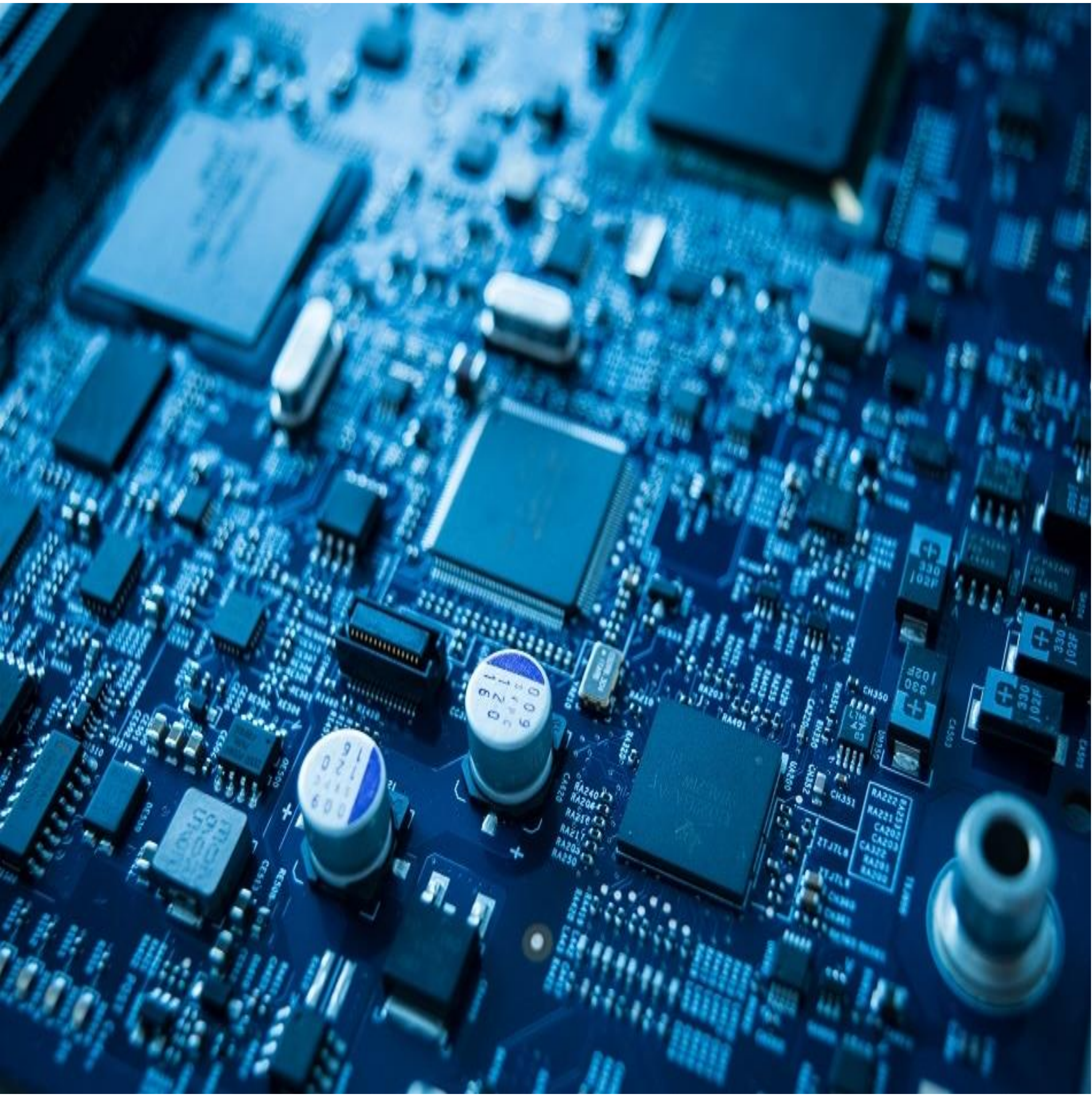


INDIVIDUAL REPORT

MICROPROCESSORS AND MICROCONTROLLERS

BY:

SHIBIL IRFAAN .I

21R238

Microcontrollers and Embedded Systems:

In an era defined by rapid technological advancements, microcontrollers and embedded systems stand as the silent architects of modern convenience and efficiency. These unassuming components are the driving force behind the seamless operation of everyday devices, from household appliances to industrial machinery. This short essay delves into the realm of microcontrollers and embedded systems, highlighting their significance, applications, and contributions to our interconnected world.

Microcontrollers at a Glance:

At the heart of microcontrollers lies their unique architecture: a single integrated circuit that combines a central processing unit (CPU), memory, input/output peripherals, and a clock source. Unlike general-purpose computers, microcontrollers are designed to perform specific tasks with precision and speed. Their compact size and low power consumption make them ideal for applications where efficiency is paramount.

Embedded Systems:

Embedded systems leverage microcontrollers to execute tasks within larger systems. These systems integrate hardware and software to perform dedicated functions, often hidden from the user's view. From medical devices monitoring vital signs to industrial robots streamlining production, embedded systems enable enhanced control and automation across diverse industries.

Development and Challenges:

Developing applications for microcontrollers requires specialized tools and knowledge. Integrated Development Environments (IDEs) facilitate coding, while languages like C and C++ provide the means to communicate with hardware. The challenges of limited resources, memory constraints, and real-time performance demand careful design and optimization.

ESP8266

The ESP8266 is a popular and versatile microcontroller module that has gained significant attention in the world of embedded systems and Internet of Things (IoT) development. Manufactured by Espressif Systems, the ESP8266 module offers a powerful platform for creating Wi-Fi-enabled projects and applications. Let's take a closer look at some of its key features, applications, and benefits.

Features of the ESP8266:

1. Microcontroller Core: The ESP8266 is powered by a Tensilica Xtensa LX106 microcontroller core, running at clock speeds up to 160 MHz.

2. Wi-Fi Connectivity: One of the standout features of the ESP8266 is its built-in Wi-Fi capability, making it suitable for connecting to wireless networks and the internet. It supports both station (client) and access point (server) modes.

3. Memory: The module is available with varying amounts of flash memory, commonly ranging from 512 KB to 4 MB. This memory is used to store program code, firmware, and data.

4. GPIO Pins: The ESP8266 has a number of General-Purpose Input/Output (GPIO) pins that can be used for digital input/output, analog input, and more.

5. UART, SPI, I2C Interfaces: The module provides multiple communication interfaces like UART (serial communication), SPI (Serial Peripheral Interface), and I2C (Inter-Integrated Circuit), which make it compatible with various sensors, displays, and other peripherals.

6. Analog-to-Digital Converter (ADC: The built-in ADC allows you to measure analog signals, making it suitable for reading sensor data.

7. Low Power Consumption: Depending on its operating modes and settings, the ESP8266 can be configured to operate in low-power modes to conserve energy.

8. Community Support: The ESP8266 has a large and active community of developers, which has led to the creation of various libraries, tools, and resources for programming and debugging.

Applications of the ESP8266:

1. IoT Devices: The ESP8266's Wi-Fi connectivity and compact size make it an ideal choice for building IoT devices that can communicate with other devices and the cloud.

2. Home Automation: It can be used to create smart home solutions, such as controlling lights, appliances, and other systems remotely.

3. Sensor Networks: The ESP8266 can interface with various sensors to collect data and transmit it wirelessly, enabling the creation of sensor networks for monitoring and data collection.

4. Remote Control: It can be used to build remote-controlled systems, such as garage door openers or remote-controlled robots.

5. Data Logging and Monitoring: With its connectivity and memory capabilities, the ESP8266 can collect and log data from various sources for later analysis.

MOBILE MANIPULATOR ROBOT

A mobile robot with a pick and place arm and a camera is commonly referred to as an "Autonomous Mobile Manipulator" or simply a "Mobile Manipulator." This type of robot combines the mobility of a wheeled or tracked platform with the capabilities of a robotic arm for picking up, moving, and placing objects, often aided by vision systems such as cameras.

The term "mobile" highlights its ability to move around its environment, while "manipulator" refers to its robotic arm's capability to manipulate objects. The addition of a camera or vision system enables the robot to perceive its surroundings, recognize objects, and make informed decisions about its actions.

In industrial and research contexts, these robots are often used for tasks such as warehouse automation, manufacturing, assembly, inspection, and even research projects. They are designed to perform tasks autonomously or under remote supervision, enhancing efficiency, accuracy, and safety in various applications.

MICROCONTROLLERS USED:

1) ESP 8266
2) ESP 32

BOARDS USED:

1) NODE MCU
2) ESP 32 CAM MODULE
3) L298N Motor Driver
4) DC-DC step down
5) 3s- BMS

OTHER COMPONENTS:

1) Chassis
2) 12V 3000Mah battery
3) 12V geared motors
4) Mg90s servos
5) Wires

SOFTWARES USED:

1) Arduino IDE
2) Blynk IOT

EMBEDDED C CODING(Arduino IDE):

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <Servo.h>

#define ENA D4

#define IN1 D6

#define IN2 D7

#define IN3 D8

#define IN4 D9

#define ENB D5


bool forward = 0;

bool backward = 0;

bool left = 0;

bool right = 0;

int Speed;

// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).
```

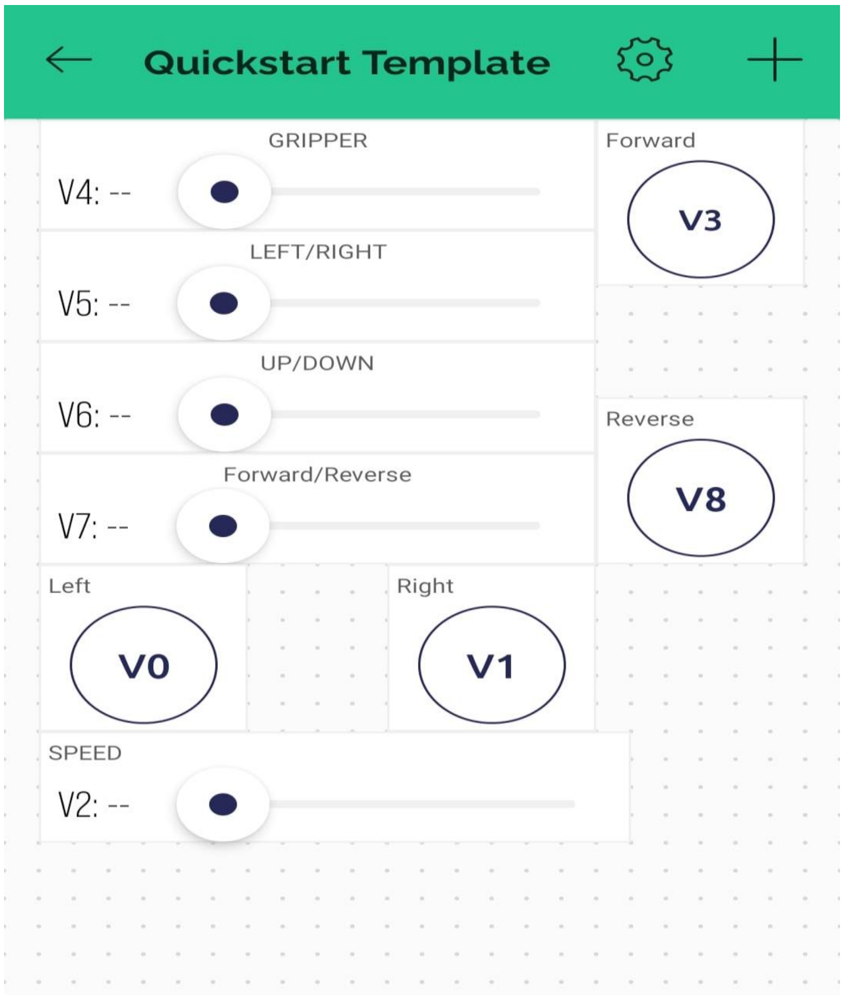
```
char auth[] = "@@@@@@@@@@@@@";  //Paste your Auth Token Here

// Your WiFi credentials.

// Set password to "" for open networks.


char ssid[] = "!!!!!!!!!!!"; //Enter your Wifi User Name

char pass[] = "********"; //Enter your Wifi Password


Servo servo;

Servo servo1;

Servo servo2;

Servo servo3;


BLYNK_WRITE(V7)

{

  servo.write(param.asInt());

}


BLYNK_WRITE(V4)

{

  servo1.write(param.asInt());

}


BLYNK_WRITE(V5)

{

  servo2.write(param.asInt());

}


BLYNK_WRITE(V6)

{

  servo3.write(param.asInt());
```

```
BLYNK_WRITE(V3) {
  forward = param.asInt();
}

BLYNK_WRITE(V8) {
  backward = param.asInt();
}

BLYNK_WRITE(V1) {
  left = param.asInt();
}

BLYNK_WRITE(V0) {
  right = param.asInt();
}

BLYNK_WRITE(V2) {
  Speed = param.asInt();
}


void setup()

{
  // Debug console
  Serial.begin(9600);
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);


  servo.attach(D0);
  servo1.attach(D1);
  servo2.attach(D2);
  servo3.attach(D3);
}
void smartcar() {
  if (forward == 1) {
    carforward();
    Serial.println("carforward");
  } else if (backward == 1) {
    carbackward();
    Serial.println("carbackward");
  } else if (left == 1) {
    carturnleft();
    Serial.println("carfleft");
  } else if (right == 1) {
    carturnright();
    Serial.println("carright");
  } else if (forward == 0 && backward == 0 && left == 0 && right == 0) {
    carStop();
    Serial.println("carstop");
  }
```

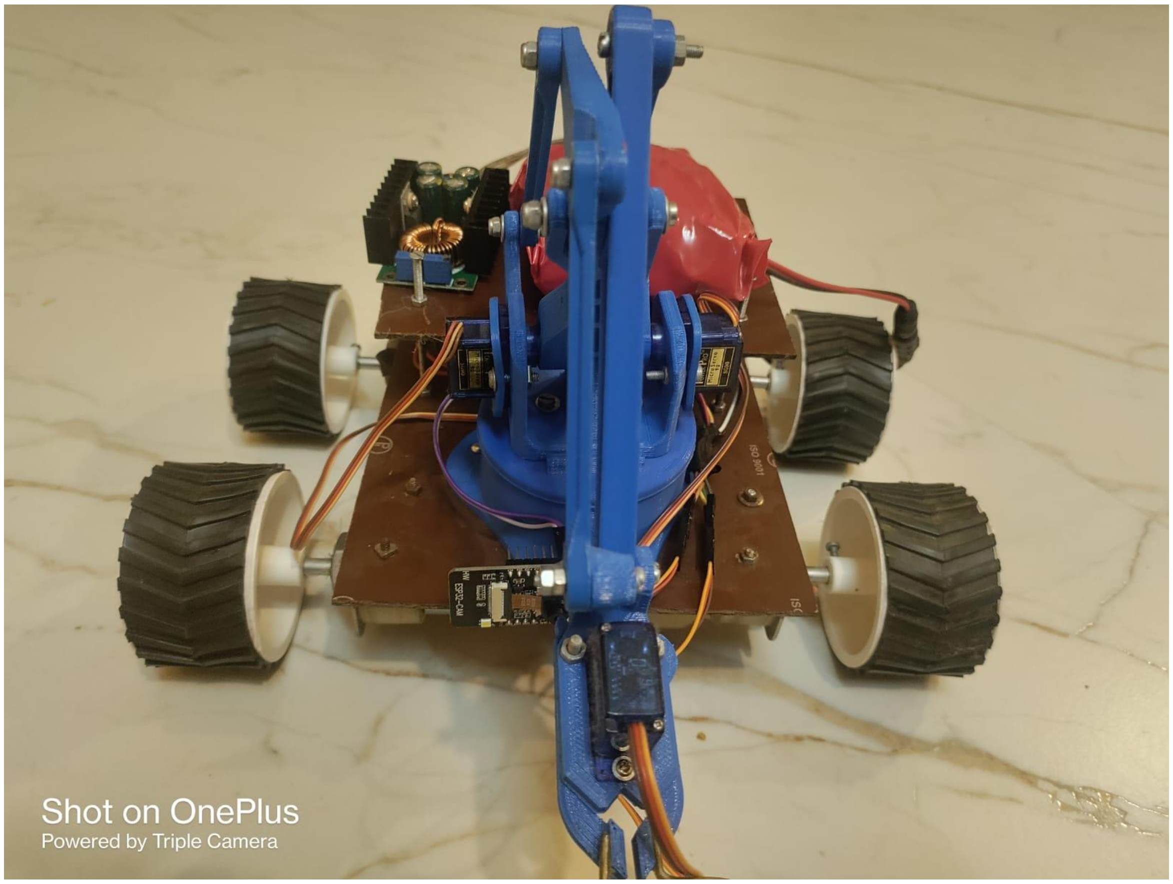
```
}
void loop()
{
  Blynk.run();
  smartcar();
}
void carforward() {
  analogWrite(ENA, Speed);
  analogWrite(ENB, Speed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void carbackward() {
  analogWrite(ENA, Speed);
  analogWrite(ENB, Speed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void carturnleft() {
  analogWrite(ENA, Speed);
  analogWrite(ENB, Speed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
void carturnright() {
```

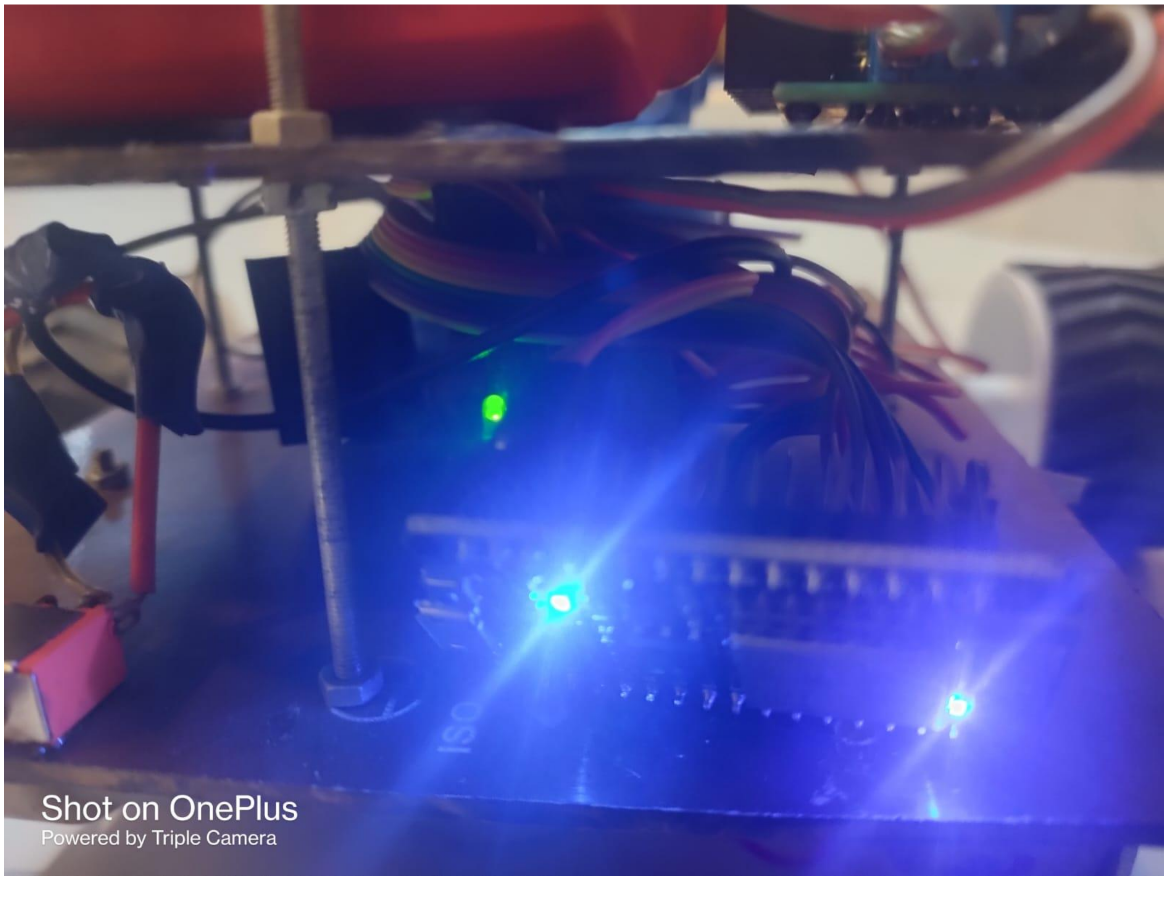
```
analogWrite(ENA, Speed);
  analogWrite(ENB, Speed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
void carStop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

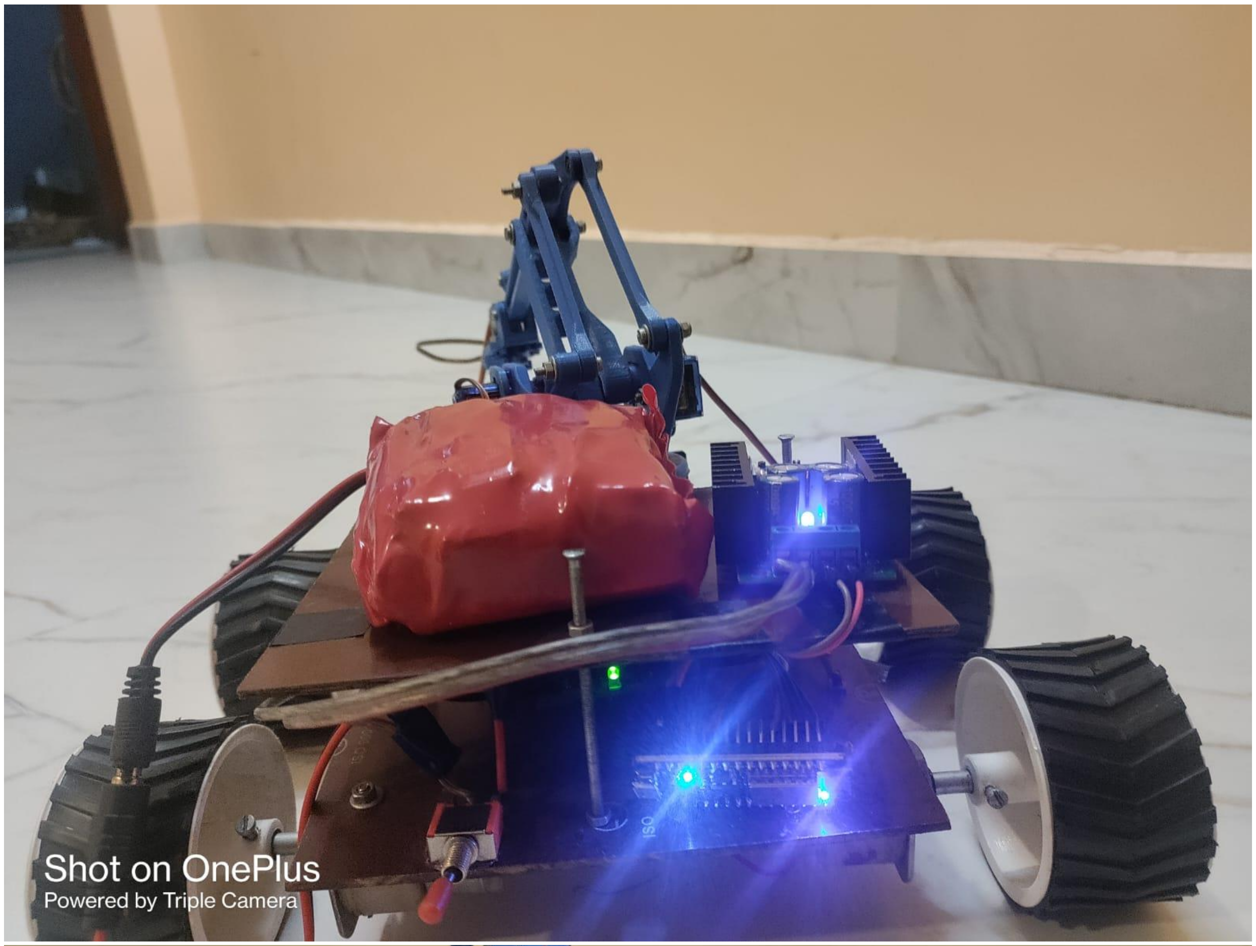
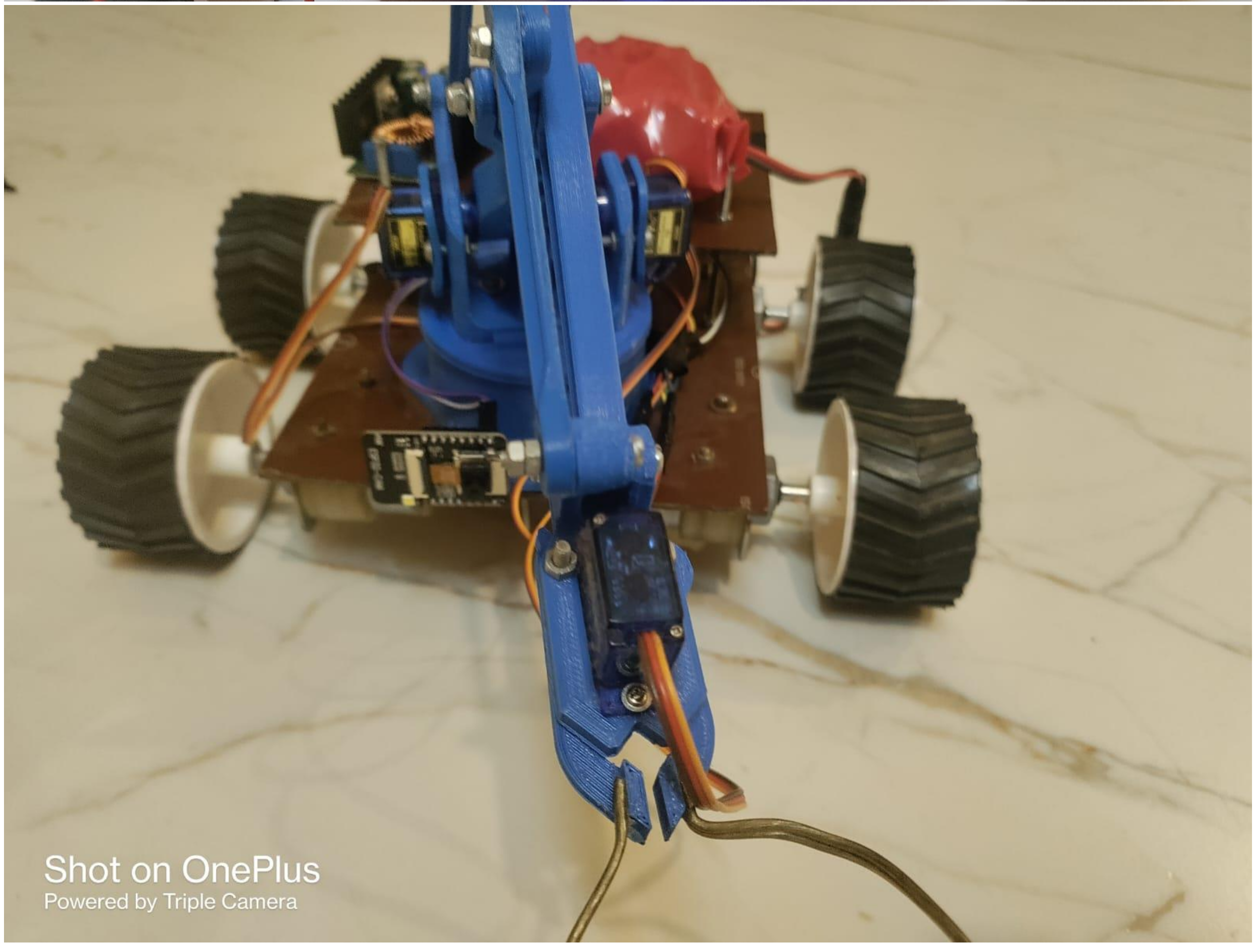
BLYNK APPLICATION INTERFACE:



IMAGES OF MY HARDWARE MODEL :






Shot on OnePlus
Powered by Triple Camera

Shot on OnePlus
Powered by Triple Camera