

## Creating a robot program for the ESP32

To create a robot program for the ESP32 you need to follow some steps to get a sample project set up. It's pretty straight-forward, and once you follow the steps, then nothing special needs to be done.

### Create a project with PlatformIO

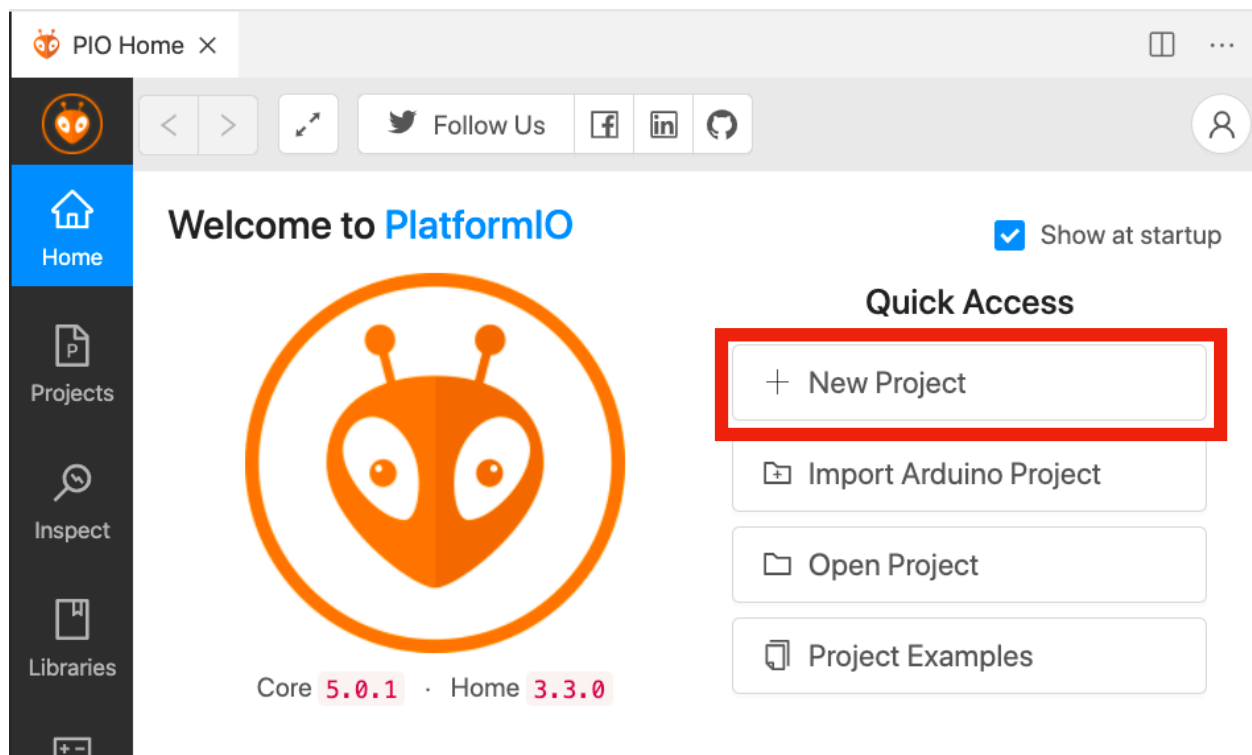
#### New window or workspace

Create a project in the desired directory using PlatformIO (the alien head tab in Visual Studio Code). Ideally there should be only one project per window, so if you have a project open, create a new window using the New Window option in the File menu (or type Shift-CMD-N on a Mac or Shift-Windows-N on a PC). If you see an empty tab in the new window called Untitled, just close it.

---

#### Create a new project

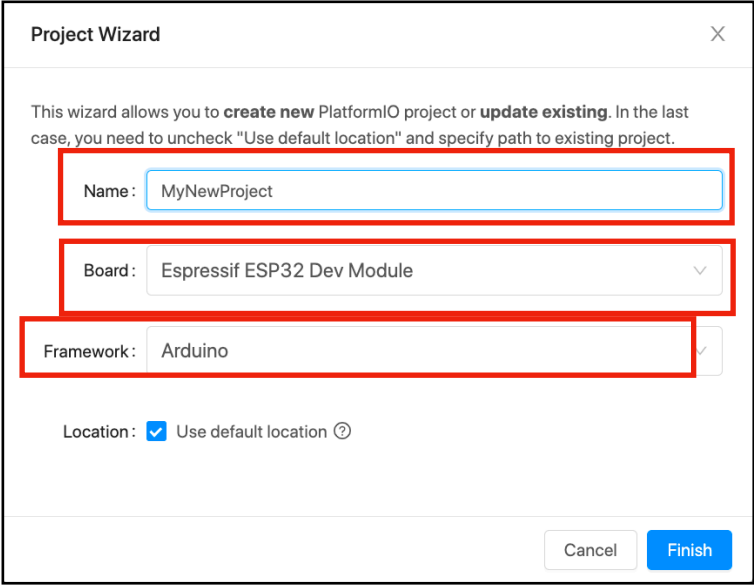
Click the “+ New Project” button in the PlatformIO interface as shown.



### Fill in the Project Wizard screen

Pick a name for your new project and fill it in the Name field. For the Board type, choose **Espressif ESP32 Dev Module**. You can get to it quickly by typing in “ESP32 Dev”. This sets the board type that you are programming.

If you leave the “Use default location” box checked if you are happy with the project being created in the Documents/PlatformIO/Projects folder. In my case where I have a lot of projects for a number of classes, I leave the box unchecked and have an RBE1001 folder inside the PlatformIO/Projects folder.



The screenshot shows the 'Project Wizard' dialog box with the following fields and options:

- Name:** MyNewProject
- Board:** Espressif ESP32 Dev Module
- Framework:** Arduino
- Location:** ☒ Use default location ⓘ

At the bottom right, there are 'Cancel' and 'Finish' buttons.

## Edit the platformio.ini file

The platform.ini file has a number of parameters that describe the hardware and software configuration for your project. Generally it is set up automatically when you complete the Project Wizard form. But for adding the RBE1001 software library to your project, you need to add some lines to the end of the file so that it looks like this:

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
lib_deps = wpiroboticsengineering/RBE1001Lib
          ESP32WifiManager
          ESP32AnalogRead
          ESP32Servo
          ESP32WifiServer
          ESP32Encoder
```

## Add the include declaration

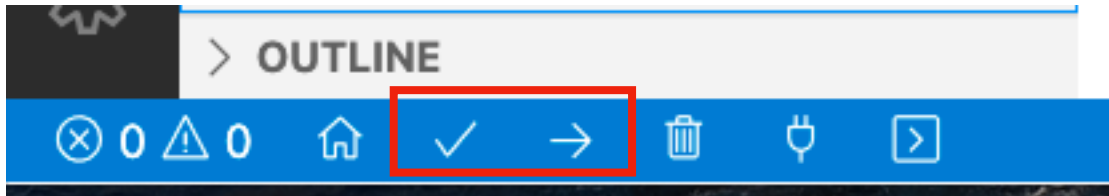
To include the declarations for the RBE1001 components that your program is using, you need to add the following “#include” line to the start of your main.cpp file in the “src” folder.

```
#include <RBE1001Lib.h>
```



## Build and download your project

You can build your project by clicking on the Checkmark icon on the bottom left of the VSCode window. This will compile all the .cpp files that make up your project and verify that the code you wrote has correct syntax. Note: correct syntax doesn't necessarily mean that the program will operate correctly.



You can then upload the program to your robot by clicking on the right-pointing arrow next to the checkmark. This will also build the project, then send it to the ESP32.

## A simple sample program to drive motors

Here's the shortest sample program that will drive the motors and verify that the encoders are correctly wired. **Be aware that there is a compatibility issue between the V2 and V3 red shim boards. The left motor definitions shown here have been tested and work with the V3 board. For the V2 board, the left motor encoders MOTOR\_LEFT\_ENCB and MOTOR\_LEFT\_ENCA should be switched.**

```
#include <Arduino.h>
#include <RBE1001Lib.h>

Motor left;
Motor right;

void setup() {
    Serial.begin(9600);
    Motor::allocateTimer(0);
    left.attach(MOTOR_LEFT_PWM, MOTOR_LEFT_DIR, MOTOR_LEFT_ENCB,
MOTOR_LEFT_ENCA);
    right.attach(MOTOR_RIGHT_PWM, MOTOR_RIGHT_DIR,
MOTOR_RIGHT_ENCA, MOTOR_RIGHT_ENCB);
}

void loop() {
    left.setSpeed(120);
    right.setSpeed(120);
    delay(1000);
    left.setSpeed(0);
    right.setSpeed(0);
    delay(1000);
}
```