

# AI AVENGERS



Figure 1 Fully assembled kit

# HARDWARE SETUP

Bot Pictures:

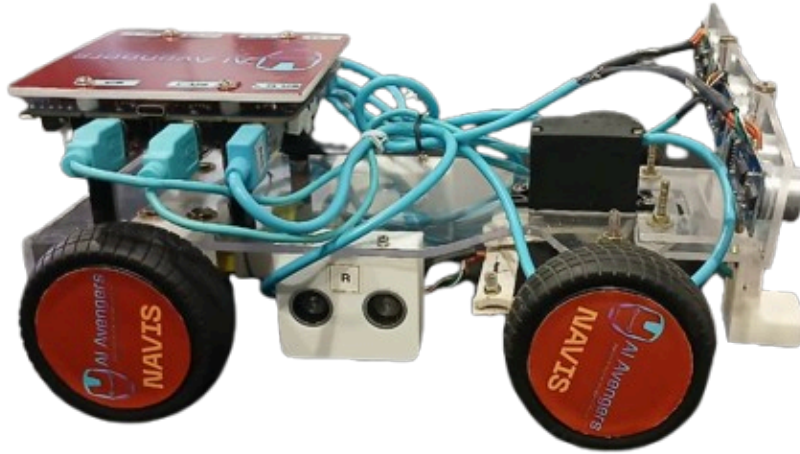


Figure 2 Mechanical Parts

## PCB CABLE CONNECTION

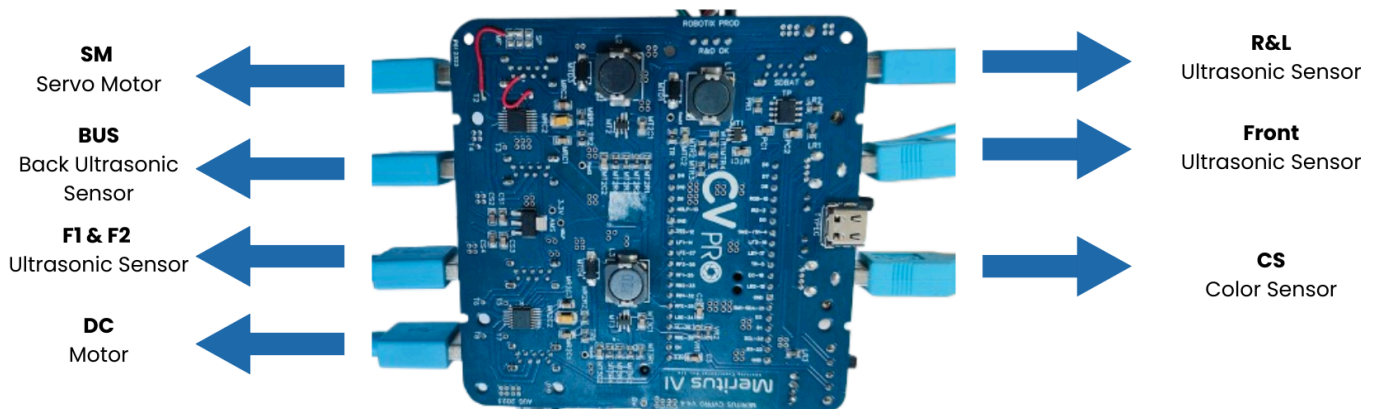


Figure 3 PCB Port and Usages (Top view)

Software Required:

The software applications required for working with 'Competition kit' - [Arduino IDE](#).  
Since we are using ESP32 board manager has to be installed. Refer to the [link](#), for the installation steps.

## Modules and Components:

### 1. ESP32 Board:

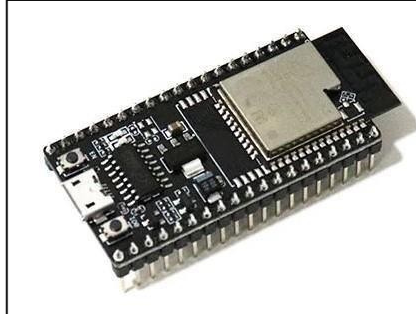


Figure 4 ESP32 Development board

The PCB contains an ESP32 as a controller. It is a versatile microcontroller, combining a dual-core processor with built-in Wi-Fi and Bluetooth capabilities. It's highly efficient and ideal for various IoT applications. Its dual-core architecture supports multitasking, making it efficient for both processing and communication tasks. With a wide range of GPIO pins, it's flexible for interfacing with sensors, motors, and other devices. Additionally, its low power consumption and compatibility with numerous development platforms make it a popular choice for IoT projects and beyond.

**Important Note:** To install the ESP32 board in your Arduino IDE, follow [the link](#).

### 2. I2C Color Sensor:



Figure 6 Color Sensor

An I2C color sensor is a device that detects and measures colors using the I2C communication protocol. It typically integrates various photodiodes, filters, and electronics to accurately perceive and differentiate different colors. By utilizing an I2C color sensor in this project, you can detect and measure colors accurately, enabling a wide range of applications that require color analysis and processing.

**Applications:** I2C color sensors find applications in a variety of fields, including industrial automation, consumer electronics, robotics, healthcare, automotive, and more. They are used for color sorting, color matching, display calibration, and various other color-related applications.

### 3. HC - SR04 Ultrasonic Sensor



Figure 7 Ultrasonic Sensor

The HC-SR04 Ultrasonic Sensor is a widely used device for measuring distances based on the time it takes for ultrasonic waves to bounce back from an object. It consists of a transmitter that emits ultrasonic waves and a receiver that detects the waves. By calculating the time between emission and reception, the sensor determines the distance from the object. With a measurement range of 2 cm to 400 cm, Easy to interface with microcontrollers, the HC-SR04 sensor offers reliable and accurate distance measurements, making it a popular choice in the electronics and robotics communities.

## PCB inbuilt functions

1. Charging and discharging - Charging will only be enabled if the bot is in OFF state.
2. Green LED will indicate when the bot is fully charge in off state. (Ensure that bot runs for 50 minutes from time of full charge condition).
3. Li-ion 3.7V single cell 3200mAh.
4. Power module for powering the board, sensors and motors.
5. Motor driver for controlling the motor.
6. ESP32 micro controller for executing both wired and wireless communications and algorithms. The programming can be done through the given Type-C port.

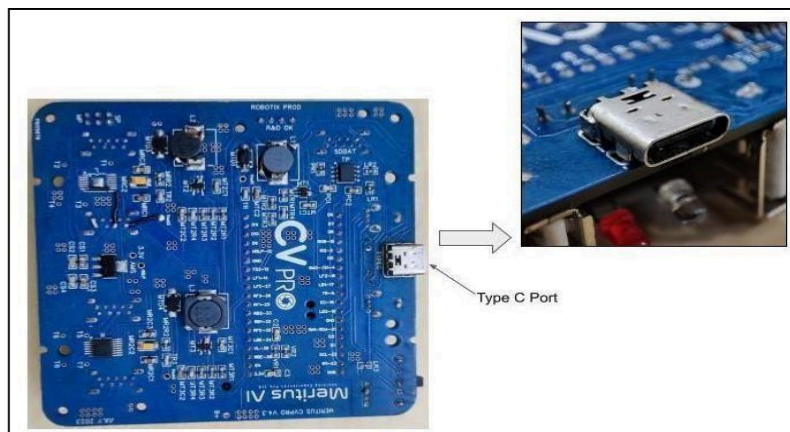


Figure 8 PCB (Bottom view) & Type C port

### Port and Pin details:

Function	Port Type	Port No	GPIO Pins
Motor (Battery Operated Motor)	USB 3.0	1	32,33
F1US & F2US (Front 1 & Front 2 Ultrasonic Sensors)	USB 3.0	2	F1-16,14 and F2-25,26
BUS (Back Ultrasonic Sensor)	USB 3.0	3	17,19
SM (Servo Motor)	USB 3.0	4	27
RUS & LUS (Right & Left Ultrasonic Sensors)	USB 3.0	5	Right-2,23 and Left-5,18
FUS (Front Ultrasonic Sensor)	USB 2.0	6	12,4
CS (Color Sensor)	USB 2.0	7	22,21
RGB LED	-	-	15
NSLEEP For Motor	-	-	13
Battery Voltage Reading	-	-	39
DPDT Push Button	-	-	34
Optical Encoder	-	-	36
		-	

- Kindly refer to the pins provided in the above table for programming firmware.

### The competition kit will be capable of:

- Detecting color on the floor using color sensor
- Avoiding obstacle using ultrasonic sensors
- Accomplishing any of the user-defined tasks

Follow the steps given below to operate the competition kit provided to you:

1. Create the Arduino program to upload in Arduino IDE.
2. Open the firmware in IDE.
3. Select the respective communication (COM) port and select the board as (Do it yourself devkit).
4. Switch on the kit.
5. Click 'Upload' button to upload the created program into the kit.
6. Turn on to accomplish the programmed task.

Maximizing performance of kit:

1. USB cables must be connected to their designated ports.
2. The kit's servo angle is fixed at 100 degrees.
3. The PCB lacks protective covering; avoid placing conductive materials on it.
4. When adjusting the angle, stay within a 20-degree range to the left and right of the center angle (100 degrees), which allows movement between 80 and 120 degrees. Deviating beyond these limits may result in damage to the product.
5. Handle with care to avoid wire wear.

Key Library files required:

The following library files are crucial for proper functioning of the bot:

- `#include <ESP32Servo.h>` - This library allows to control servo motors with precision and ease, enabling precise movements in our system.
- `#include <NewPing.h>` - NewPing is used for accurate and efficient distance sensing, particularly useful when working with ultrasonic sensors. It simplifies the task of measuring distances.
- `#include <Wire.h>` - The Wire library facilitates communication on the I2C bus, which is commonly used for connecting multiple devices in our setup.
- `#include "Adafruit_TCS34725.h"` - This library is essential for working with Adafruit TCS34725 color sensor, enabling us to capture and process color data effectively.

**Note:** Consider exploring additional compatible libraries as well.



## Library and its Versions:

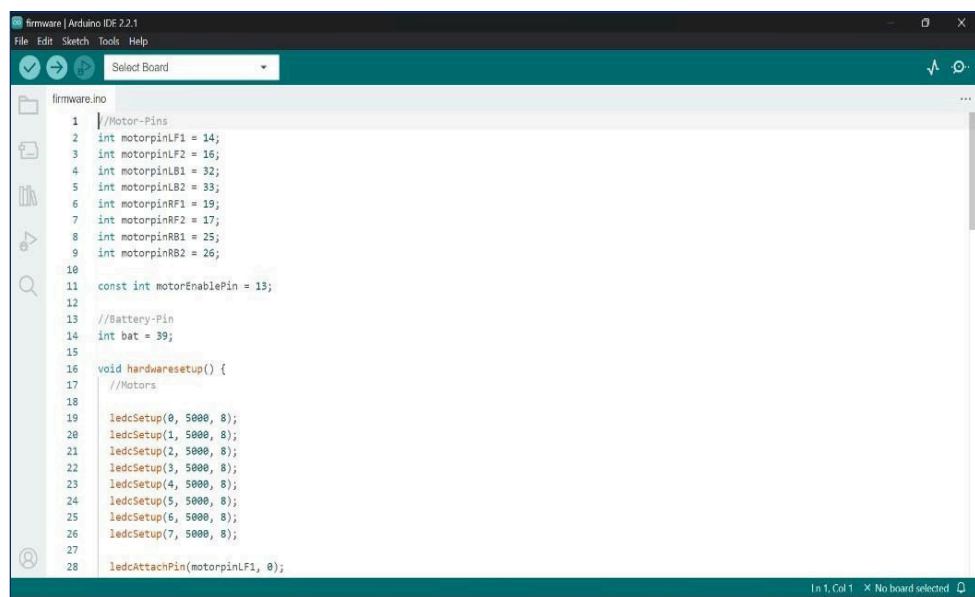
1. Go to Board Manager and type `esp32`, Select Version `2.0.17` and install.
2. Go to Library Manager and type `NewPing`, Select Version `1.9.0` and install.
3. Go to Library Manager and type `FastLED`, Select Version `7.7.1` and install.
4. Go to Library Manager and type `ESP32Servo`, Select Version `3.71.2.1` and install.
5. Go to Library Manager and type `Adafruit TCS34725`, Select Version `1.4.4` and install.

**\*\* Kindly Install the Respective Libraries Mentioned \*\***

## Bot Test Codes:

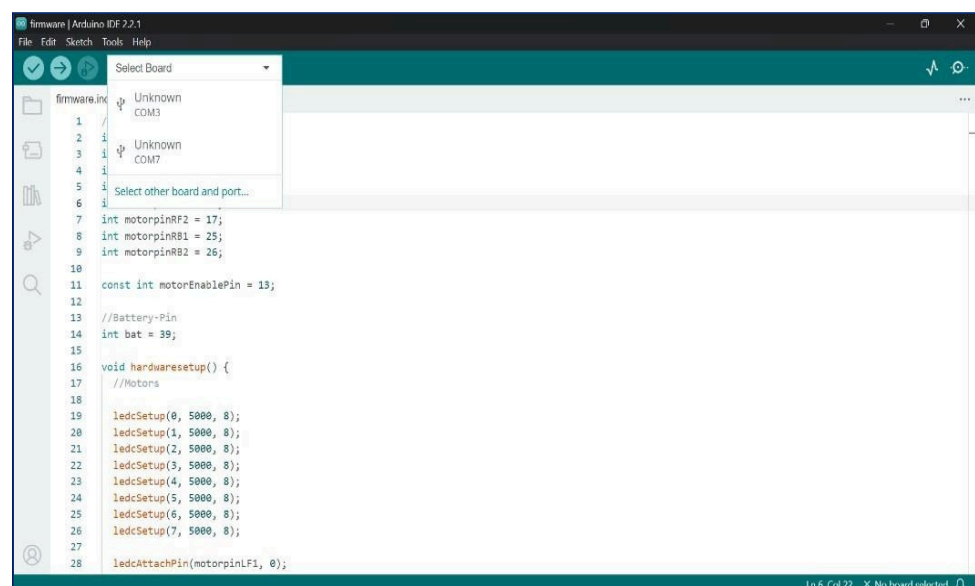
It is imperative to test the bot for its functionality. For testing the test codes available in the downloaded GitHub repository can be used. It is available in the path – ‘Firmware/Bot-TestCodes’.

1. Launch the Arduino IDE on your computer and open your ‘.ino’ file. A sample is shown below.



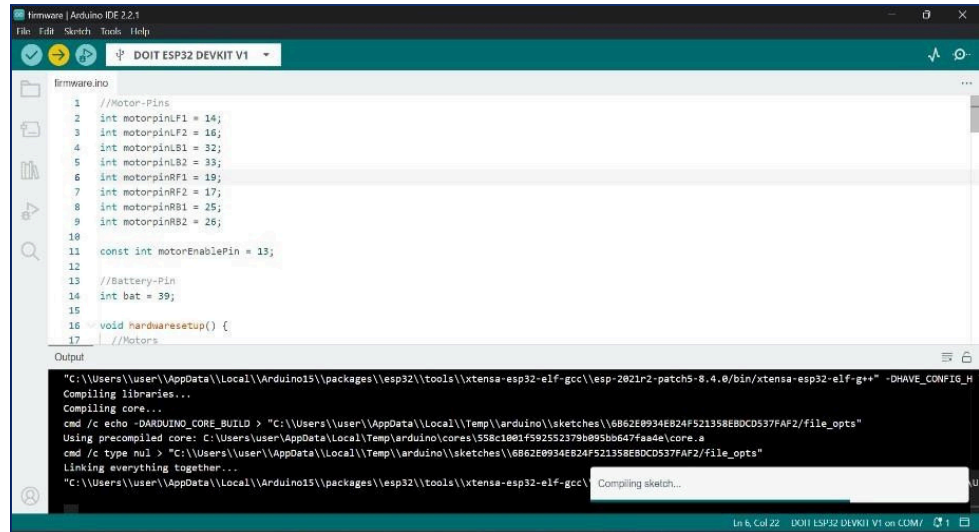
```
1 //Motor-Pins
2 int motorpinLF1 = 14;
3 int motorpinLF2 = 16;
4 int motorpinLB1 = 32;
5 int motorpinLB2 = 33;
6 int motorpinRF1 = 19;
7 int motorpinRF2 = 17;
8 int motorpinRB1 = 25;
9 int motorpinRB2 = 26;
10
11 const int motorEnablePin = 13;
12
13 //Battery-Pin
14 int bat = 39;
15
16 void hardwareSetup() {
17     //Motors
18
19     ledcSetup(0, 5000, 8);
20     ledcSetup(1, 5000, 8);
21     ledcSetup(2, 5000, 8);
22     ledcSetup(3, 5000, 8);
23     ledcSetup(4, 5000, 8);
24     ledcSetup(5, 5000, 8);
25     ledcSetup(6, 5000, 8);
26     ledcSetup(7, 5000, 8);
27
28     ledAttachPin(motorpinLF1, 0);
```

2. Select the respective board and COM port as shown below.



```
1 //
2
3
4
5
6
7 int motorpinRF2 = 17;
8 int motorpinRB1 = 25;
9 int motorpinRB2 = 26;
10
11 const int motorEnablePin = 13;
12
13 //Battery-Pin
14 int bat = 39;
15
16 void hardwareSetup() {
17     //Motors
18
19     ledcSetup(0, 5000, 8);
20     ledcSetup(1, 5000, 8);
21     ledcSetup(2, 5000, 8);
22     ledcSetup(3, 5000, 8);
23     ledcSetup(4, 5000, 8);
24     ledcSetup(5, 5000, 8);
25     ledcSetup(6, 5000, 8);
26     ledcSetup(7, 5000, 8);
27
28     ledAttachPin(motorpinLF1, 0);
```

- Click on 'Upload' button, and upload the test code file, 'ColorSensor\_Test\_Code' to start the testing. You may try with each of the test codes provided. The IDE first compiles the code and then upload the code to the 'Esp32' board.



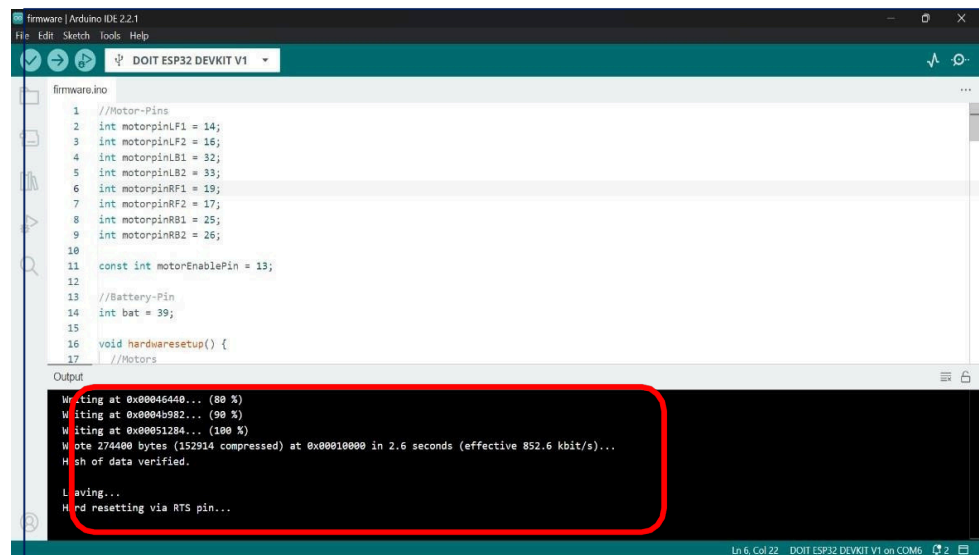
The screenshot shows the Arduino IDE 2.2.1 interface. The 'Sketch' tab is active, displaying a file named 'firmware.ino'. The code in the editor includes pin definitions for motor pins (14, 16, 32, 33, 19, 17, 25, 26) and a battery pin (39), along with a motor enable pin (13). The 'Output' window at the bottom shows the compilation process, including the command to compile the sketch using the ESP32 toolchain. The status bar at the bottom indicates 'Ln 6, Col 22 DOIT ESP32 DEVKIT V1 on COM7'.

```
firmware.ino
1 //Motor-Pins
2 int motorpinLF1 = 14;
3 int motorpinLF2 = 16;
4 int motorpinLB1 = 32;
5 int motorpinLB2 = 33;
6 int motorpinRF1 = 19;
7 int motorpinRF2 = 17;
8 int motorpinRB1 = 25;
9 int motorpinRB2 = 26;
10
11 const int motorEnablePin = 13;
12
13 //Battery-Pin
14 int bat = 39;
15
16 void hardwareSetup() {
17 //Motors
```

```
Output
"C:\Users\User\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf-gcc\esp-2021r2-patch5-8.4.0/bin/xtensa-esp32-elf-g++" -DHAVE_CONFIG_H
Compiling libraries...
Compiling core...
cmd /c echo -DARDUINO_CORE_BUILD > "C:\Users\User\AppData\Local\Temp\arduino\sketches\6862E0934E824F521358E8DCD537FAF2/file_opts"
Using precompiled core: C:\Users\User\AppData\Local\Temp\arduino\cores\558c1081f592552379b095b647faa4e\core.a
cmd /c type nul > "C:\Users\User\AppData\Local\Temp\arduino\sketches\6862E0934E824F521358E8DCD537FAF2/file_opts"
Linking everything together...
"C:\Users\User\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf-gcc\ Compiling sketch...
```

Ln 6, Col 22 DOIT ESP32 DEVKIT V1 on COM7

- The following message (highlighted) will be displayed when the uploading is complete.



The screenshot shows the Arduino IDE 2.2.1 interface. The 'Sketch' tab is active, displaying the same 'firmware.ino' file. The 'Output' window at the bottom shows the upload progress, including the command to upload the sketch to the ESP32 board. The status bar at the bottom indicates 'Ln 6, Col 22 DOIT ESP32 DEVKIT V1 on COM6'.

```
firmware.ino
1 //Motor-Pins
2 int motorpinLF1 = 14;
3 int motorpinLF2 = 16;
4 int motorpinLB1 = 32;
5 int motorpinLB2 = 33;
6 int motorpinRF1 = 19;
7 int motorpinRF2 = 17;
8 int motorpinRB1 = 25;
9 int motorpinRB2 = 26;
10
11 const int motorEnablePin = 13;
12
13 //Battery-Pin
14 int bat = 39;
15
16 void hardwareSetup() {
17 //Motors
```

```
Output
Writing at 0x00046440... (80 %)
Writing at 0x0004b982... (90 %)
Writing at 0x00051284... (100 %)
Wrote 274400 bytes (152914 compressed) at 0x00100000 in 2.6 seconds (effective 852.6 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...
```

Ln 6, Col 22 DOIT ESP32 DEVKIT V1 on COM6