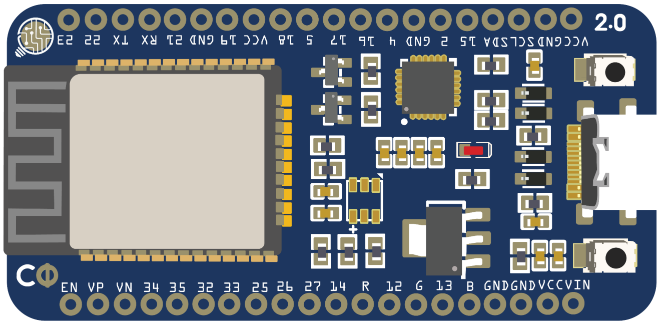
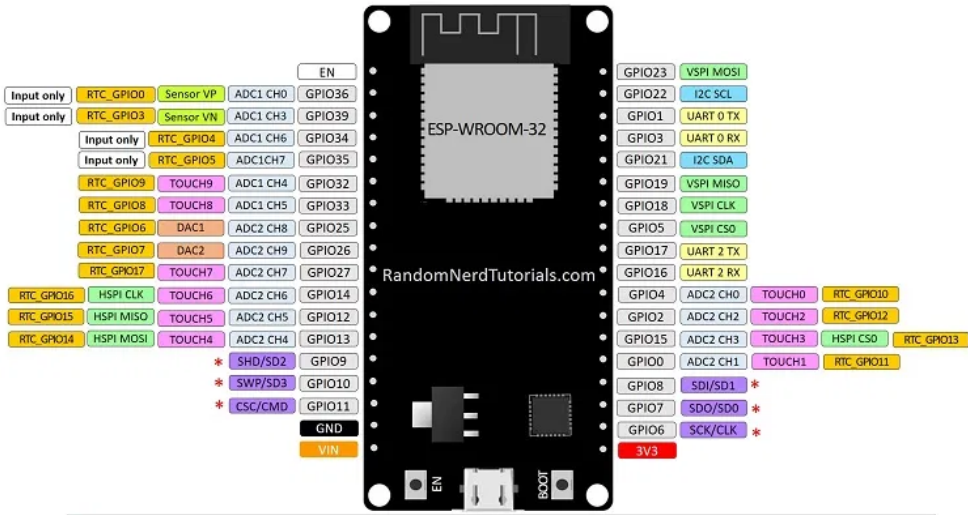
Dev Board 2.0

Built around the **ESP-WROOM-32 SoC**, this development board has many cool features. It has a *16 MB flash* which can be used for high computing processes, and a cool internal *RGB LED*. It uses the USB-C interface to connect to the PC.


2.0
EN VP VN 34 35 32 33 25 26 27 14 R 12 G 13 B GND GND VCC VIN

Pinout Diagram:


ESP-WROOM-32
RandomNerdTutorials.com
EN
BOOT
EN
Input only RTC_GPIO0 Sensor VP ADC1 CH0 GPIO36
Input only RTC_GPIO3 Sensor VN ADC1 CH3 GPIO39
Input only RTC_GPIO4 ADC1 CH6 GPIO34
Input only RTC_GPIO5 ADC1CH7 GPIO35
RTC_GPIO9 TOUCH9 ADC1 CH4 GPIO32
RTC_GPIO8 TOUCH8 ADC1 CH5 GPIO33
RTC_GPIO6 DAC1 ADC2 CH8 GPIO25
RTC_GPIO7 DAC2 ADC2 CH9 GPIO26
RTC_GPIO17 TOUCH7 ADC2 CH7 GPIO27
RTC_GPIO16 HSPI CLK TOUCH6 ADC2 CH6 GPIO14
RTC_GPIO15 HSPI MISO TOUCH5 ADC2 CH5 GPIO12
RTC_GPIO14 HSPI MOSI TOUCH4 ADC2 CH4 GPIO13
* SHD/SD2 GPIO9
* SWP/SD3 GPIO10
* CSC/CMD GPIO11
GND
VIN
GPIO23 VSPI MOSI
GPIO22 I2C SCL
GPIO1 UART 0 TX
GPIO3 UART 0 RX
GPIO21 I2C SDA
GPIO19 VSPI MISO
GPIO18 VSPI CLK
GPIO5 VSPI CS0
GPIO17 UART 2 TX
GPIO16 UART 2 RX
GPIO4 ADC2 CH0 TOUCH0 RTC_GPIO10
GPIO2 ADC2 CH2 TOUCH2 RTC_GPIO12
GPIO15 ADC2 CH3 TOUCH3 HSPI CS0 RTC_GPIO13
GPIO0 ADC2 CH1 TOUCH1 RTC_GPIO11
GPIO8 SDI/SD1 *
GPIO7 SDO/SD0 *
GPIO6 SCK/CLK *
3V3

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Comparison with other Microcontrollers

Here is a detailed comparison table of this Dev Board vs Arduino UNO:

	Arduino UNO	**Dev Board 2.0**
Built around	ATmega 328p	ESP-WROOM-32
Dimensions	53.4 x 68.6 mm	30 x 62 mm
Clock Frequency	16 MHz	240 MHz
Cores	1	2
Operating Voltage	5V	3.3V
Digital I/O Pins	14	24
Analog Input Pins	6	15
SPI/I2C/I2S/UART	1/1/1/1	4/2/2/2
Flash Memory	32 KB	16 MB
SRAM	2 KB	520 KB
WiFi	No	Yes
Bluetooth	No	Yes
Capacitive Touch Pins	No	Yes

RGB LED

Apart from the above pins our development board has 3 extra pins namely - *R(Red), G(Green) and B(Blue)*. These 3 pins are for the in-built RGB LED.
Here's an example code for programming the RGB LED:
Connections:
R → Pin 4
G → Pin 16
B → Pin 17

```
int red = D4;
int green = D16;
int blue = D17;

// the setup routine runs once when you press reset:
void setup()
{
// initialize the digital pin as an output.
pinMode(red, OUTPUT);
pinMode(green, OUTPUT);
pinMode(blue, OUTPUT);
digitalWrite(red, HIGH);
digitalWrite(green, HIGH);
digitalWrite(blue, HIGH);
}

// the loop routine runs over and over again forever:
void loop() {
digitalWrite(red, LOW); // turn the LED on
delay(1000); // wait for a second
digitalWrite(red, HIGH); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
digitalWrite(green, LOW); // turn the LED on
delay(1000); // wait for a second
digitalWrite(green, HIGH); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
digitalWrite(blue, LOW); // turn the LED on
delay(1000); // wait for a second
digitalWrite(blue, HIGH); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

Shields

Shields are boards that can be plugged on top of a Development Board PCB extending its capabilities. There are many advantages to using shields:

- Easy to mount and cheap to produce
- Circuit is much more compact and robust
- We can avoid using breadboards and messy jumper wires

Along with the board, we've developed 3 custom exclusive shields - **Motor Driver Shield**, **Peripheral Shield** and **Game Controller Shield.**

Motor Driver Shield:

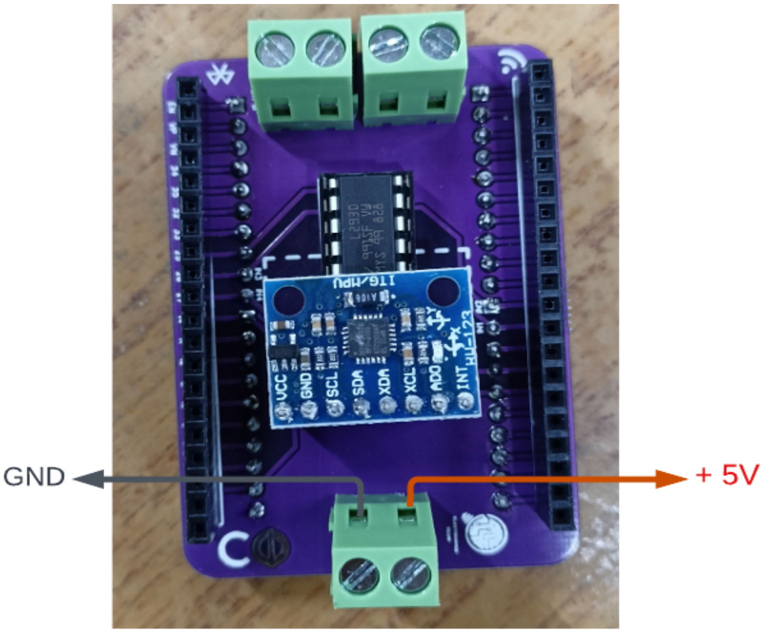
A motor driver is an integrated chip which is used for controlling motors based on the commands it receives from the micro-controller. But why do we need a motor driver? A decent motor requires high voltages from 5V - 12V, but the output we get from a microcontroller is around 3.3V - 5V, and hence it cannot give enough power from its I/O pin to drive a motor. To supply this voltage/current from the microcontroller to the motor, we need this Motor driver IC in between our motor and controller.

This motor driver shield uses the **L293D IC,** which can control up to 2 motors. For more details about the IC, refer to its datasheet. Here's the pic of the shield:

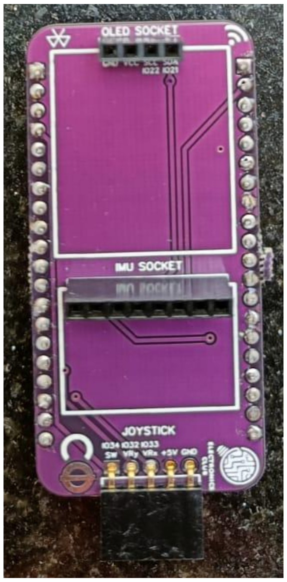
It also contains the IMU socket where we can plugin the **MPU6050 sensor**. This shield has its applications mainly in robotics projects.

The input pins and enable pins of **L293D IC** are connected to the following GPIO pins:

- Input 1 - 16
- Input 2 - 17
- Input 3 - 26
- Input 4 - 27
- Enable A - 14
- Enable B - 25



Peripheral Shield:



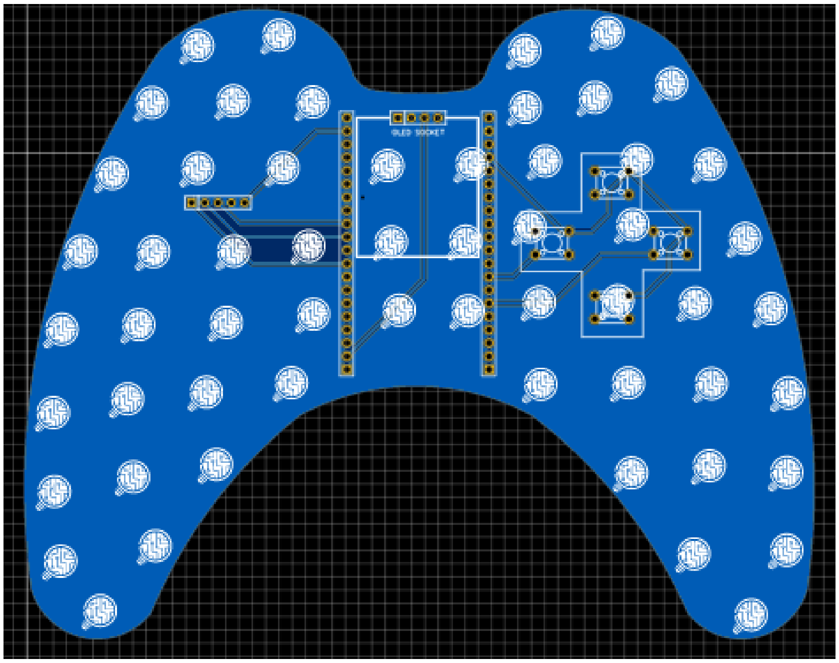
Originally built for a Smartwatch, this shield can be used for many applications where an OLED or IMU or Joystick module will be used. Here's the pic of the shield

The joystick module is connected to the following GPIO pins:

- ➔ VRx - 33
- ➔ VRy - 32
- ➔ SW - 34

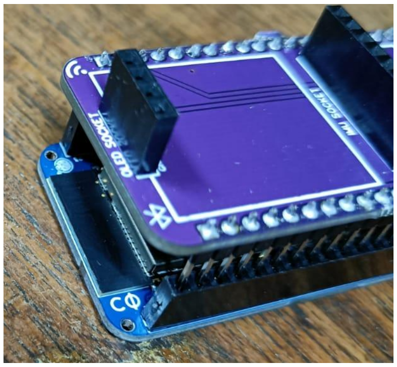
Game Controller Shield

This shield is designed in the shape of a joystick game controller. The idea behind the shield was to interface the ESP32 with your laptop and play games with this. It has 4 push buttons on the right and a joystick module on the left. Here's the pic:



How to mount the Shields on the Dev Board?

We made **Wi-Fi and Bluetooth symbols** on the shields at two of the corners. These are the reference points for mounting them on the dev board. When you are mounting the board, ensure that the Bluetooth symbol aligns with the CFI logo, and the Wi-Fi symbol aligns with the Electronics Club logo on the Dev Board.



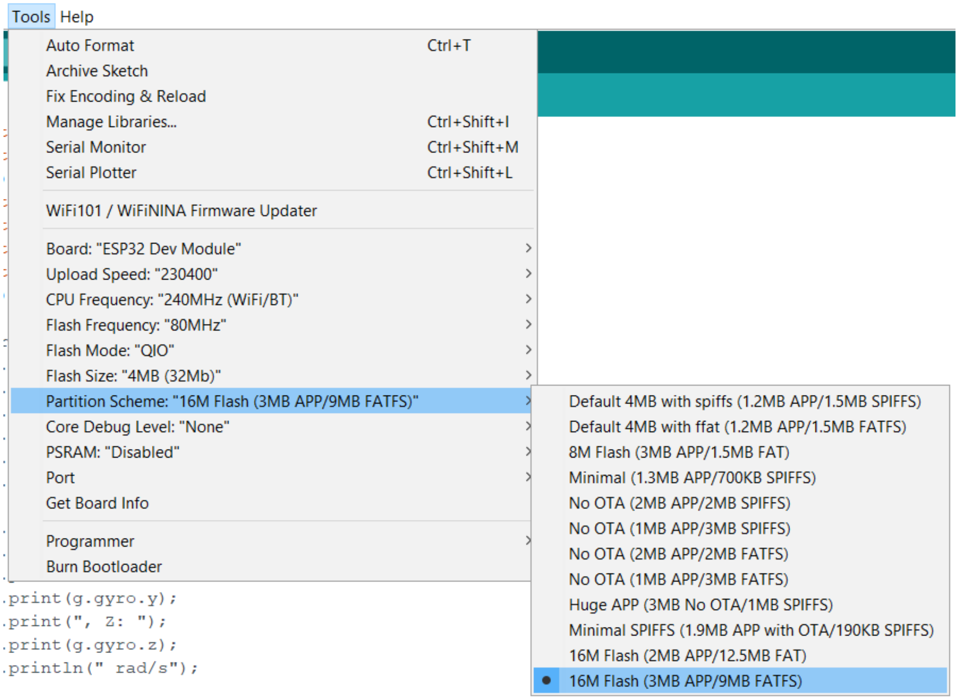
Setting Up Arduino IDE for ESP32

➔ Step 1: Install ***Arduino IDE*** by clicking the following link: https://downloads.arduino.cc/arduino-1.8.19-windows.exe.
Note: ***Do not install Arduino IDE from Microsoft Store!***

➔ Follow this tutorial to set up ***ESP32*** in your *Arduino IDE* https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

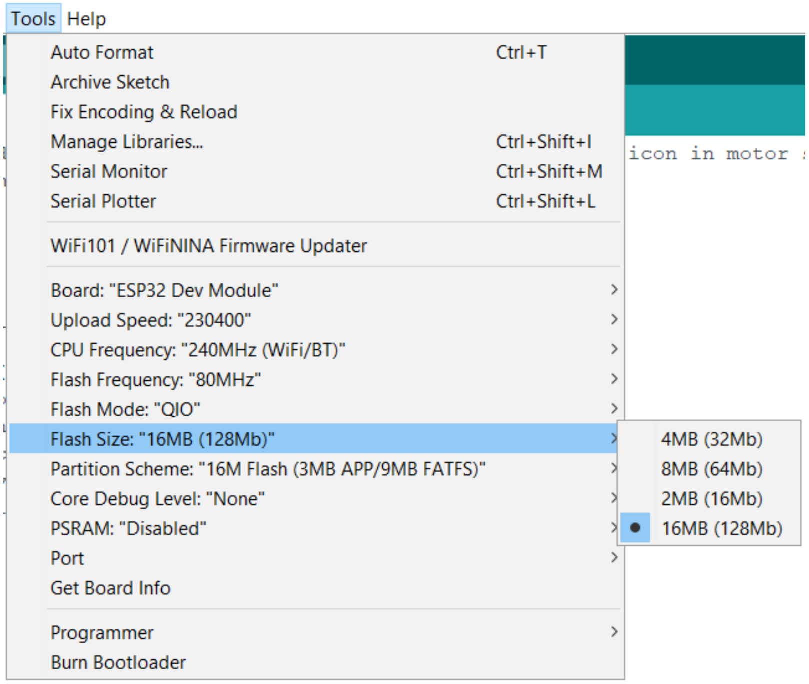
➔ That's it! You can now start off your journey with ESP32 programming. To start off, you can go through this tutorial: https://randomnerdtutorials.com/getting-started-with-esp32/

➔ Before uploading your code, ensure the following settings in the *Tools* menu:


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Port
Get Board Info
Programmer
Burn Bootloader
Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)
Default 4MB with ffat (1.2MB APP/1.5MB FATFS)
8M Flash (3MB APP/1.5MB FAT)
Minimal (1.3MB APP/700KB SPIFFS)
No OTA (2MB APP/2MB SPIFFS)
No OTA (1MB APP/3MB SPIFFS)
No OTA (2MB APP/2MB FATFS)
No OTA (1MB APP/3MB FATFS)
Huge APP (3MB No OTA/1MB SPIFFS)
Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)
16M Flash (2MB APP/12.5MB FAT)
● 16M Flash (3MB APP/9MB FATFS)

Board: "ESP32 Dev Module"
Upload Speed: "230400"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "16M Flash (3MB APP/9MB FATFS)"
Core Debug Level: "None"
PSRAM: "Disabled"

```
.print(g.gyro.y);
.print(", Z: ");
.print(g.gyro.z);
.println(" rad/s");
```



Board: "ESP32 Dev Module"
Upload Speed: "230400"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "16MB (128Mb)"
Partition Scheme: "16M Flash (3MB APP/9MB FATFS)"
Core Debug Level: "None"
PSRAM: "Disabled"

Designed By:

- Aneesh Kandi
- Gokul C