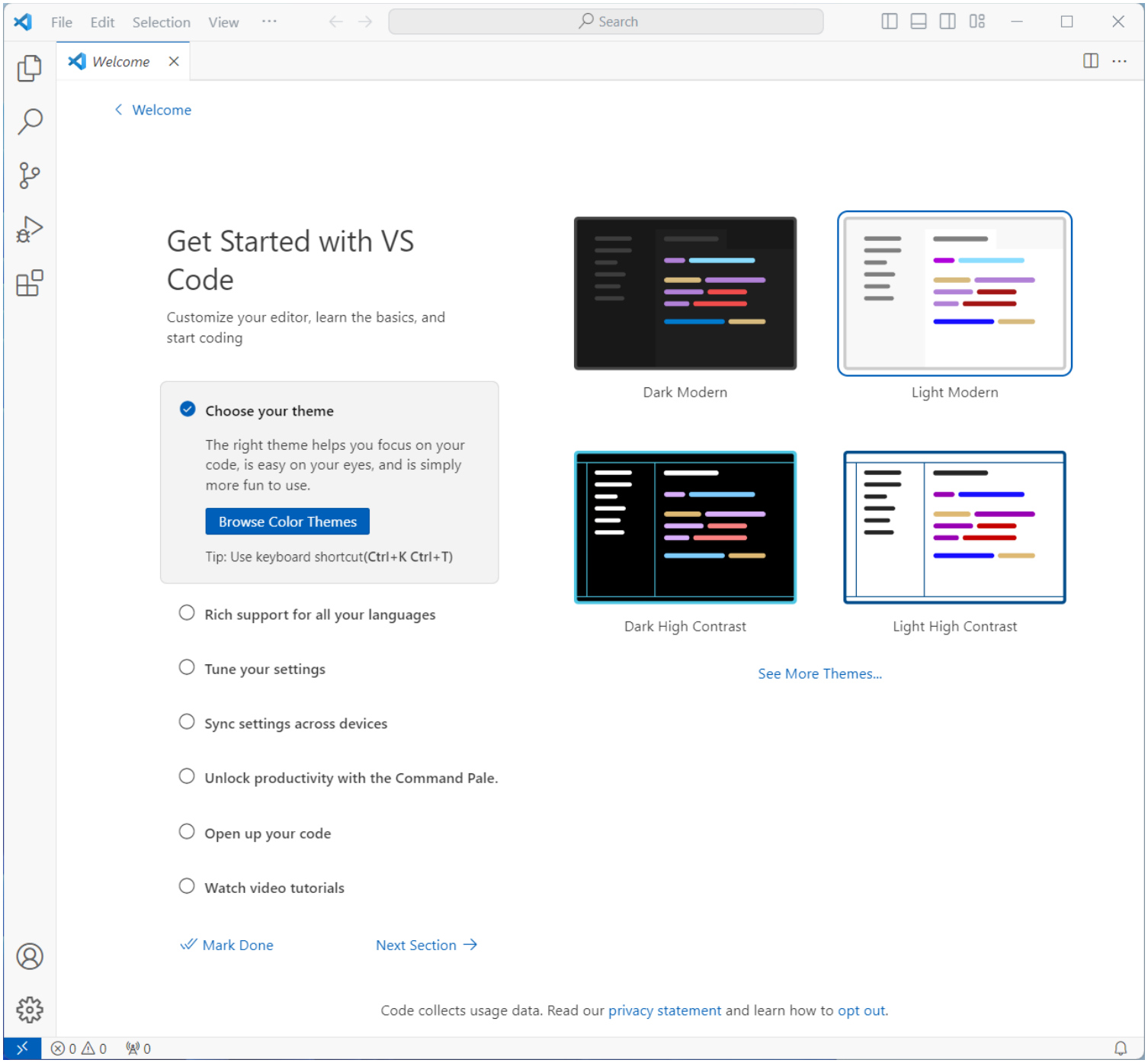






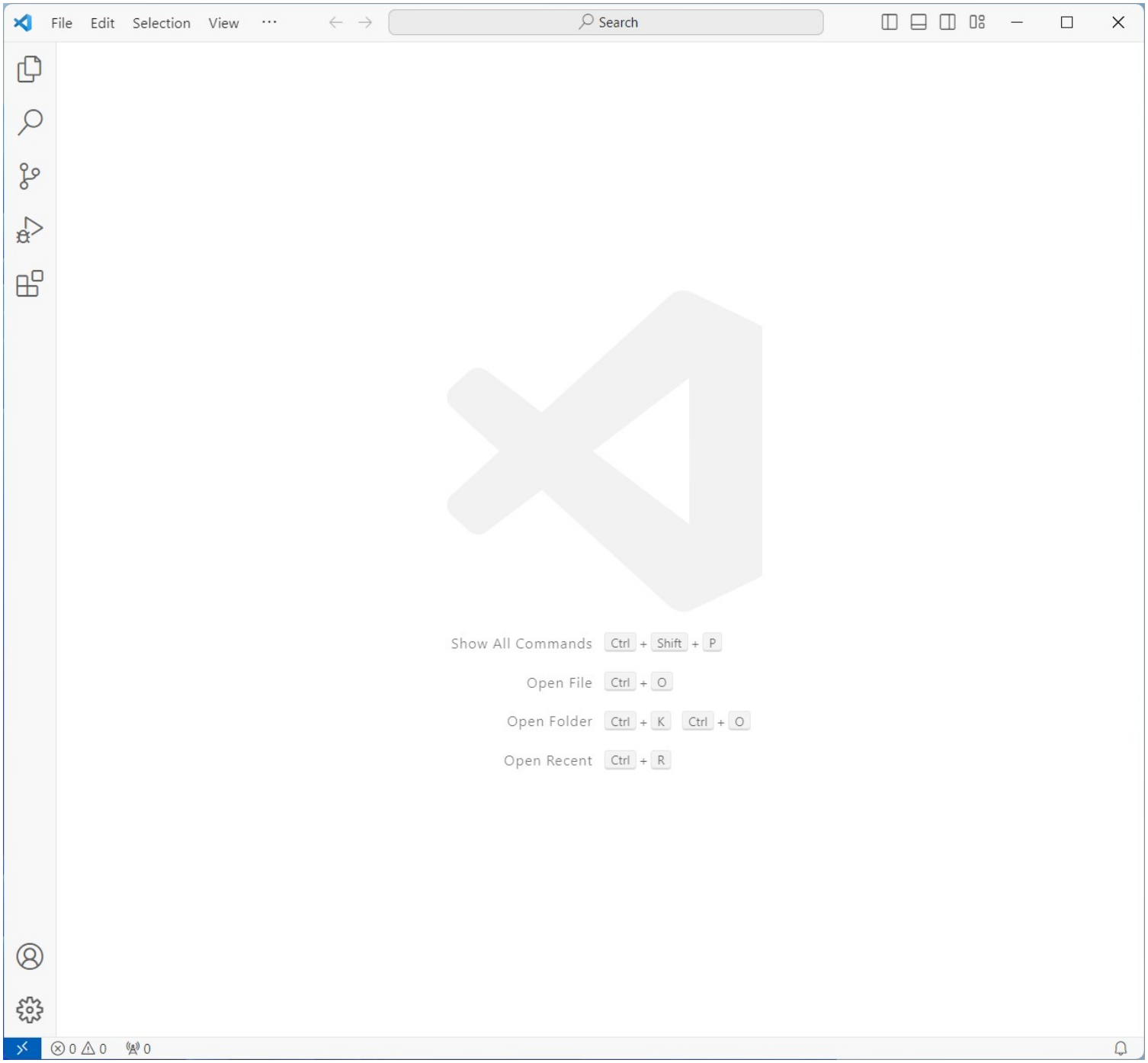
Overview of Visual Studio Code Interface

Once you have Visual Studio Code installed, you'll be introduced to its user-friendly interface.



The Welcome tab, which appears when you first open Visual Studio Code, is helpful for initial navigation but can be closed once you're familiar with the environment. You can always reopen it later if needed, through the 'Help' menu.

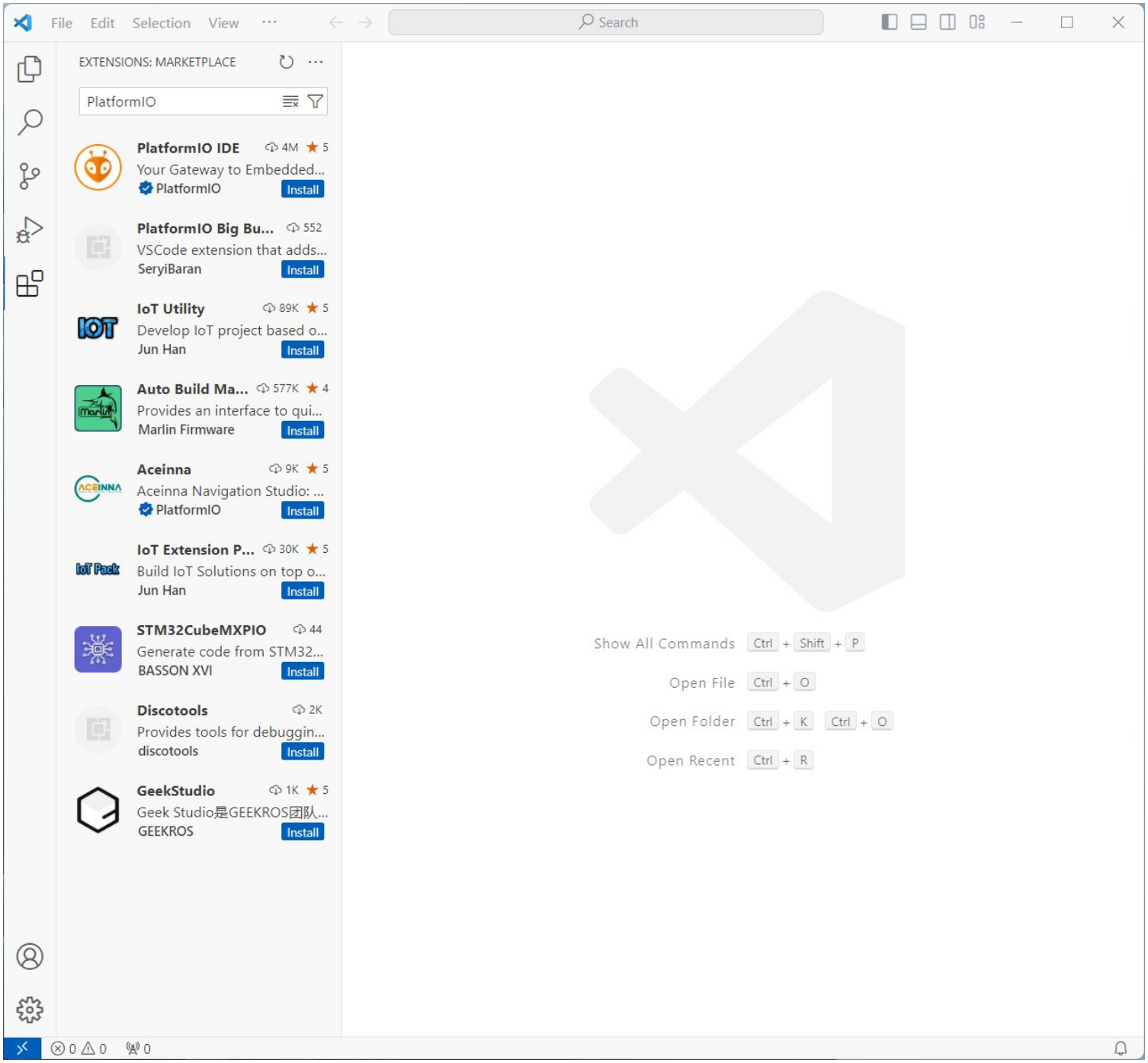
Extensions in VS Code add new capabilities to the software. This is where you can search for and install additional functionalities, like PlatformIO, which is essential for ESP32 development. Extensions can range from language support, themes, debuggers, to code linters and more.



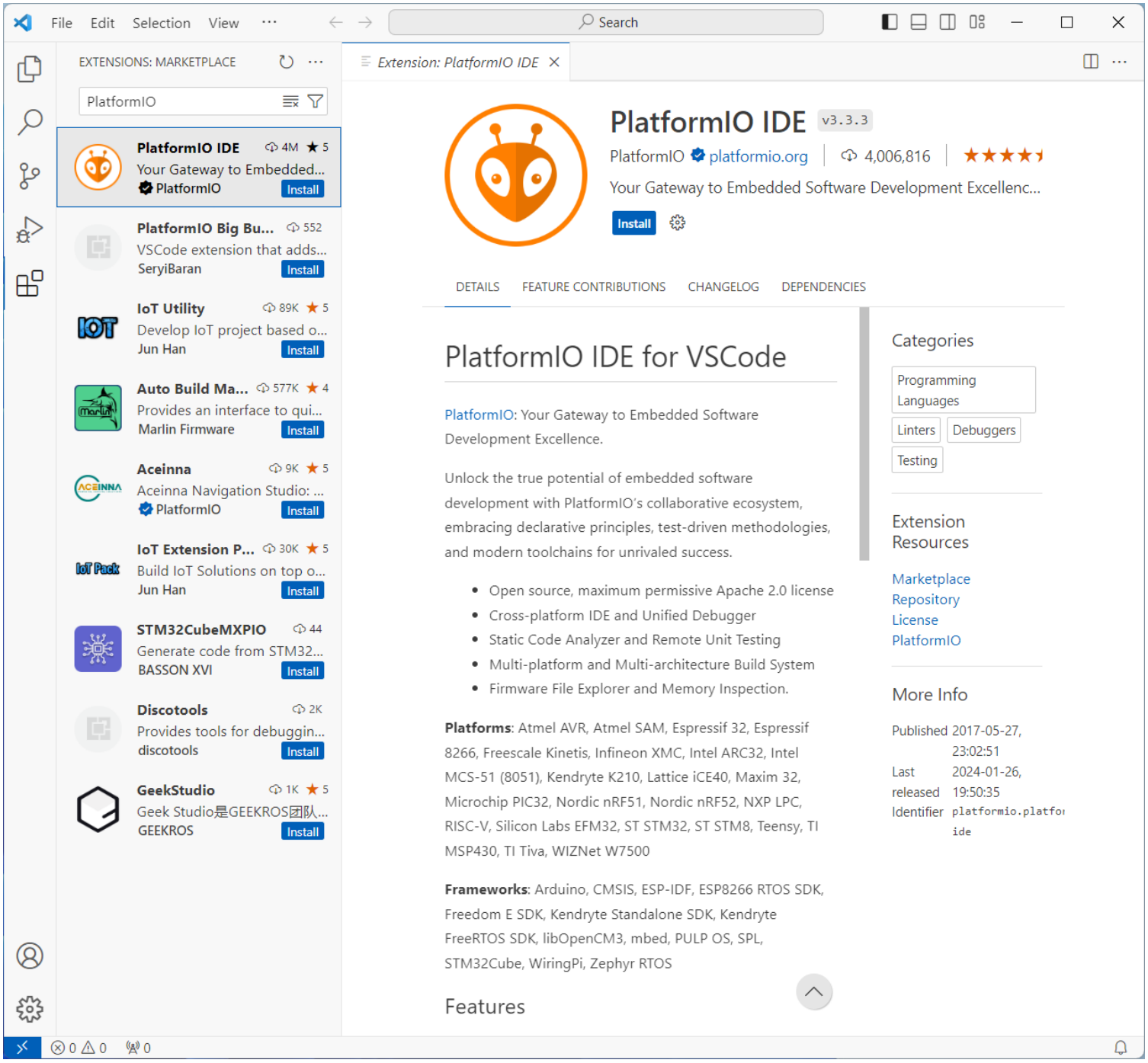
Installing PlatformIO

PlatformIO is an essential extension for Visual Studio Code that provides a powerful platform for IoT development, including with the ESP32. Follow these steps to install the PlatformIO IDE extension:

1. Open the Extensions View:
   - In Visual Studio Code, click on the Extensions icon in the Activity Bar on the side of the window.
2. Search for PlatformIO:
   - In the Extensions view, enter "PlatformIO" in the search bar. You'll see a list of extensions related to PlatformIO.



1. Install PlatformIO IDE Extension:
   - Find "PlatformIO IDE" in the search results. It's an official extension by PlatformIO.
   - Click on the 'Install' button. Visual Studio Code will download and install the PlatformIO IDE extension.
   - After installation, a PlatformIO icon will appear in the Activity Bar, indicating that the extension is now active.

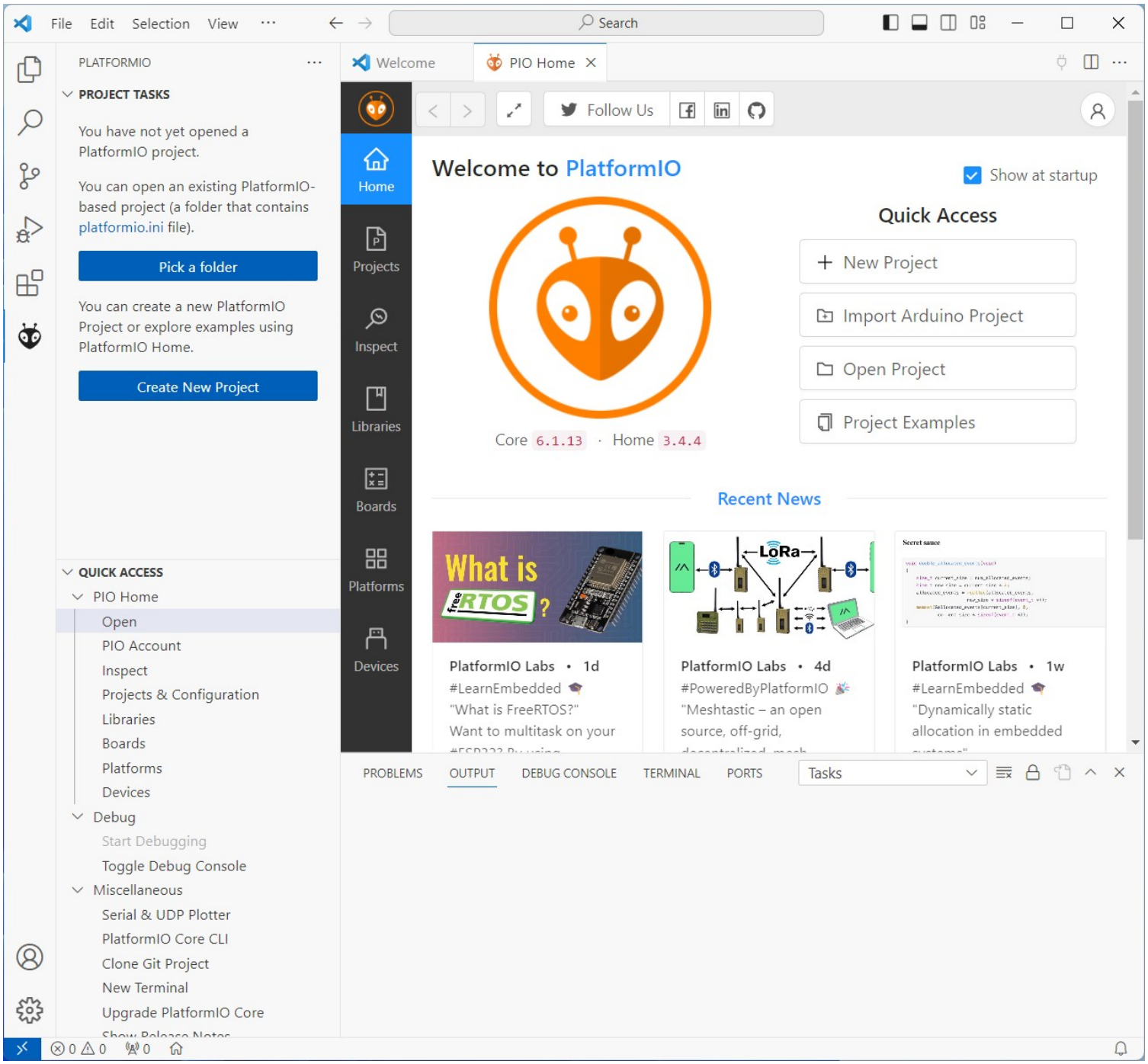


   - Upon first installation, PlatformIO might take a few moments to install necessary core packages and dependencies.
   - Once the setup is complete, you might need to restart Visual Studio Code to activate the extension fully.

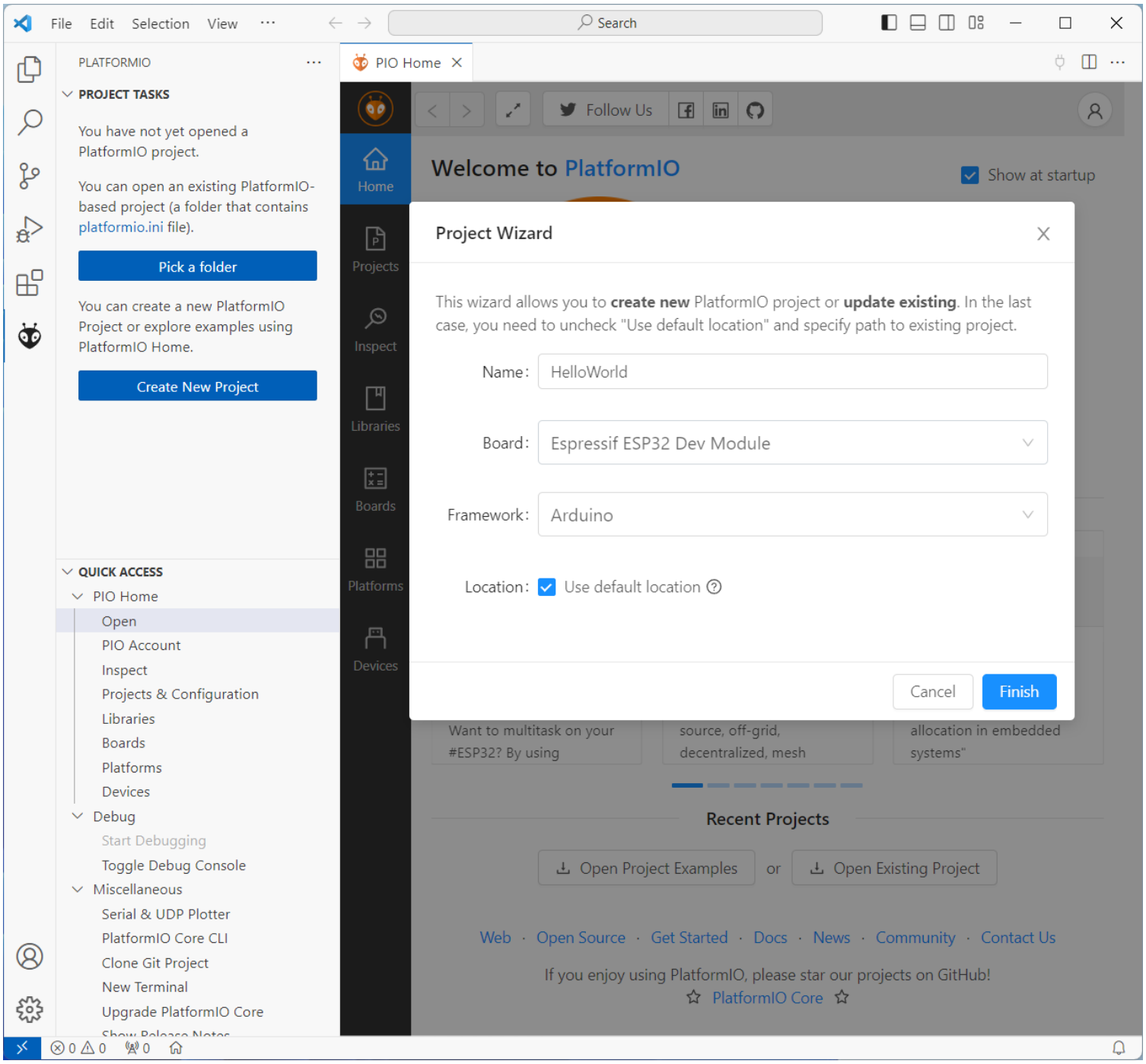
Creating Your First Project

Create a New Project:

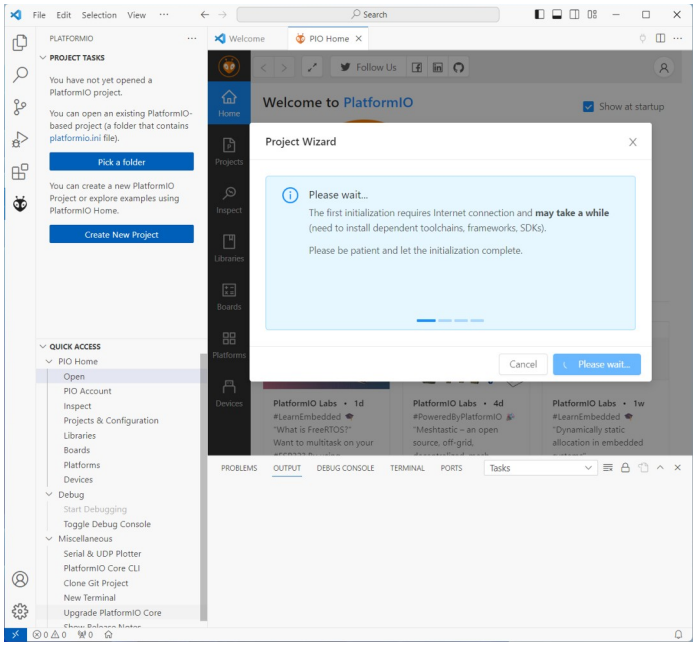
- Click on "New Project" in the PlatformIO Home screen.

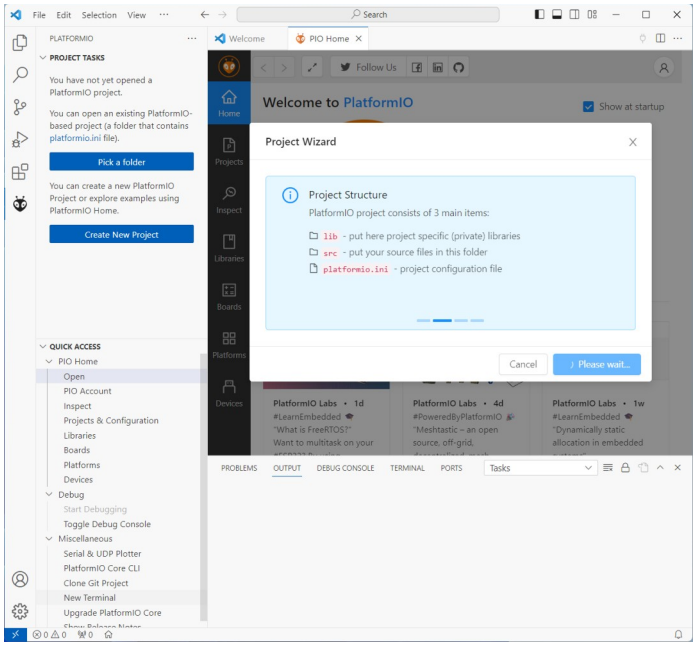


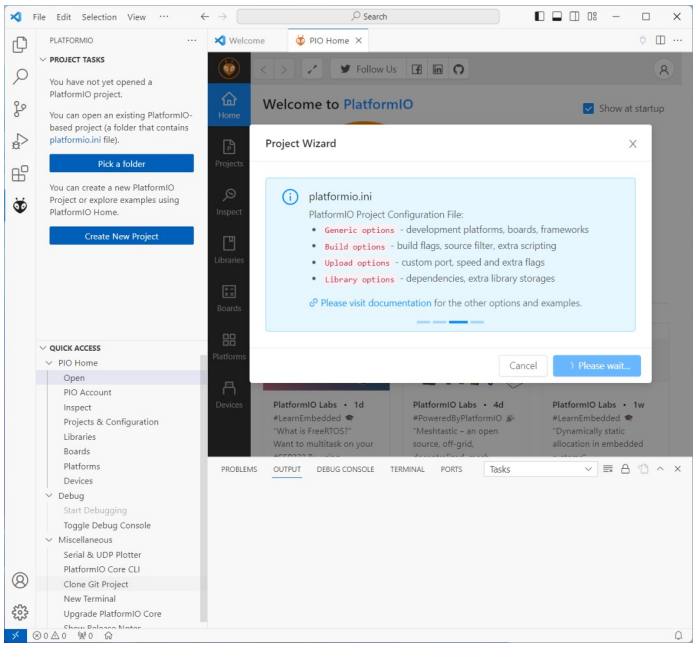
- Name your project “HelloWorld”.
- Under the board selection, choose "Espressif ESP32 Dev Module".
- It's recommended to use the default location for saving the project for ease of access.
- Click "Finish" to create the project.

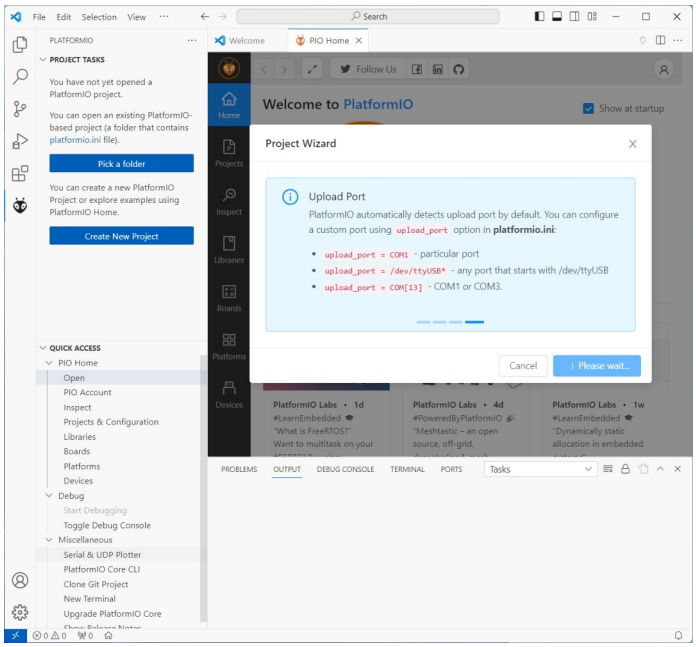


- Once you click "Finish," PlatformIO will start setting up your project. This process might take some time, especially for the first project, as PlatformIO installs necessary toolchains and dependencies.

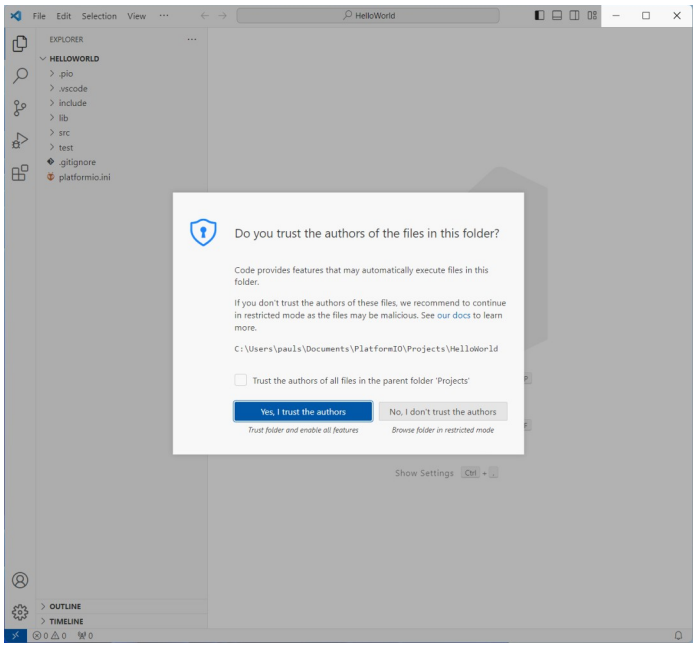


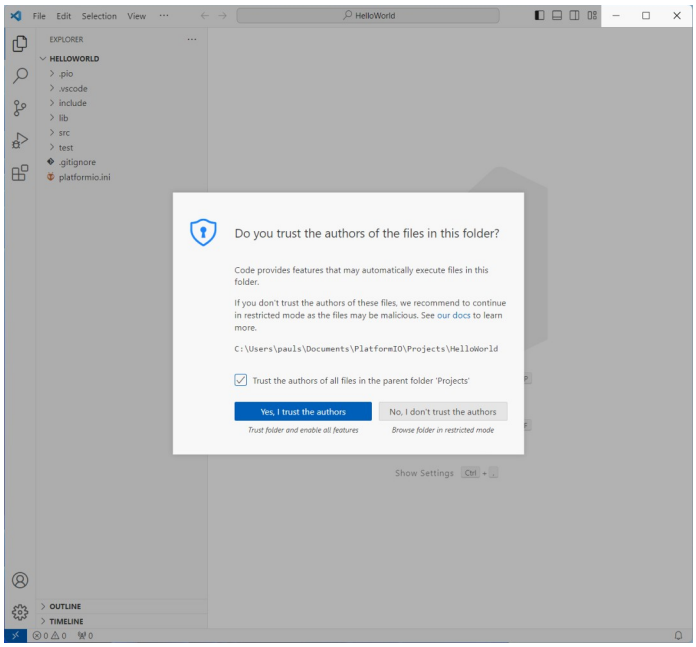






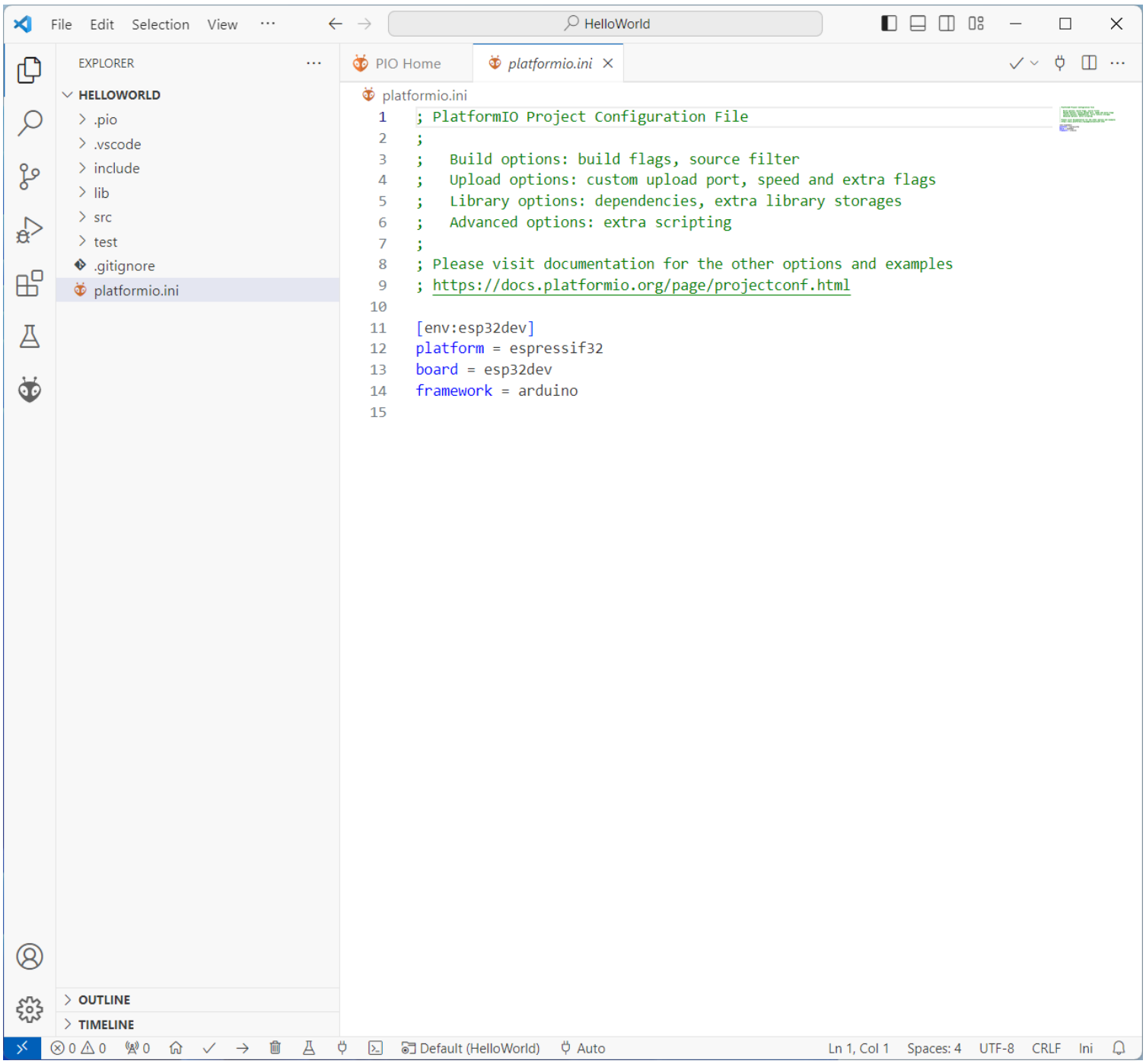
- During this process, if prompted with a message to "Do you trust the authors of the files in this folder?" tick the box and choose "Yes". This action allows Visual Studio Code to trust the files that PlatformIO is creating and using.





platformio.ini File:

- This is the configuration file for your project. It contains settings like the board type, framework, and other configurations specific to your project.

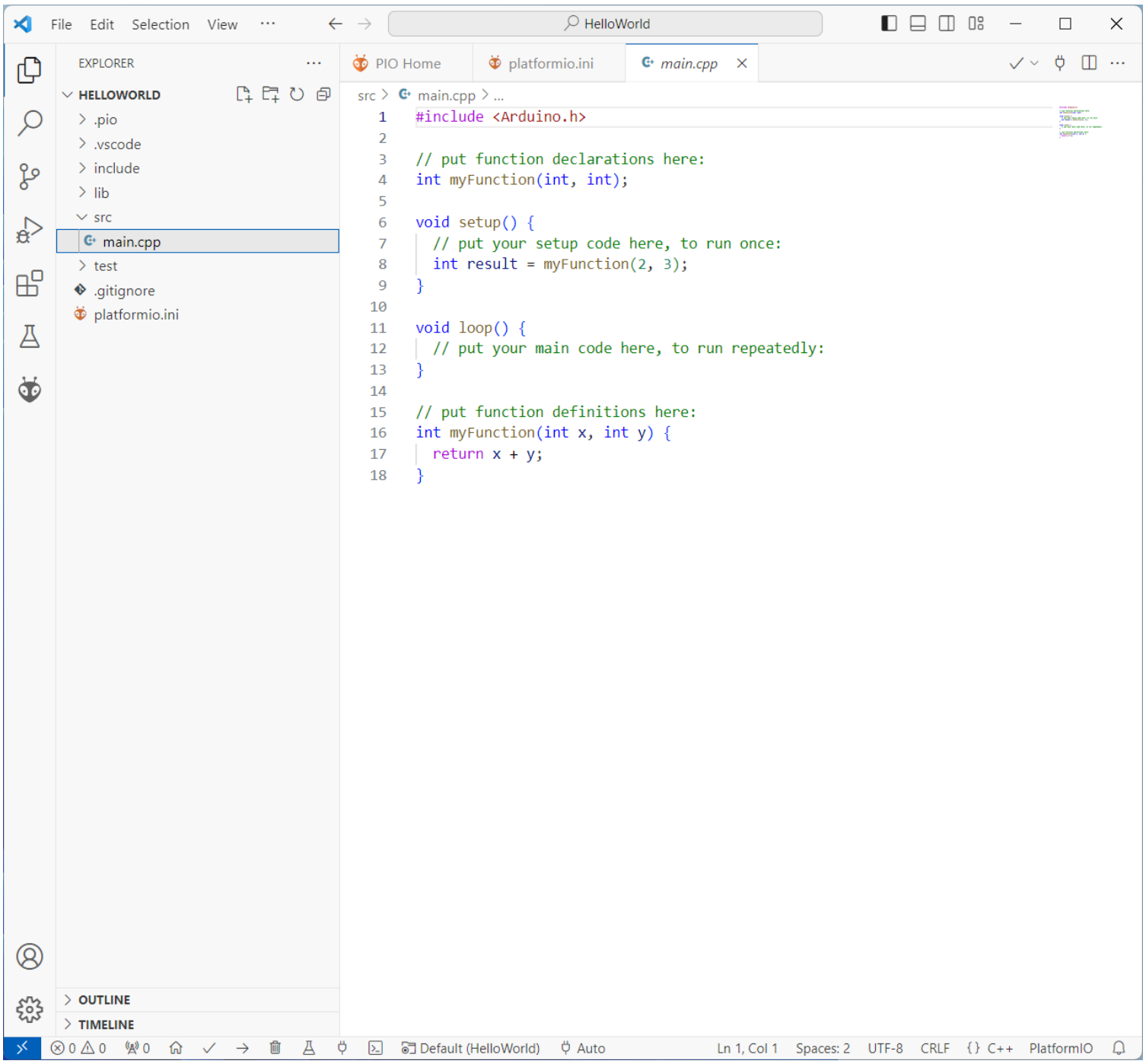


The Default platformio.ini File

- Take a moment to look at the default “platformio.ini” file. It will have pre-configured settings for the ESP32 board you selected. Understanding these settings is crucial for more advanced projects.

src Folder:

- This folder contains your source files. By default, PlatformIO creates a main.cpp file where you can start writing your program.



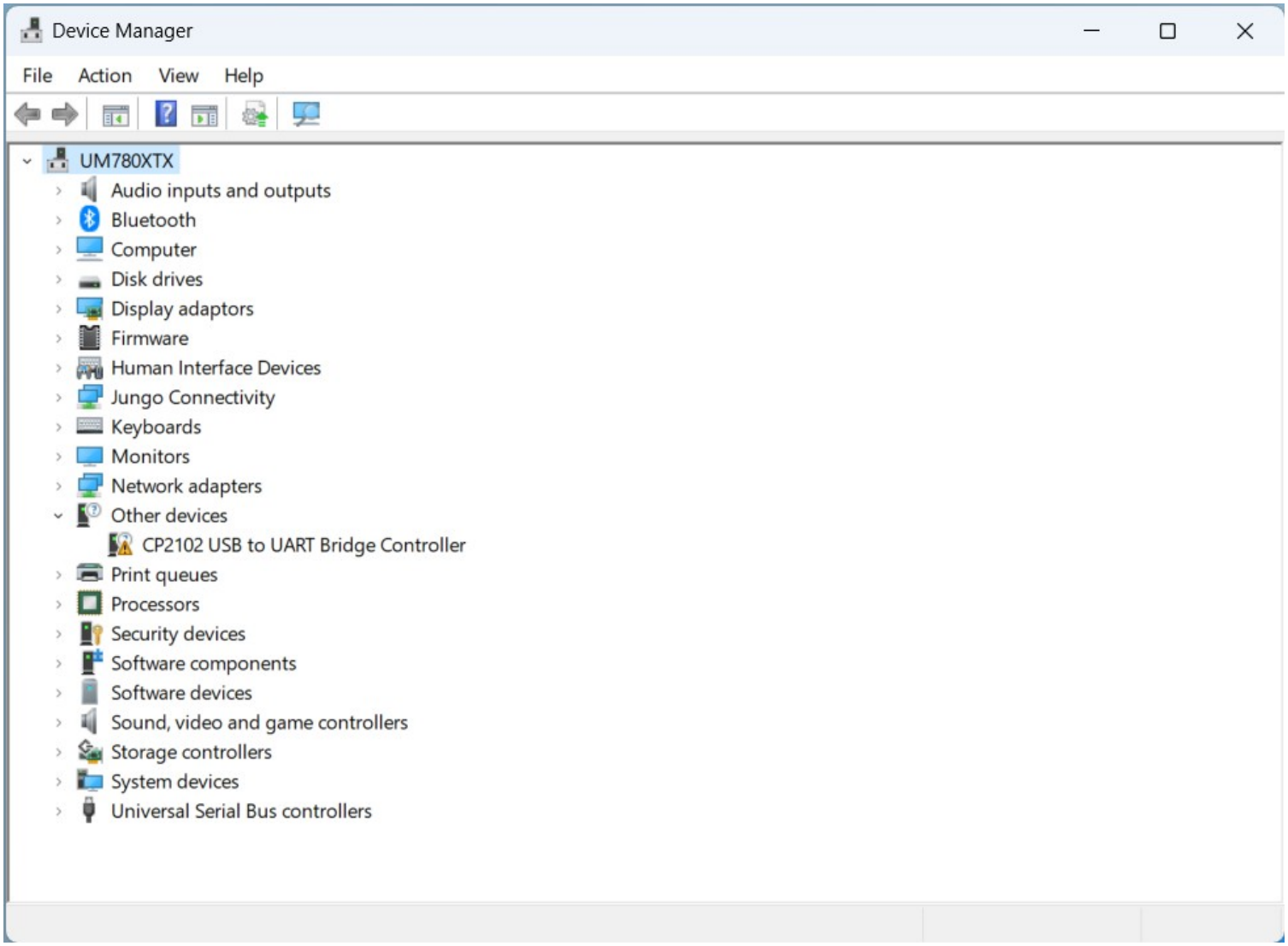
The Default main.cpp File

- PlatformIO automatically creates a “main.cpp” file in the “src” folder. This file is where you'll write the code for your ESP32 project.

Finding COM Port of the Device

To program the ESP32, you need to know its COM port number when it's connected to your PC. Follow these steps to find and set up the COM port:

1. Connect Your ESP32 Board:
   - Use a micro USB cable to connect the ESP32 board to your computer.
2. Open Device Manager:
   - On your Windows PC, search for and open "Device Manager".

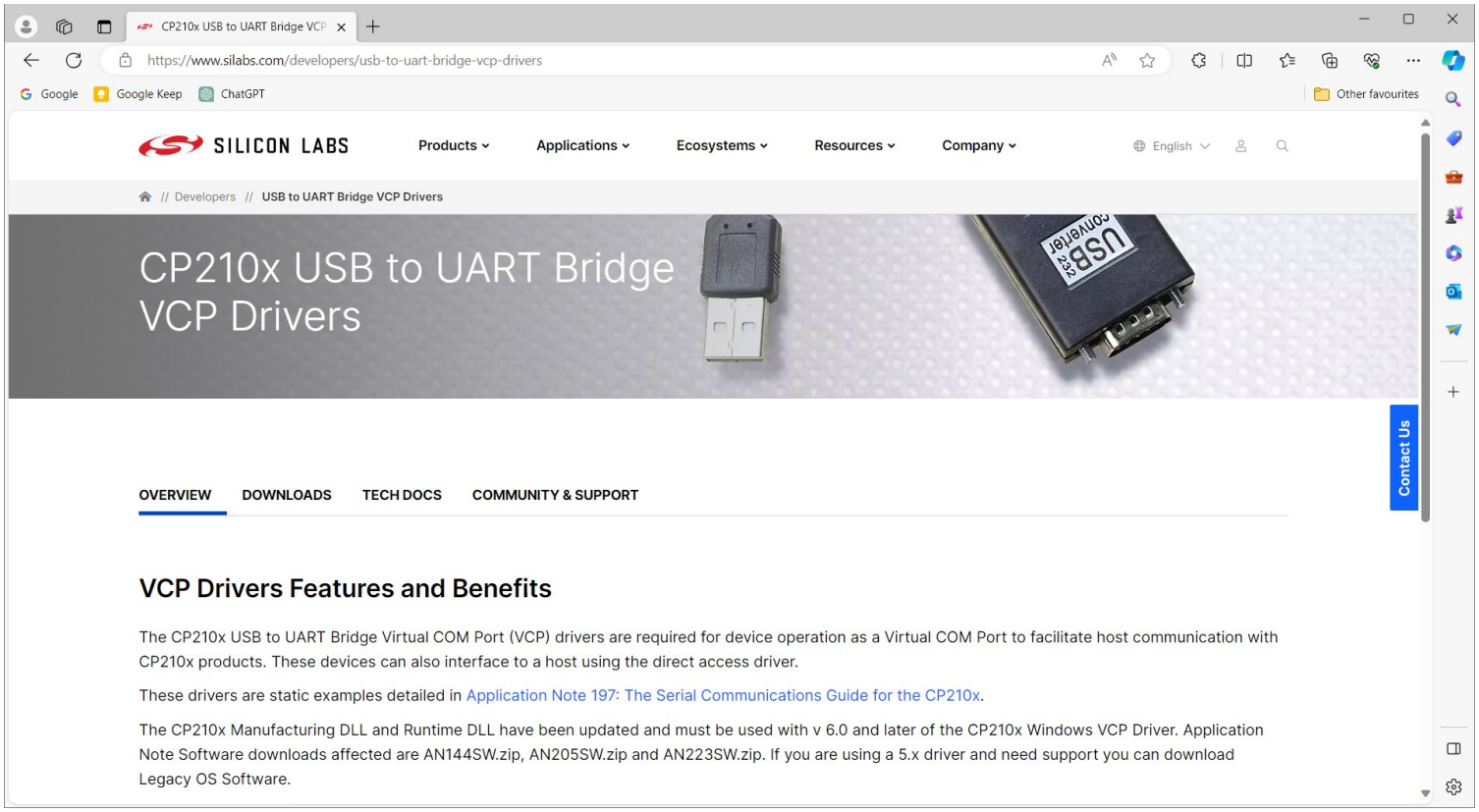


Check Device Manager

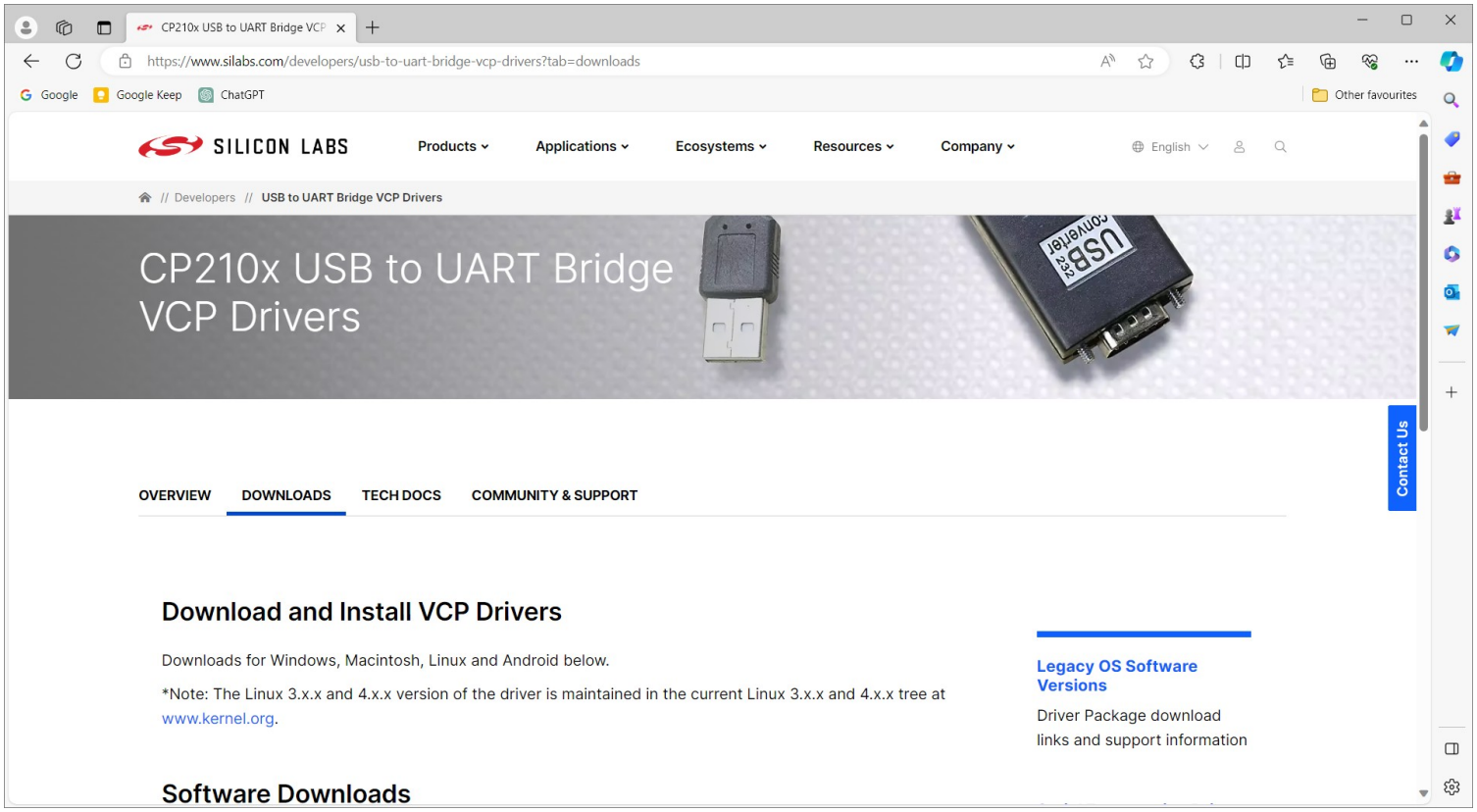
- Once in Device Manager, look under the "Ports (COM & LPT)" section.
- If you see a device listed as "Silicon Labs CP210x USB to UART Bridge (COMx)", where 'x' is a number, this is your ESP32's COM port. Make a note of this COM port number, as it will be used in your project settings. **Proceed to page 22 for further instructions**.
- If you notice a yellow exclamation mark (!) next to the UART controller, and it's listed under “Other devices”, this typically indicates a driver issue. In this case, follow the subsequent steps to install or update the necessary drivers.

Download Drivers:

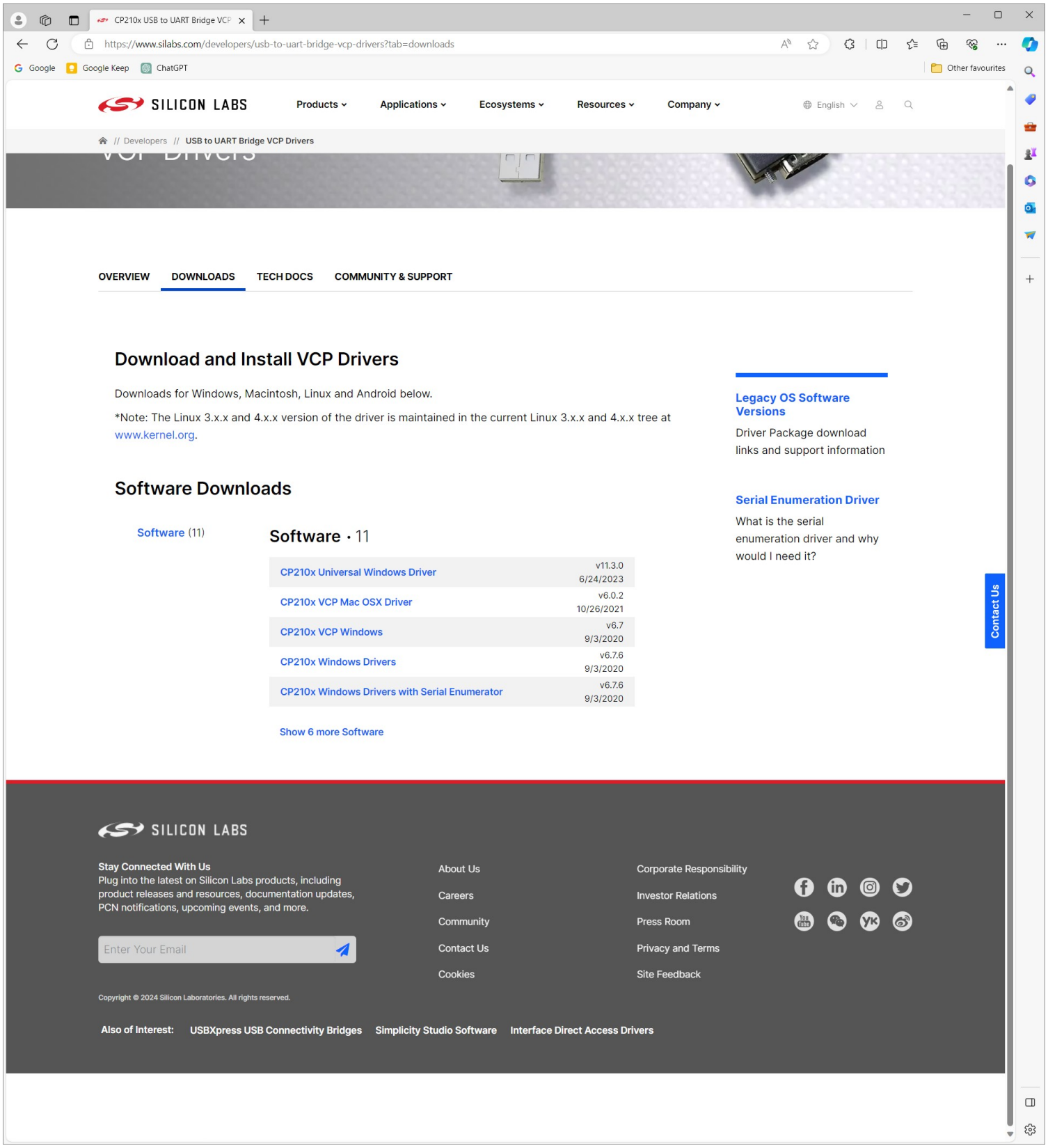
- Go to silabs.com/developers/usb-to-uart-bridge-vcp-drivers.



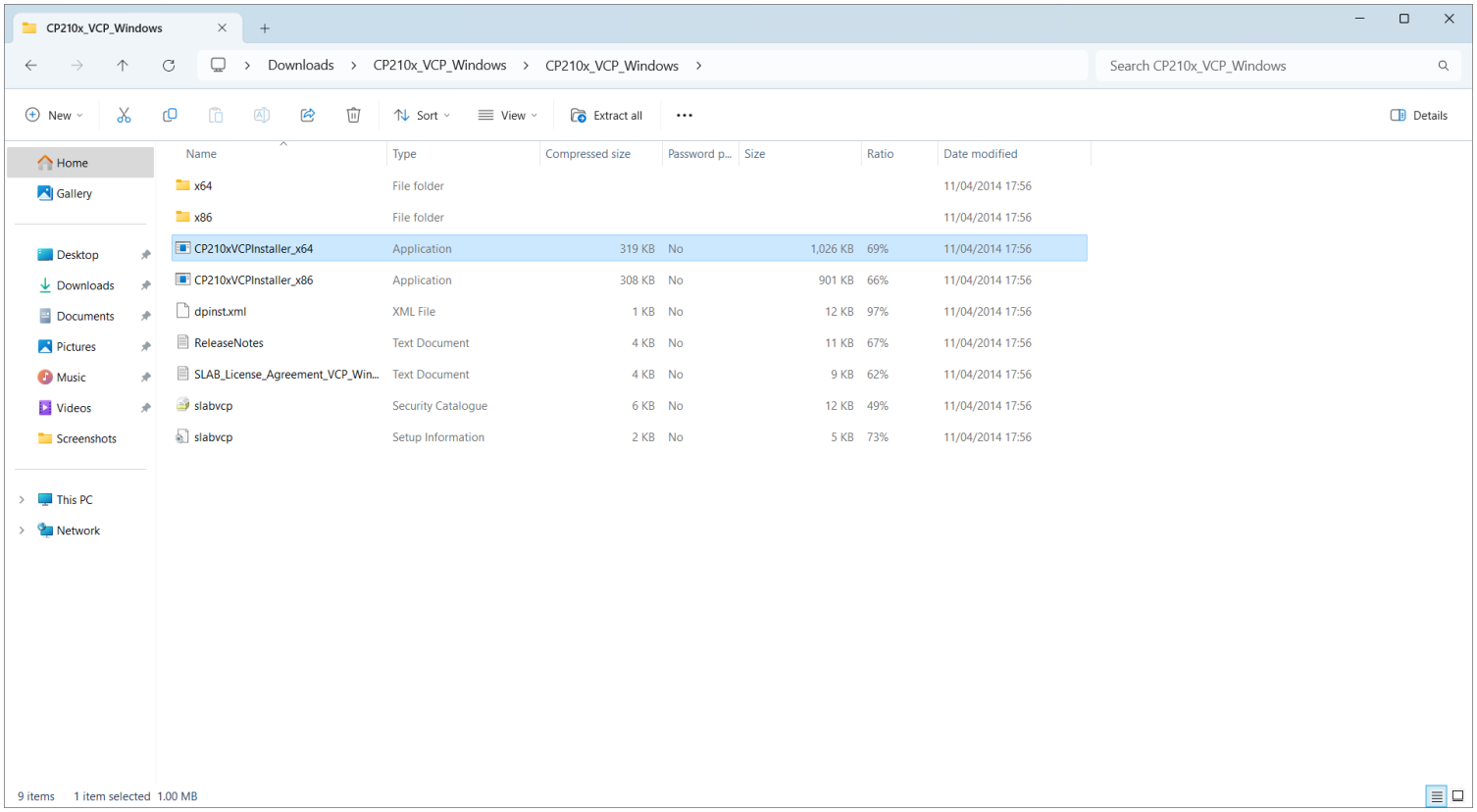
- Navigate to the 'Download' tab.



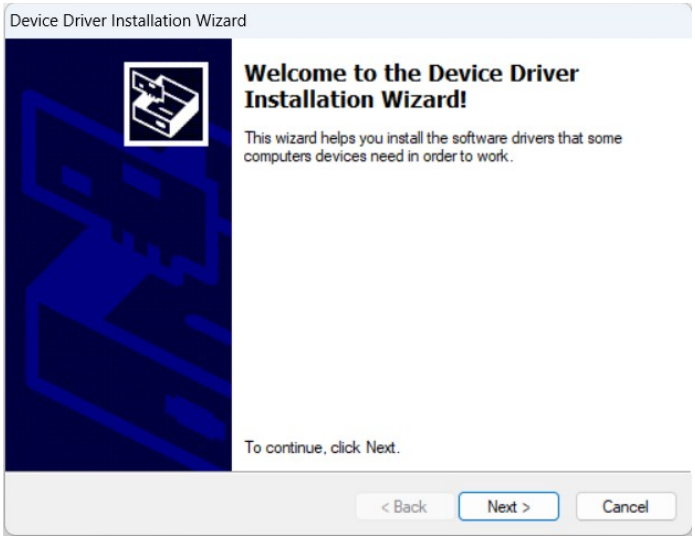
Download the “CP210x VCP Windows” driver.

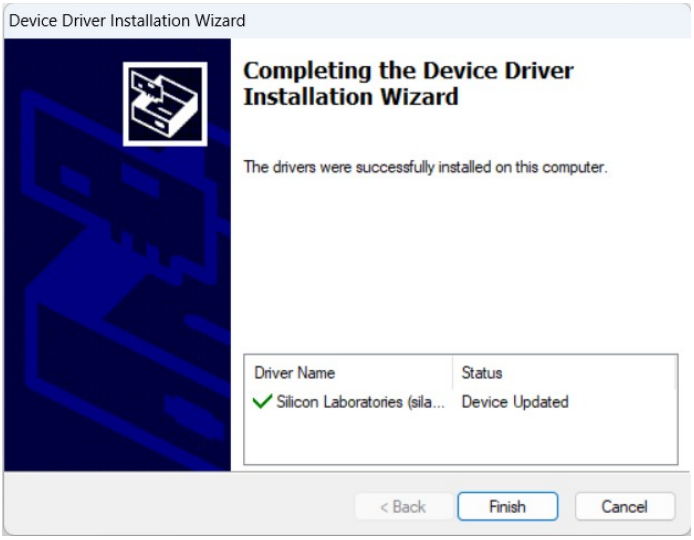


Extract the downloaded files.



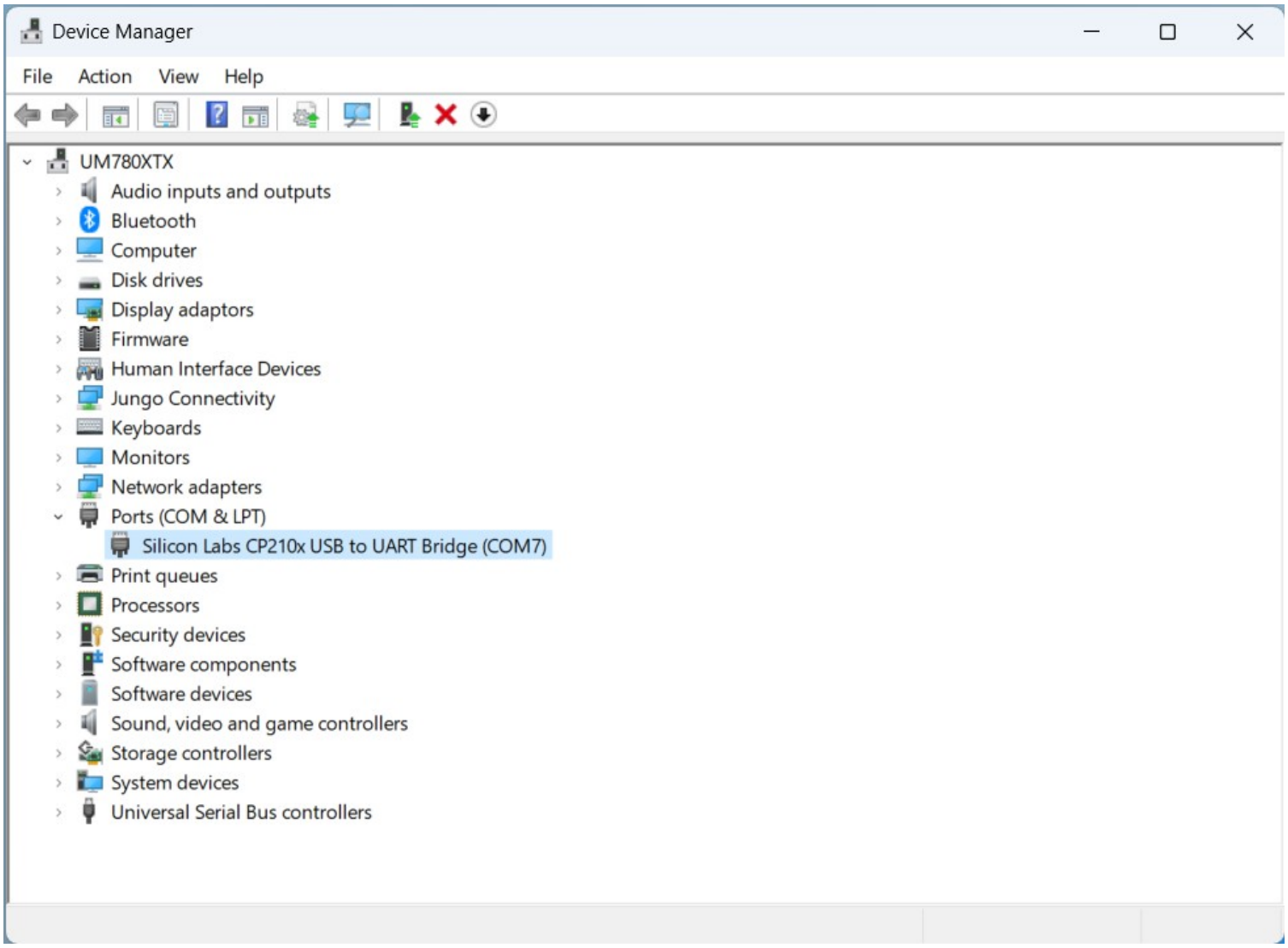
- Run the installer appropriate for your system (e.g., _x64 for Windows 11 64-bit).
- Follow the on-screen instructions to install the drivers.



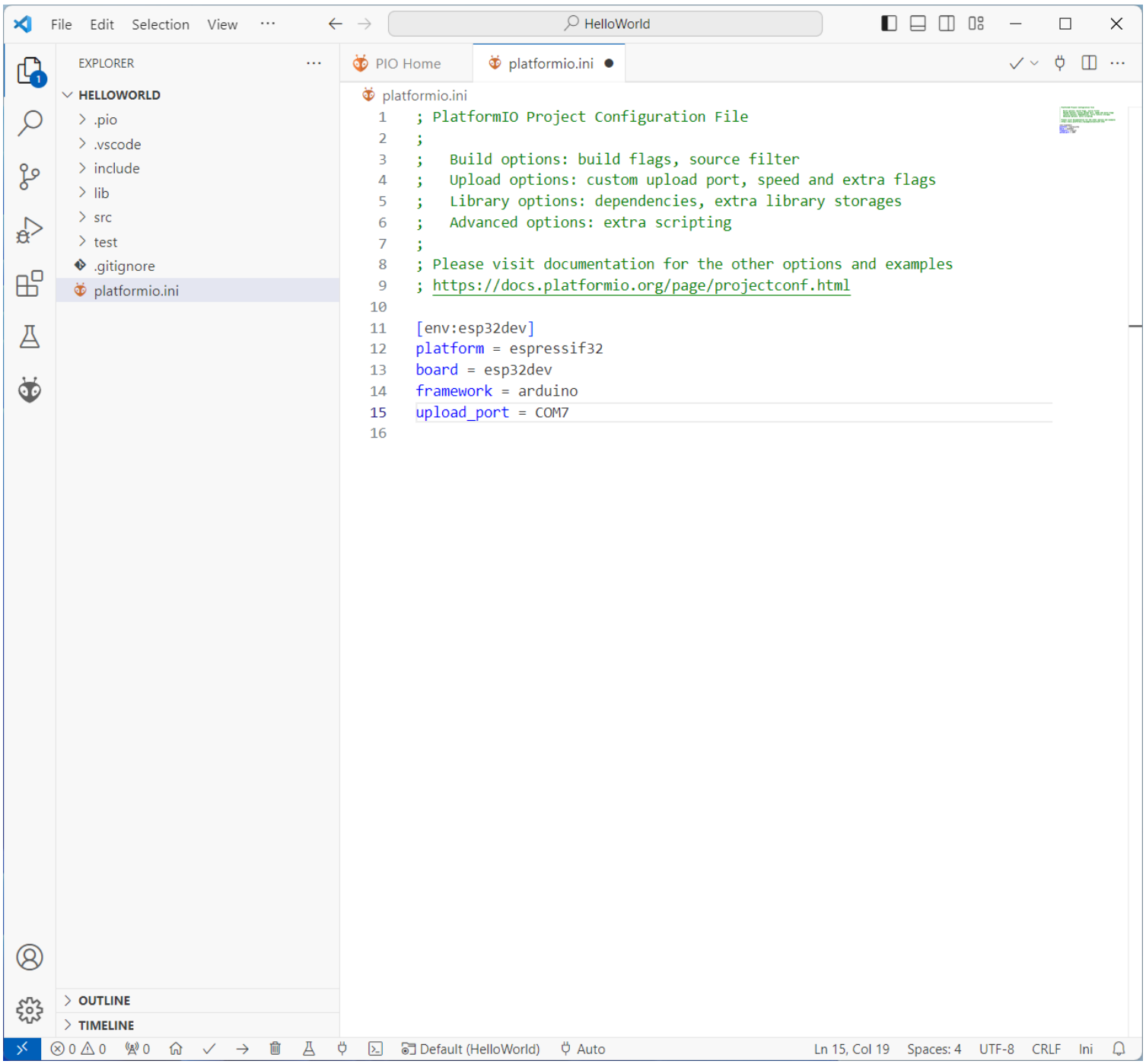


- After installation, reopen Device Manager.
- Now, it should show "Silicon Labs CP210x USB to UART Bridge (COMx)", where 'x' can be any number.

Make a note of the COM port number; this guide will use COM7 as an example.



Update PlatformIO Configuration.



- In your PlatformIO project, open the platformio.ini file.
- Add or update the line: upload_port = COM7 (replace COM7 with your actual COM port number).
- This ensures PlatformIO knows which COM port to use for uploading code to the ESP32.

Note: The COM port number (e.g., COM7) can vary depending on your machine and other connected devices. Always check the Device Manager to confirm the correct COM port assigned to your ESP32.

Updating Your First ESP32 Program

Now that your development environment is set up and your project is ready, let's update the default main.cpp code to create a simple program that controls an LED and uses serial communication. Follow these steps:

Modify the main.cpp Code

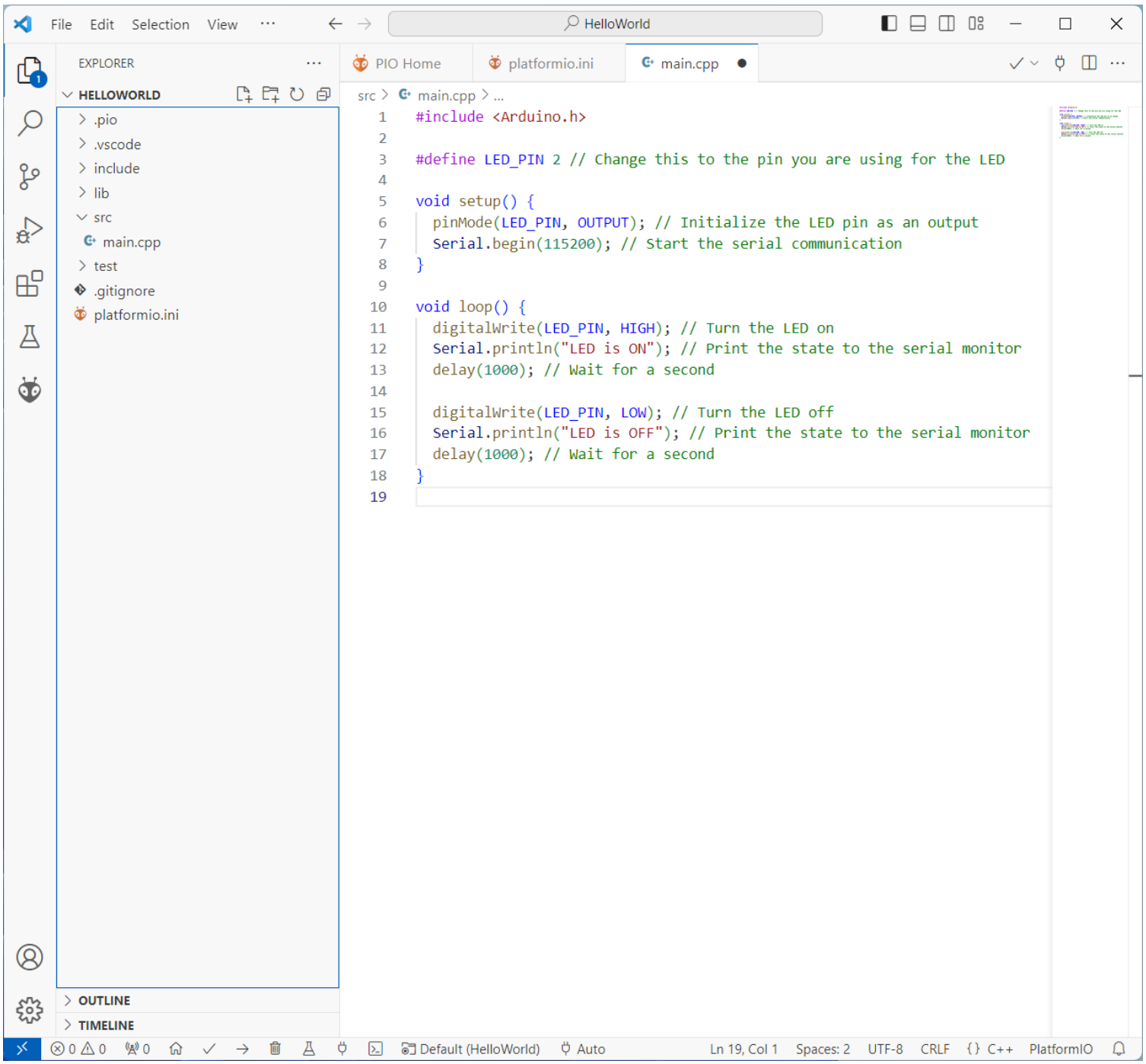
1. Open main.cpp:
   - In your PlatformIO project, navigate to the src folder and open the main.cpp file.
2. Replace the Code:
   - Copy the following code snippet:

```
#include <Arduino.h>

#define LED_PIN 2 // Change this to the pin you are using for the LED

void setup() {
  pinMode(LED_PIN, OUTPUT); // Initialize the LED pin as an output
  Serial.begin(115200); // Start the serial communication
}

void loop() {
  digitalWrite(LED_PIN, HIGH); // Turn the LED on
  Serial.println("LED is ON"); // Print the state to the serial monitor
  delay(1000); // Wait for a second

  digitalWrite(LED_PIN, LOW); // Turn the LED off
  Serial.println("LED is OFF"); // Print the state to the serial monitor
  delay(1000); // Wait for a second
}
```

- Paste it into your main.cpp file, replacing any existing content.

1. Note on LED Pin:
   - The #define LED_PIN 2 line sets the GPIO pin for the LED. Adjust this if your LED is connected to a different pin.
2. Copy Accurately:
   - To avoid syntax errors, it's important to copy and paste the code exactly as shown.

The screenshot provides a visual confirmation of how your IDE should look with the updated main.cpp code.



Verify the Updated Code in the IDE

- Once you have replaced the code, your IDE (Integrated Development Environment) should reflect these changes.
- Ensure that the code is correctly formatted and free of errors. The IDE will typically highlight any syntax errors or issues.

Compile and Upload the Program to the ESP32

Once you have updated your code, the next step is to compile and upload it to the ESP32 board. This process translates your code into machine language that the ESP32 can execute and then transfers it to the board.

Compiling and Uploading

1. Start the Process:
   - In Visual Studio Code, locate the arrow icon (→) at the bottom of the IDE. This is the upload button.
   - Click this button to start compiling your program. After compilation, PlatformIO will automatically begin uploading the program to your connected ESP32.
2. Entering Program Mode:
   - Most ESP32 boards will automatically enter program mode when uploading. However, in some cases, you may need to manually put the ESP32 into program mode.
   - This is typically done by holding down the "BOOT" button on your ESP32 while starting the upload process, then releasing it once uploading begins (See the **note** below).
3. Monitor the Upload:
   - Keep an eye on the console output in Visual Studio Code. It will display the progress of the compilation and upload
   - Once the upload is complete, you should see a message indicating a successful upload.
4. Troubleshooting:
   - If you encounter errors during this process, check your connections, ensure the correct COM port is selected, and that the correct board is configured in your platformio.ini file.
   - Revisit the section on finding the COM port if necessary, and ensure your ESP32 board is properly connected to your PC.

Note on Manual Programming Mode:

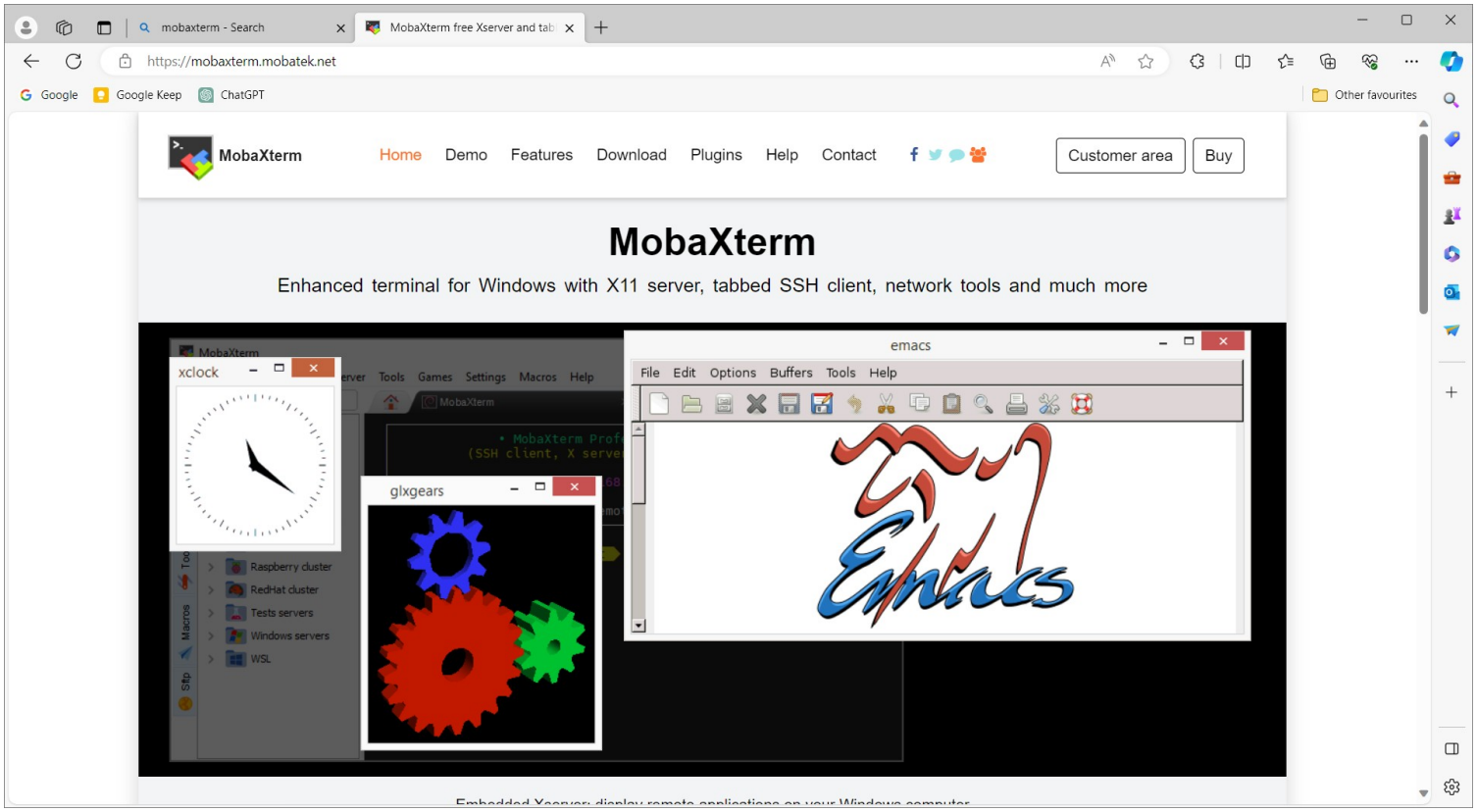
1. Some ESP32 boards might not automatically enter program mode during the upload process. If your board does not start programming automatically, you need to manually put it into program mode.
2. When to Act: Pay attention to the console output in Visual Studio Code. When you see the message "**Connecting.....**", this is your cue.
3. Action Steps:
   - Hold the BOOT Button: Press and hold the "BOOT" button on your ESP32 board.
   - Release After Connection: Once you see a change in the output message or indication of the programming process starting, release the "BOOT" button.
4. This manual step prompts the ESP32 to enter program mode, allowing the upload to proceed. If done correctly, the compilation and uploading process should continue successfully.

It’s important to follow these steps precisely to ensure successful programming of your ESP32 board, especially if your model does not support automatic programming mode.

Setting Up a Serial Terminal with MobaXterm

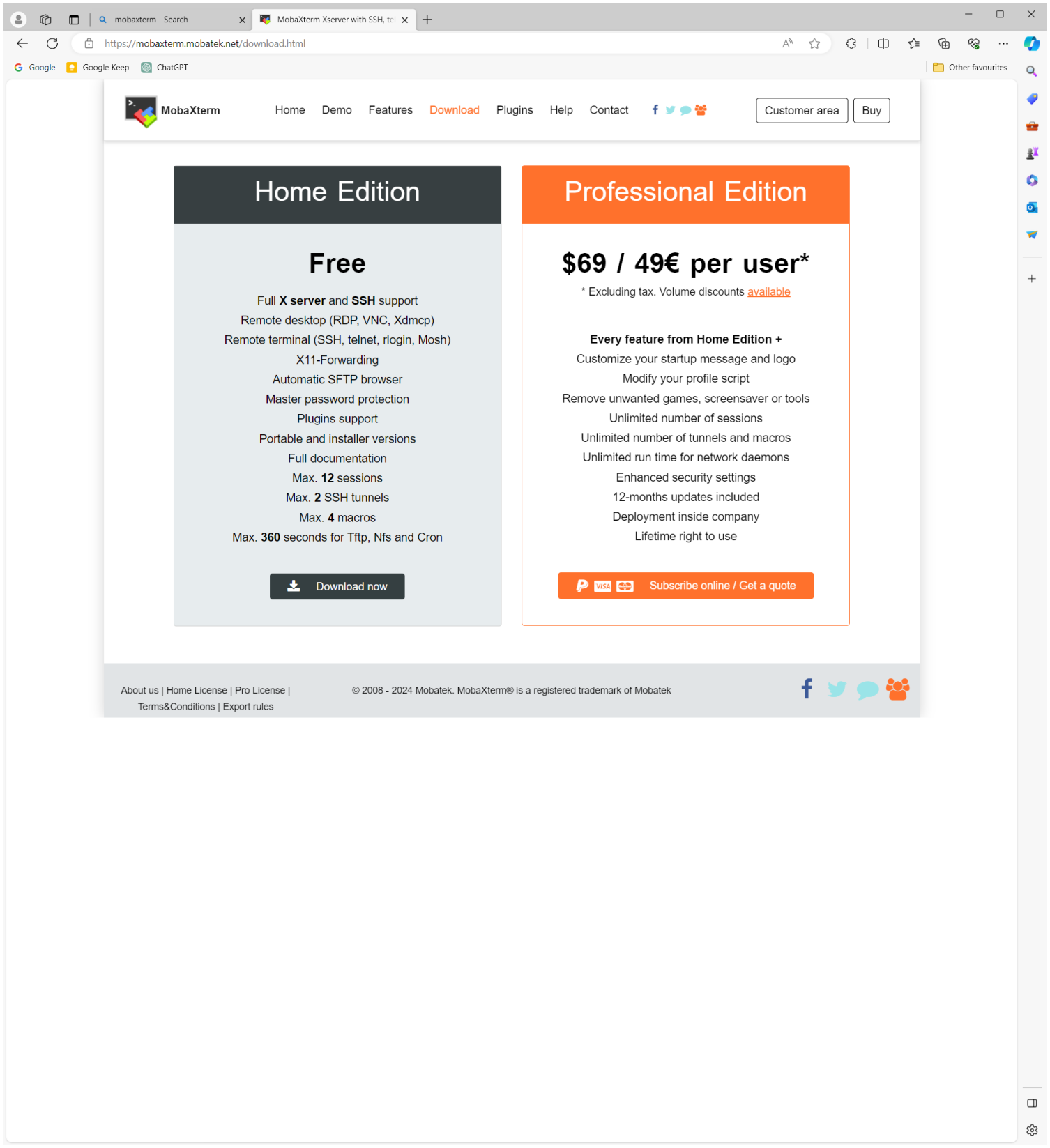
Communicating with your ESP32 via a serial terminal is crucial for debugging and observing your program's output. MobaXterm is a versatile terminal program that can be used for this purpose. Here’s how to set it up:

Download MobaXterm:

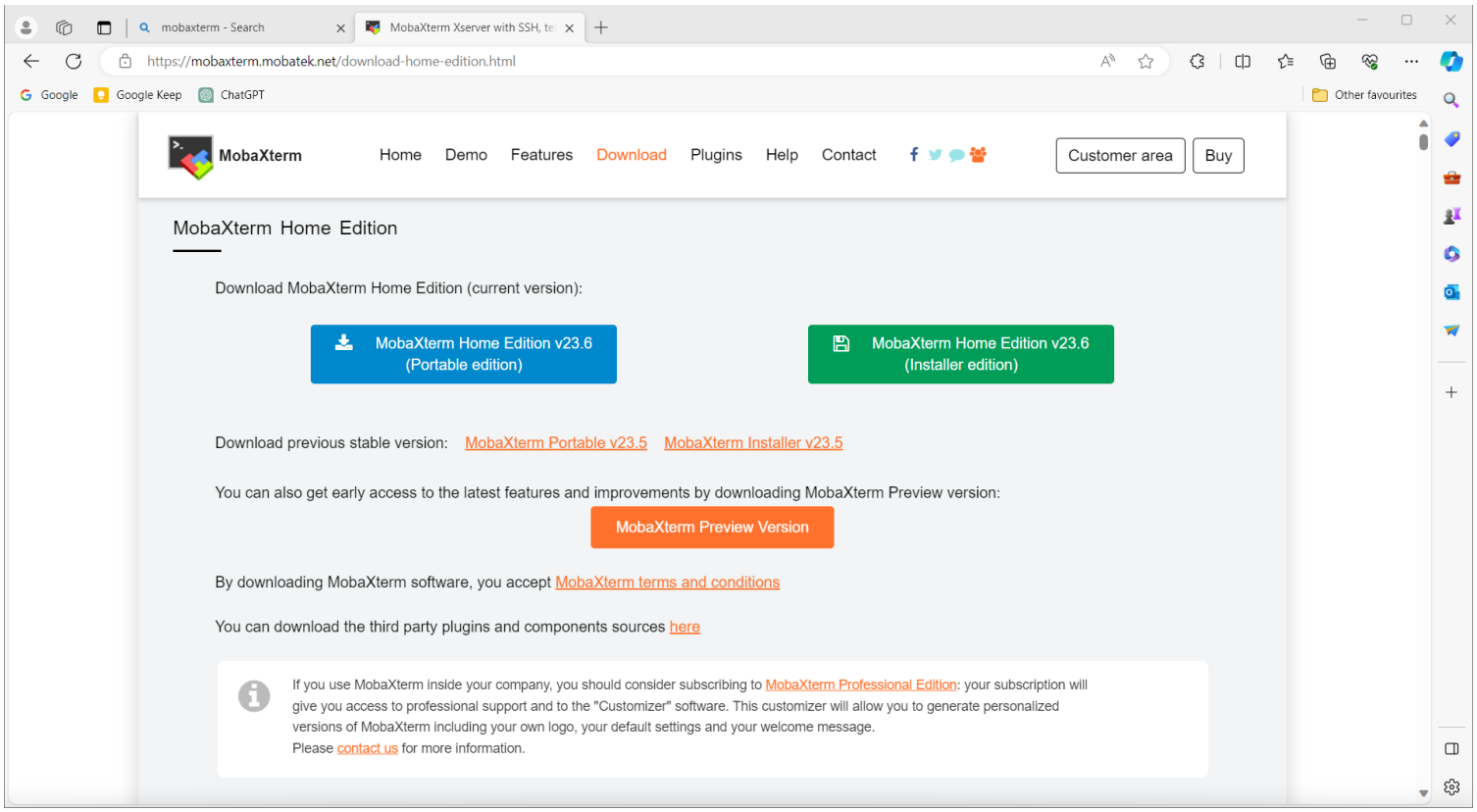


- Visit the MobaXterm website mobaxterm.mobatek.net.

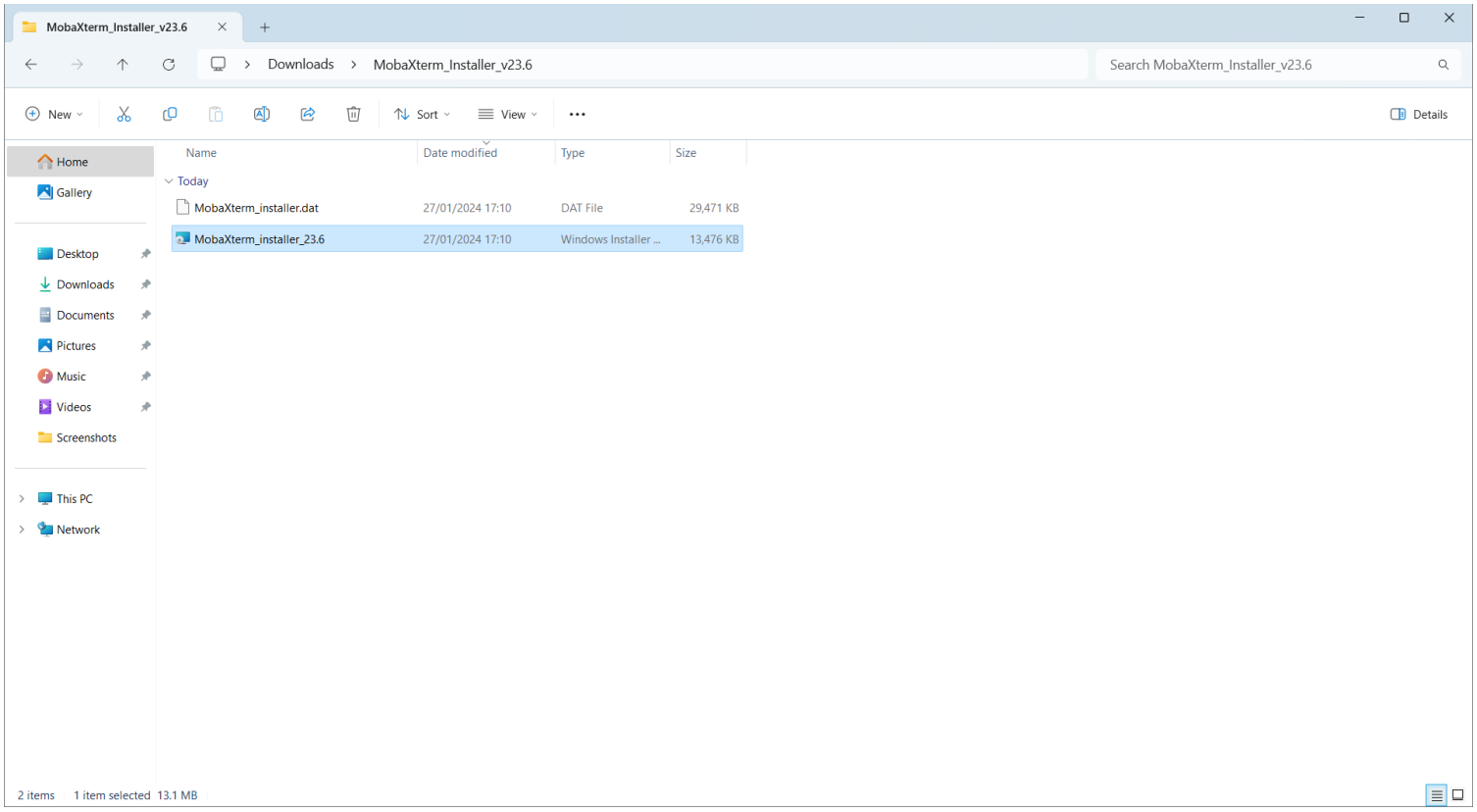
- Navigate to the download page.
- Select Download now (Free).



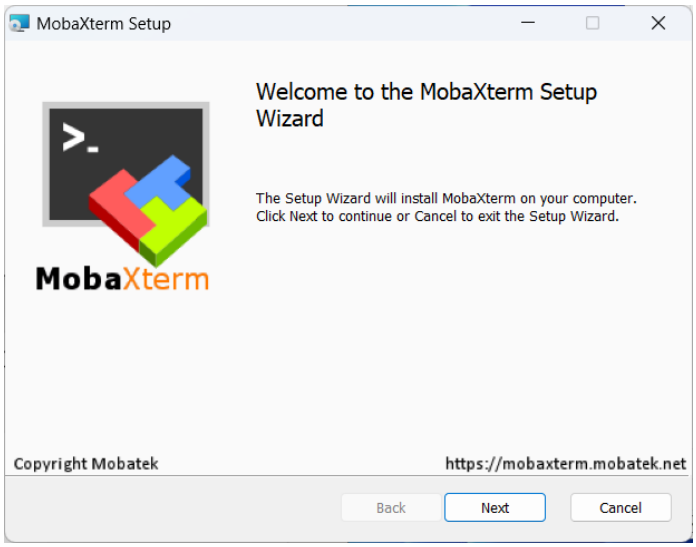
- Select and download the "Installer Edition" of MobaXterm.

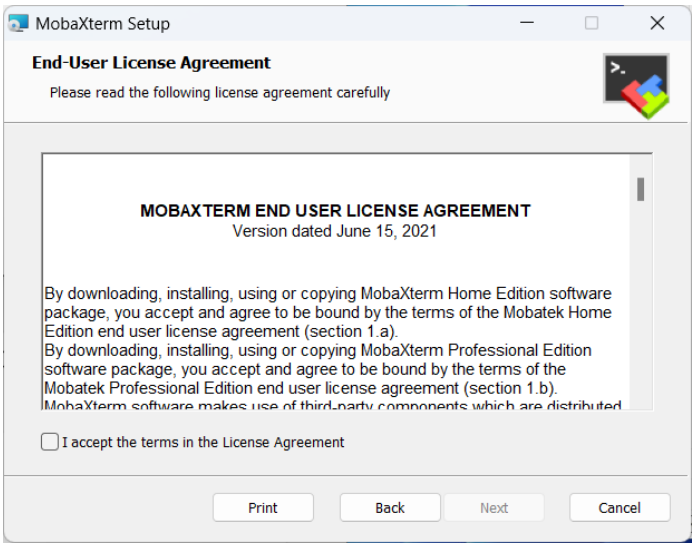


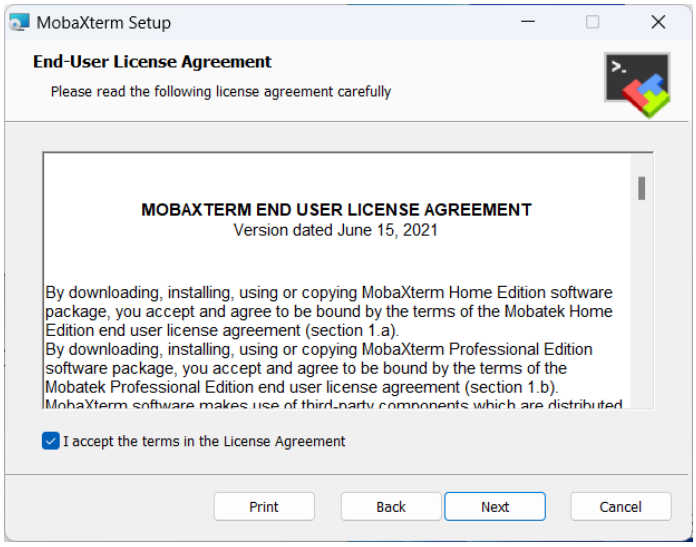
Locate the downloaded file and extract it & run the installer.

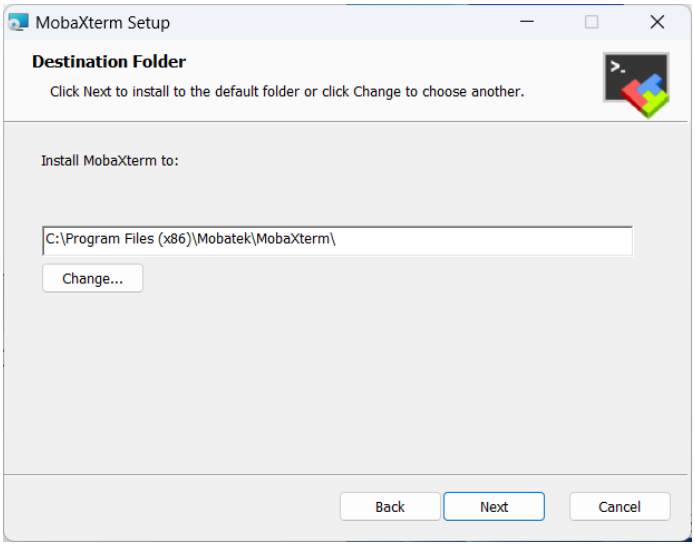


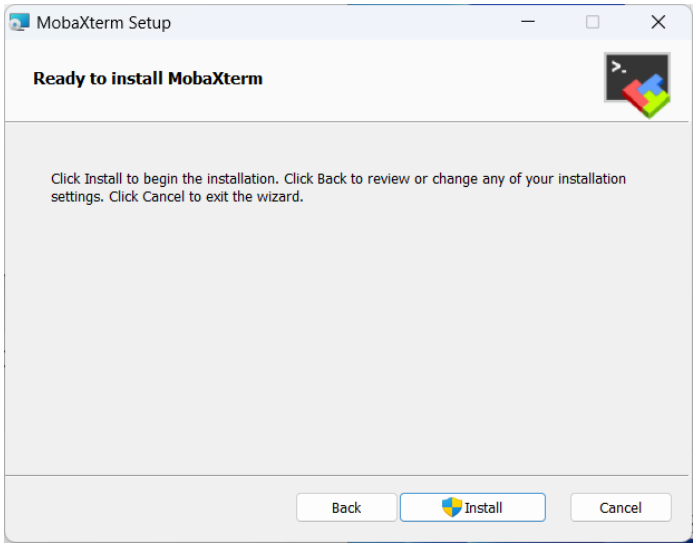
Follow the installation steps, clicking 'Next' through the process to complete the installation.

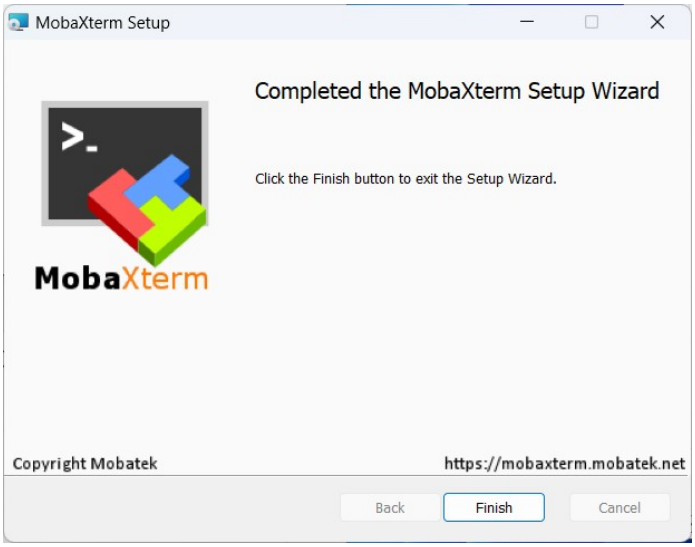




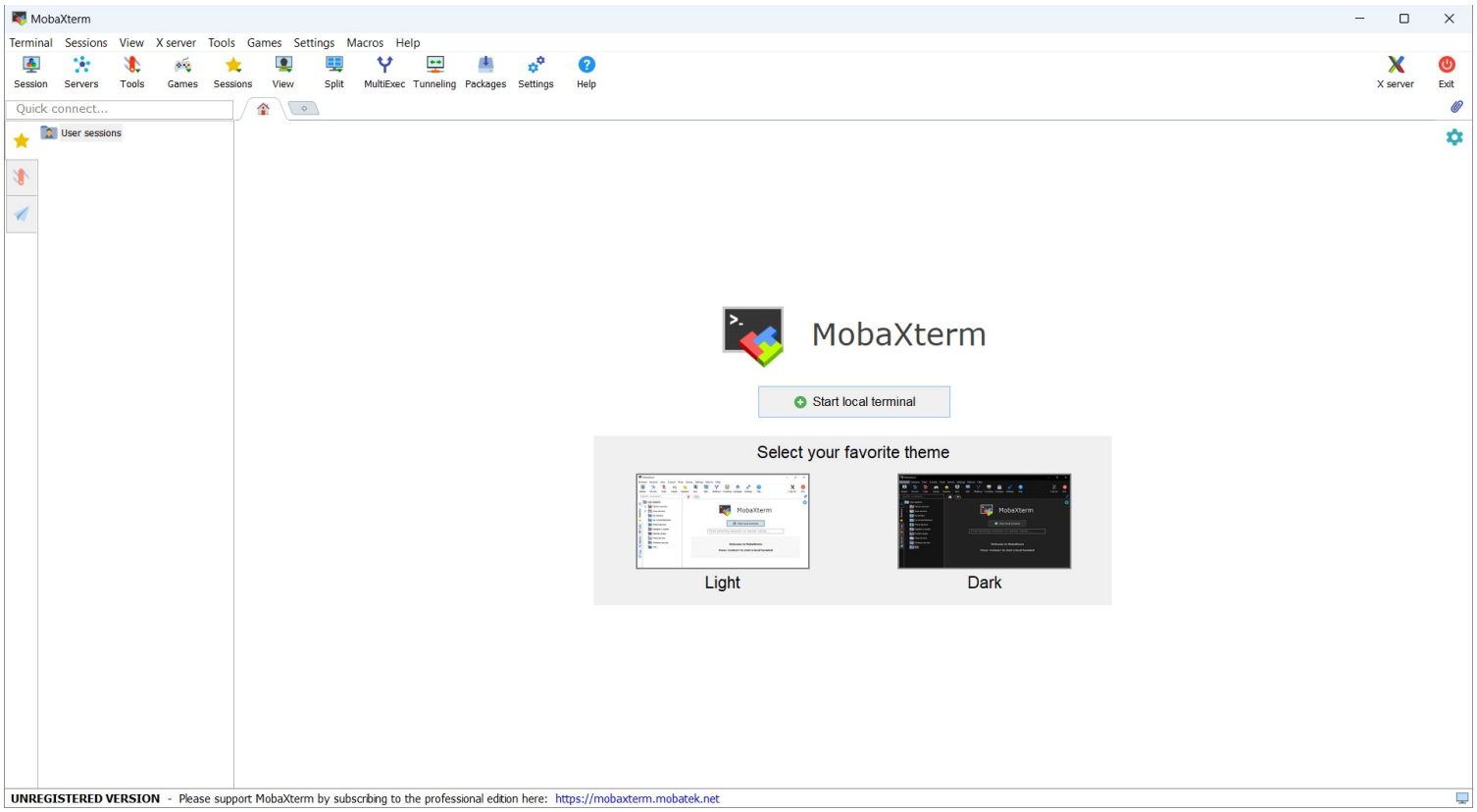




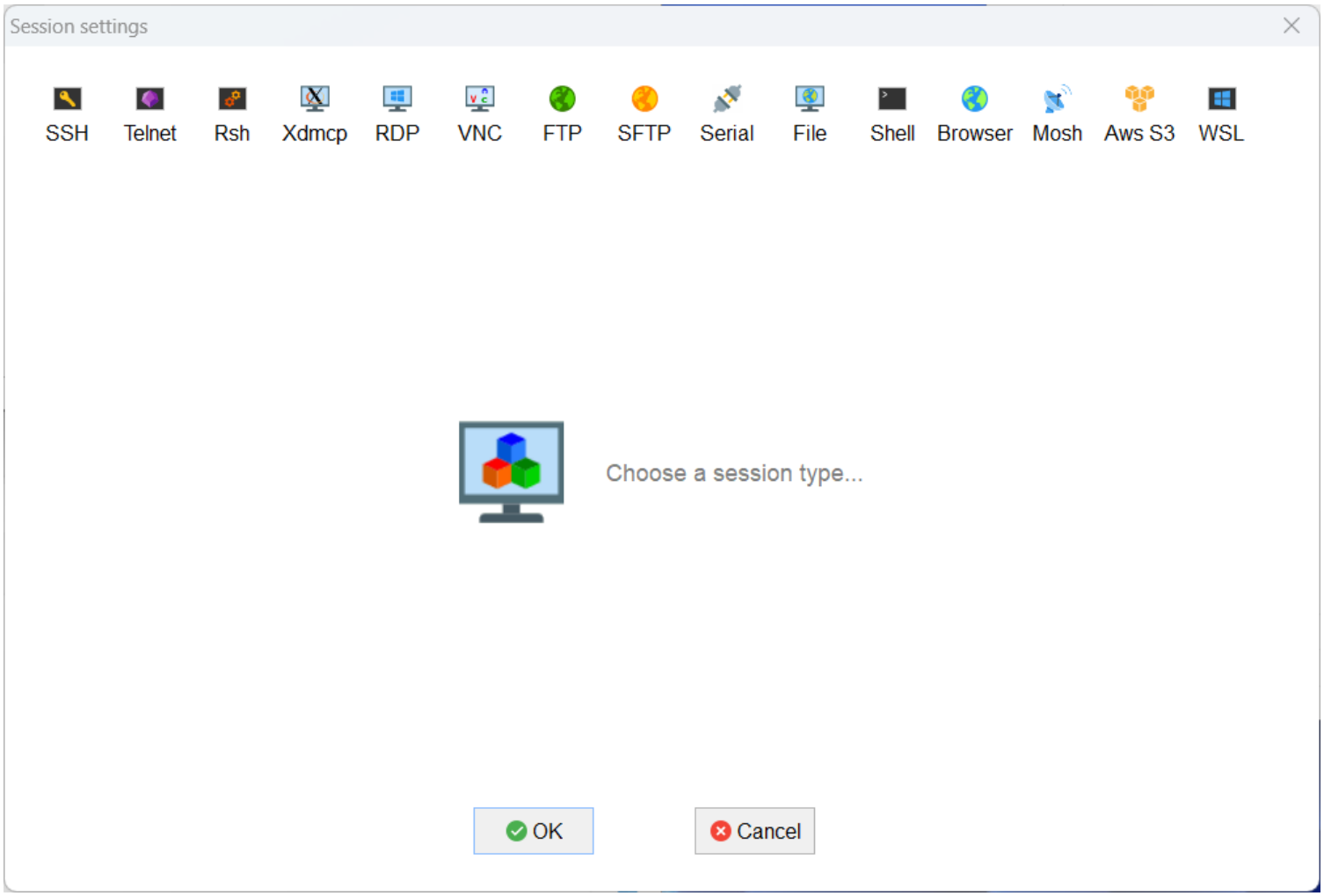




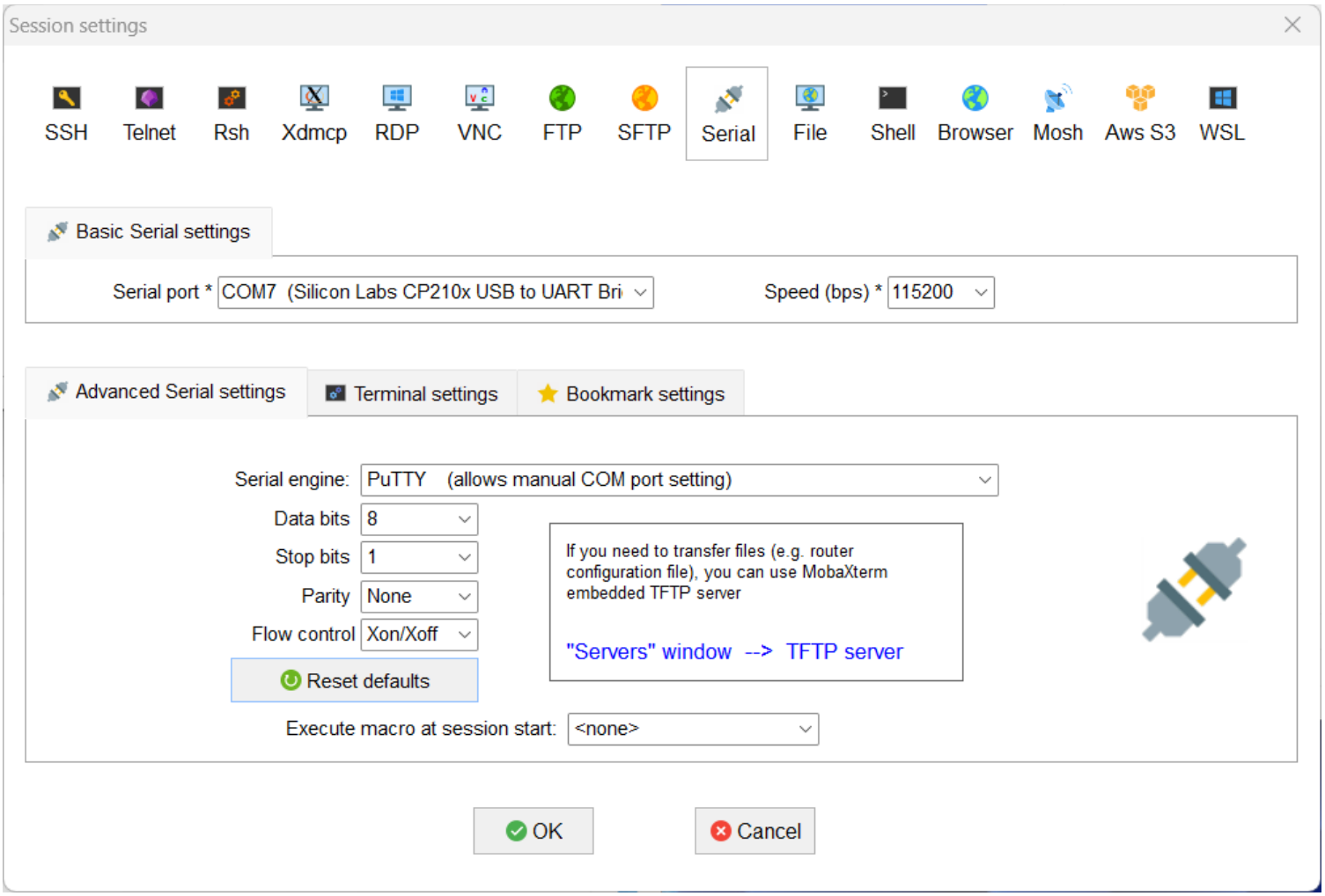
1. Select a Theme (Optional):
   - Upon first launch, you may choose a theme for MobaXterm. This is optional and based on personal preference.
2. Open a New Session:
   - Click on the 'Session' button in the top left corner of MobaXterm.



- Choose the 'Serial' option.



1. Configure Serial Connection:
   - Configure the serial settings as follows (you can refer to the provided screenshot for guidance):
     1. Serial port: Select the COM port that your ESP32 is connected to (e.g., COM7). Remember, this will vary based on what you found in the Device Manager.
     2. Speed (baud rate): Set this to 115200.

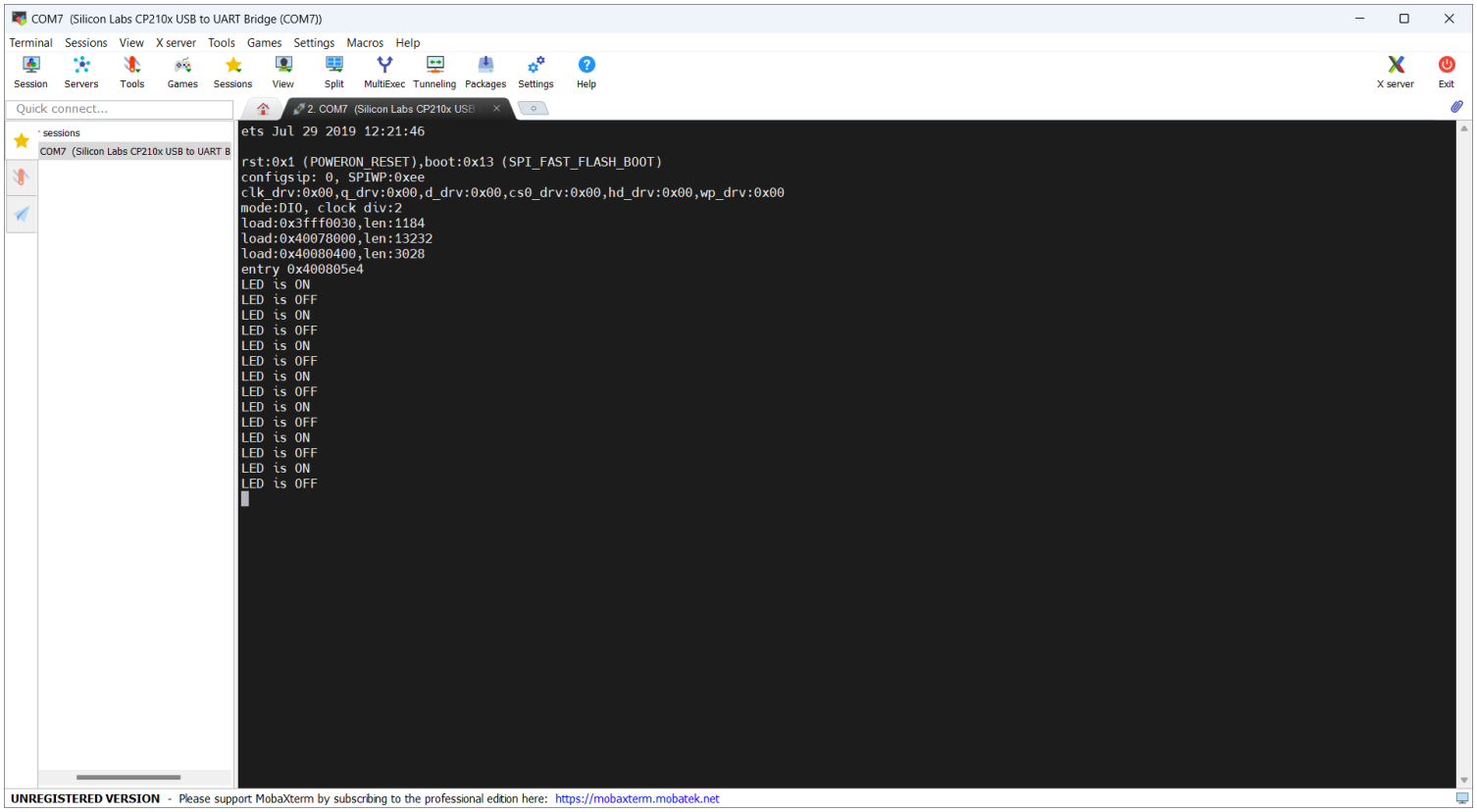


Connect to ESP32:

- Click 'OK' to close the settings window and establish the connection with your ESP32 through the selected COM port.

By following these steps, you will have MobaXterm installed and configured, ready to interface with your ESP32 for serial communication.

Once the serial monitor is open and connected to the correct COM port, you will be able to see real-time outputs from your ESP32. If your setup and code are correct, you should see the "LED is ON" and "LED is OFF" messages being printed in the serial monitor every second.



Setting Up the LED Circuit with ESP32

Now that you have successfully programmed the ESP32 and verified its output through the serial monitor, the next step is to set up the physical circuit for the LED.

Preparing the Circuit

1. Close the Serial Terminal Program:
   - Ensure that MobaXterm or any other serial terminal program you're using is closed.
2. Power Down the ESP32:
   - Before beginning any work on the circuit, first disconnect the USB cable from your ESP32. This precaution is crucial for safety and to prevent any electrical issues.
3. Gather the Components:
   - For this setup, you will need an LED, a resistor (150-200 Ohms), the ESP32 board, a breadboard, and jumper wires.
4. Connect the Resistor and LED:
   - Insert the LED into the breadboard.
   - Connect one leg of the resistor to the anode (longer leg) of the LED, ensuring the other leg of the resistor is in a separate row on the breadboard.
5. Wire the Circuit to ESP32:
   - Connect the free leg of the resistor to GPIO pin 2 on the ESP32, as defined in the code.
   - Connect the cathode (shorter leg) of the LED to a GND pin on the ESP32 using a jumper wire.
6. Double-Check Connections:
   - Carefully check that all connections are secure and correctly positioned as per your circuit design.

Testing the Circuit

- After setting up the circuit, reconnect the USB cable to your ESP32 and to your computer.
- Open your serial terminal program again and, if necessary, re-upload the program to the ESP32.
- Watch the LED. It should blink in sync with the "LED is ON" and "LED is OFF" messages displayed in the serial monitor.

Note on Programming the ESP32

- Close Serial Terminal Before Programming:
  - It is important to close the serial terminal program, such as MobaXterm, before you start programming the ESP32. An open serial connection can interfere with the programming process. Ensuring the serial terminal is closed will facilitate a smooth and successful upload of new code to the board.

This hands-on experience with setting up a basic LED circuit controlled via the ESP32 lays the groundwork for more complex projects in the future.