AR Gesture Controlled Car

Summer Project 2024

Overview

- Aim of the Project
- Part I: Hardware
- Ideation Process
- Computer Vision
- Implementation of Augmented Reality
- Problems Faced
- Future Goals

Aim of the Project

01. **Hardware:** Implementation of microcontrollers, motor drivers, batteries, DC motors and input devices like IMU sensors and Flex sensors to make and control a car using a user's hand gestures.

02. **Computer Vision:** Integration of a camera situated within the car with the application in the phone using CV enables us to view the car's usual FPV as a thrilling race track taking the car into AR.

03. **Augmented Reality:** Modification and enhancement of user experience, by overlaying of virtual objects in the camera feed obtained using on-board camera, like that of ESP32 CAM. This involves use of marker-based and markerless AR.

Ideation process

01

SETTING UP SERIAL AND WIRELESS COMMUNICATION BETWEEN TWO NODE MCUs

02

USING A MPU 6050 FOR DIRECTING THE CAR MOVEMENT ACCORDING TO OUR HAND GESTURES

03

POWERING THE DC MOTOR USING L293D MOTOR DRIVER AND USING MPU6050 FOR CALCULATING ROLL AND PITCH

04

TRANSMITTING THE READINGS FROM MPU6050 SENSOR AND ACCORDINGLY ADJUSTING THE CAR'S MOVEMENT

Ideation process

05

IMPLEMENTATION OF AUGMENTED REALITY (AR) PART USING UNITY SDK

06

DEVELOPMENT OF APPLICATIONS FOR MARKER-BASED AND MARKERLESS AR

07

USING THE CAMERA FEED FROM ONBOARD CAMERA TO FEED AS INPUT TO THE DEVELOPED APPLICATION TO OVERLAY IMAGES AND 3D OBJECTS ONTO THIS CAMERA FEED.

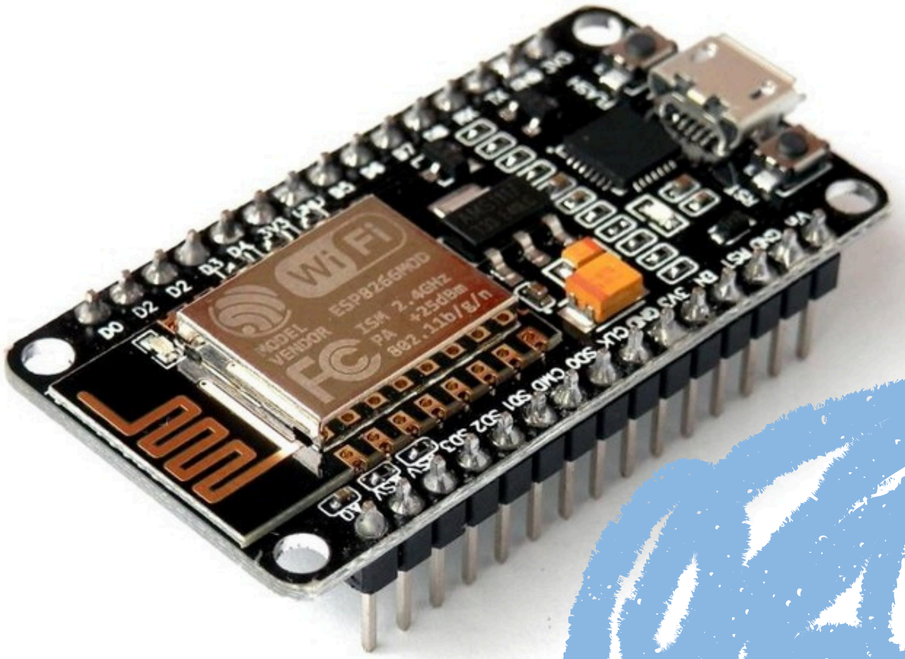
08

DISPLAY OF THE CAMERA FEED, ENHANCED WITH OVERLAY OBJECTS ON MOBILE/COMPUTER IN A CUSTOM-BUILT APPLICATION TO ENHANCE USER EXPERIENCE

Part I : Hardware

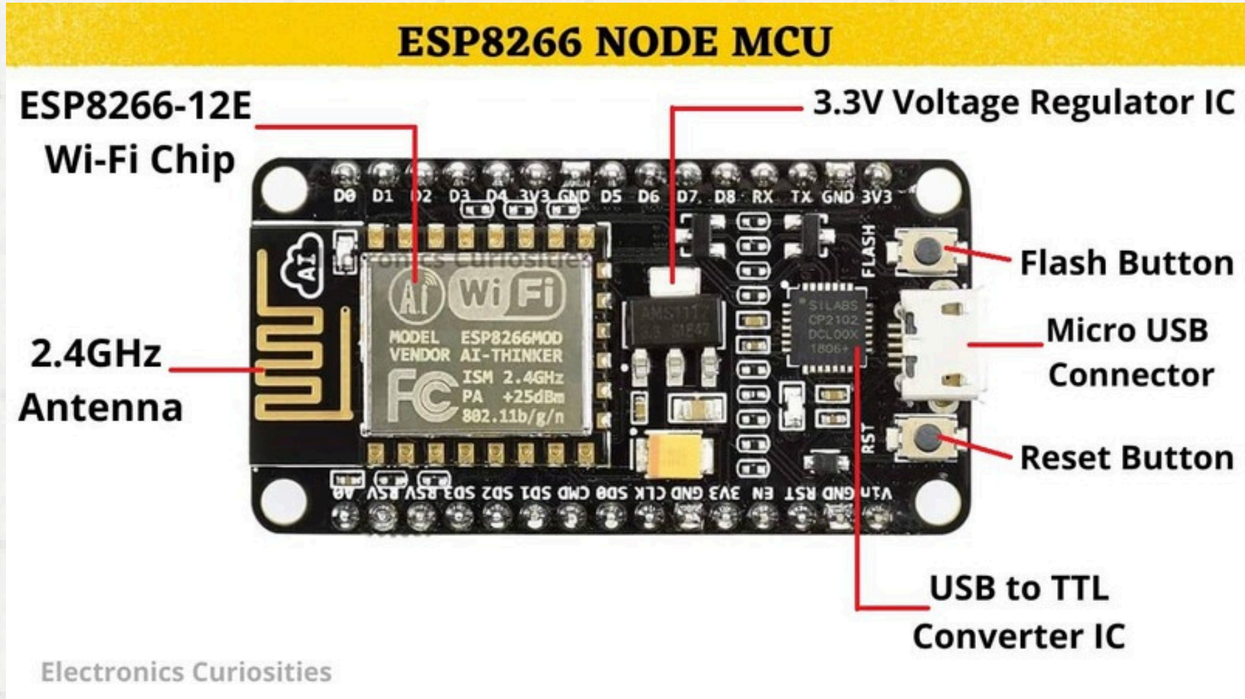
To control our car by using only hand gestures, the following components were necessary to be set up:

- Arduino
- NodeMCU (ESP8266)
- NodeMCU (ESP32 CAM)
- IMU (MPU6050)
- Motor Driver (L293D)



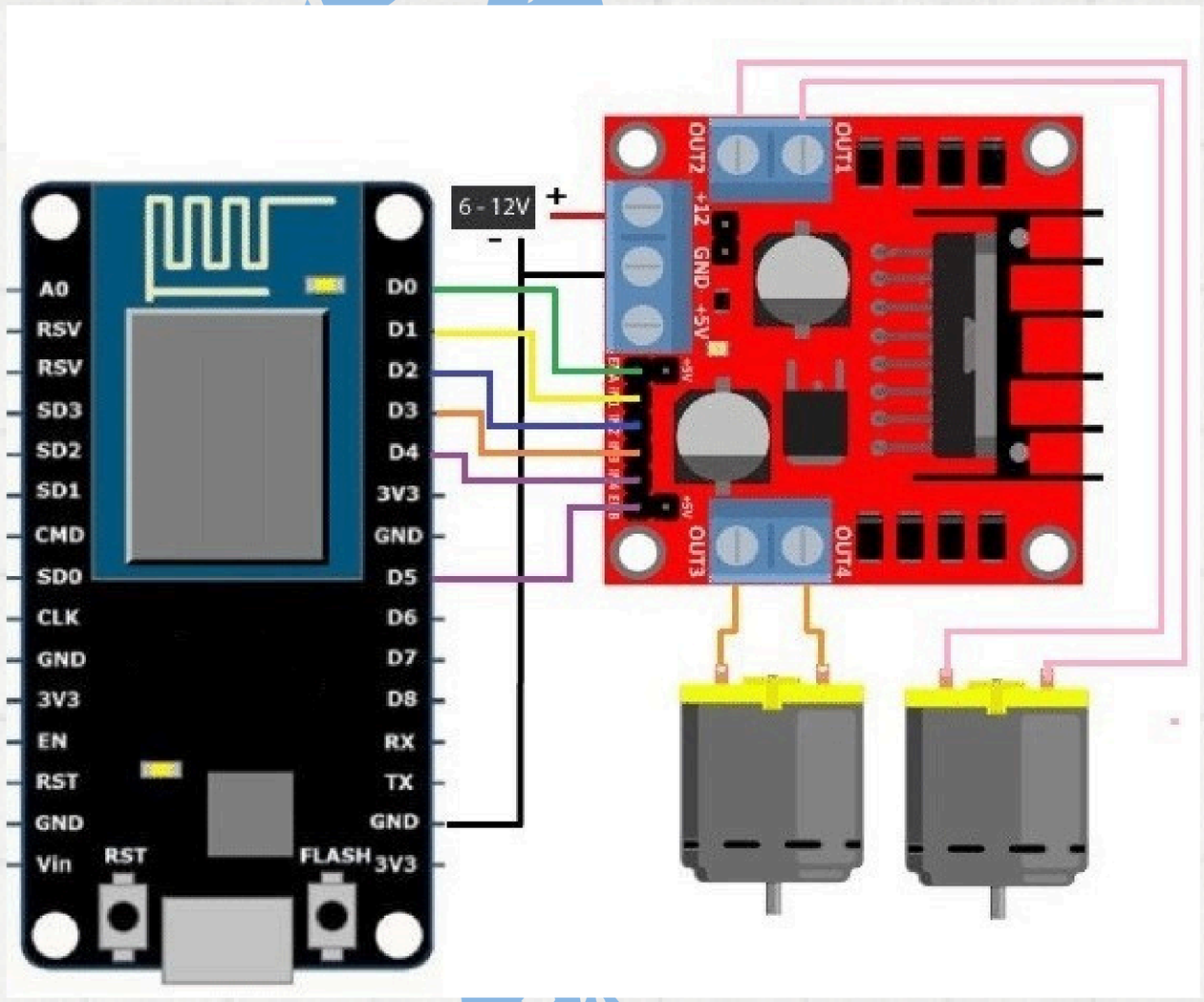
NodeMCU (ESP8266)

- Based on the ESP8266 or ESP32 microcontroller, which has built-in WiFi capability
- Generally has more RAM and flash memory compared to standard Arduino boards. For example, the ESP8266 typically has 4MB flash memory.
- Higher clock speed (e.g., 80 MHz or 160 MHz for ESP8266, up to 240 MHz for ESP32) and more processing power compared to Arduino UNO.



ROLL AND PITCH FROM MPU6050

```
int16_t ax, ay, az;

  // Get raw acceleration values
  mpu.getAcceleration(&ax, &ay, &az);

  // Convert raw values to 'g' values for accelerometer
  float accX = ax / 16384.0 ; // Assuming the accelerometer range is set to ±2g
  float accY = ay / 16384.0 ;
  float accZ = az / 16384.0 ;

  // Calculate pitch and roll angles
  float roll  = atan2(accY, accZ) * 180 / PI;
  float pitch = atan2(-accX, sqrt(accY * accY + accZ * accZ)) * 180 / PI;

  // Serial.print("Roll: "); Serial.print(roll);
  // Serial.print(" Pitch: "); Serial.print(pitch);

  // Convert pitch and roll angles to radians
  float roll_rad  = roll * PI / 180;
  float pitch_rad = pitch * PI / 180;


  delay(5);
  controlCar(roll_rad,pitch_rad);
```



POWERING DC MOTORS

We utilized the L293D motor driver to power the DC motors, facilitating the movement of the car.

CAR MOVEMENT BASED ON PITCH

USING THRESHOLD VALUES
OF PITCH AND ROLL

```
void controlCar(float roll, float pitch) {
  // Forward/Backward control based on Y-axis acceleration
  if (pitch > 0.2) { // Threshold value for forward motion
    moveForward();
  } else if (pitch < -0.2) { // Threshold value for backward motion
    moveBackward();
  } else {
    stopMoving();
  }

  // Left/Right control based on X-axis acceleration
  if (roll > 1.1) { // Threshold value for left turn
    turnRight();
  } else if (roll < 0.5) { // Threshold value for right turn
    turnLeft();
  }
}
```

DIRECTING CAR MOVEMENT

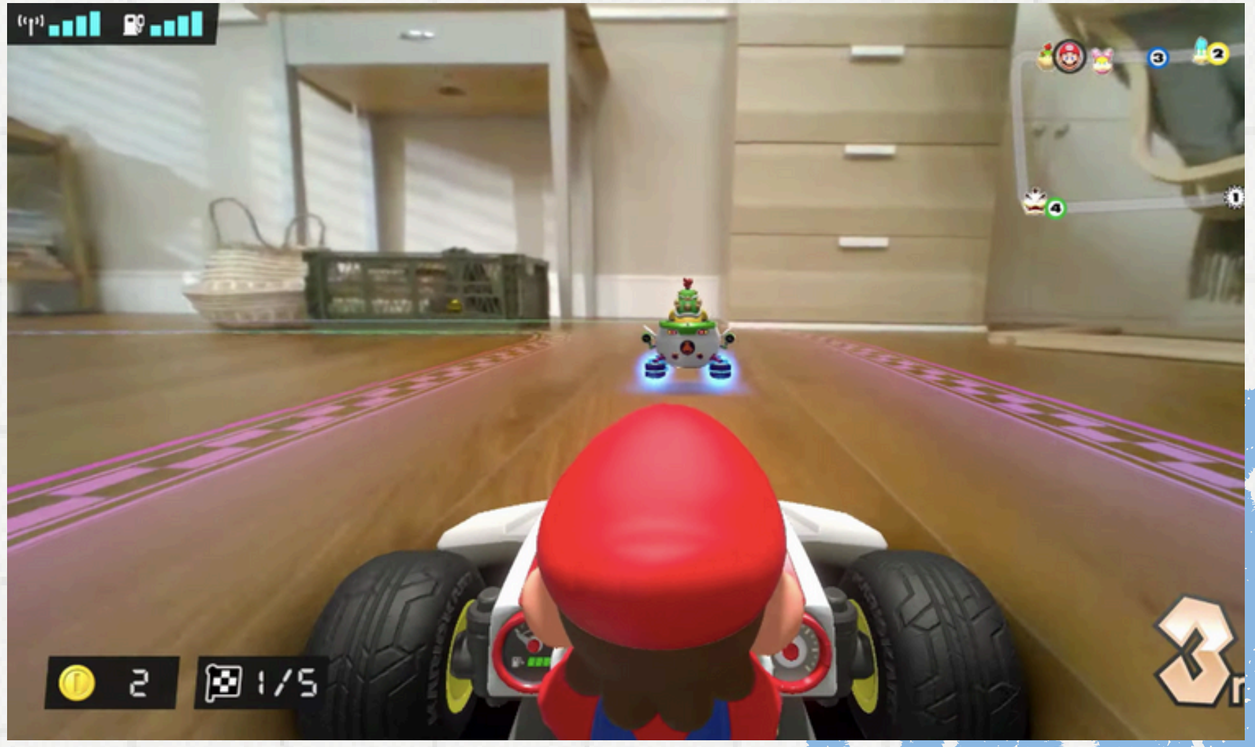
void moveForward() { Untitled-1

```
1   void moveForward() {
2     digitalWrite(IN1, HIGH);
3     digitalWrite(IN2, LOW);
4     digitalWrite(IN3, HIGH);
5     digitalWrite(IN4, LOW);
6     // Serial.println( )
7   }
8
```

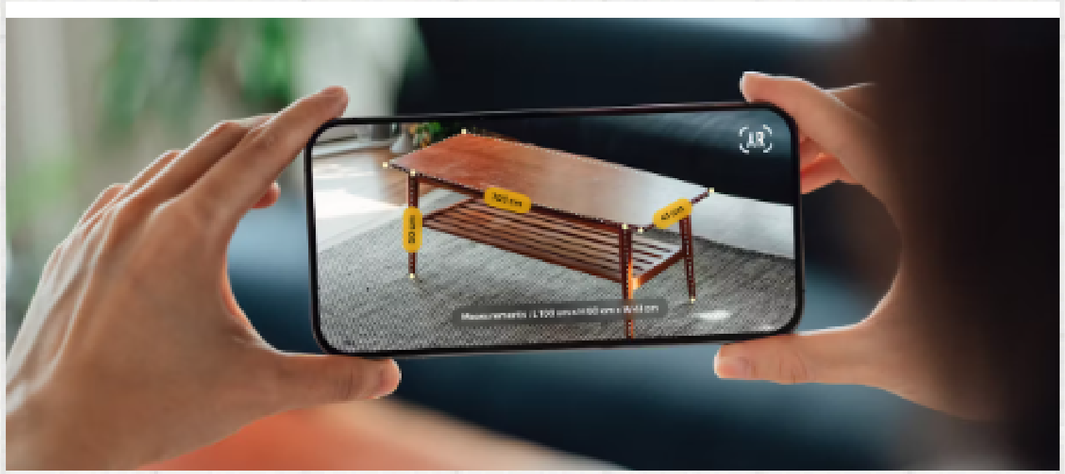
Part II : Computer Vision

Computer vision is a field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos.

To view our car in Augmented Reality (AR) and VR we aim to use Computer Vision on the camera footage on the car to detect obstacles in the road.



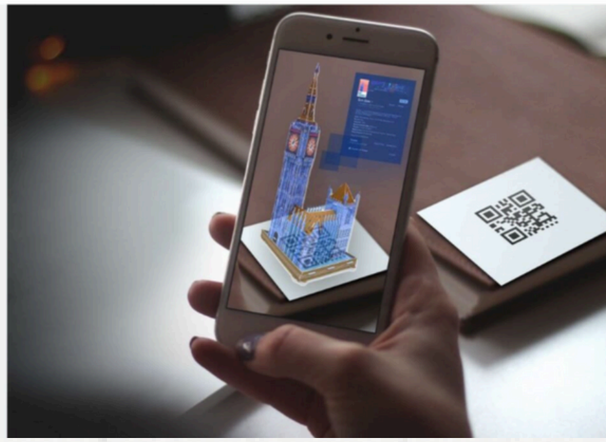
Augmented Reality (AR)



Augmented Reality (AR) is a technology that overlays digital information, such as images, sounds, or text, onto the real-world environment in real-time.

1. MARKER BASED AR

Marker-based AR uses specific visual markers, such as QR codes or unique patterns, to trigger the display of digital content. When a device's camera recognizes the marker, it overlays the corresponding digital information onto the real-world view.



Libraries used for Implementation

- CV2
- MATH
- ARGPARSE
- NUMPY

2. MARKERLESS AR

Markerless AR, also known as location-based or SLAM (Simultaneous Localization and Mapping) AR, uses sensors like GPS, accelerometers, and gyroscopes to overlay digital content onto the physical environment. This allows for a more flexible and immersive AR experience without needing predefined markers.

EXAMPLES OF MARKERLESS AR

Problem faced

Problems with Arduino IDE

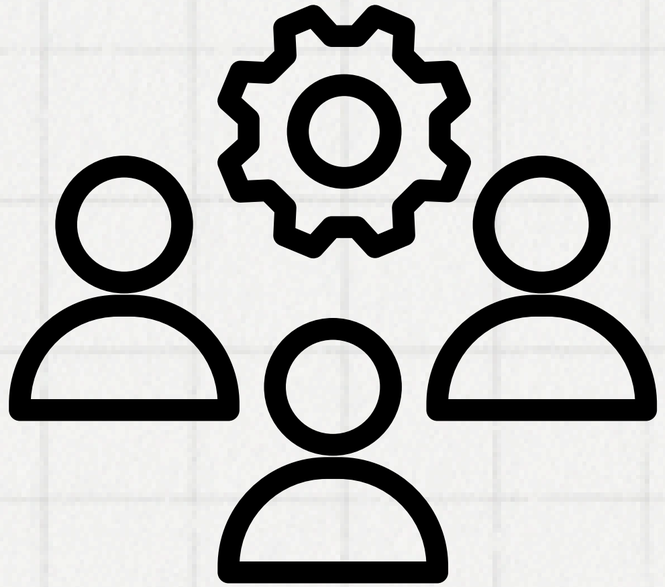
- Unable to upload code.
- Compilation error.
- Initially having trouble in understanding the code.

Problems with Hardware

- The unavailability of some necessary hardware components led to issues in testing the code.
- Loose connections between jumper wires.
- loose connection of cables
- Faulty components
- Insufficient battery power .

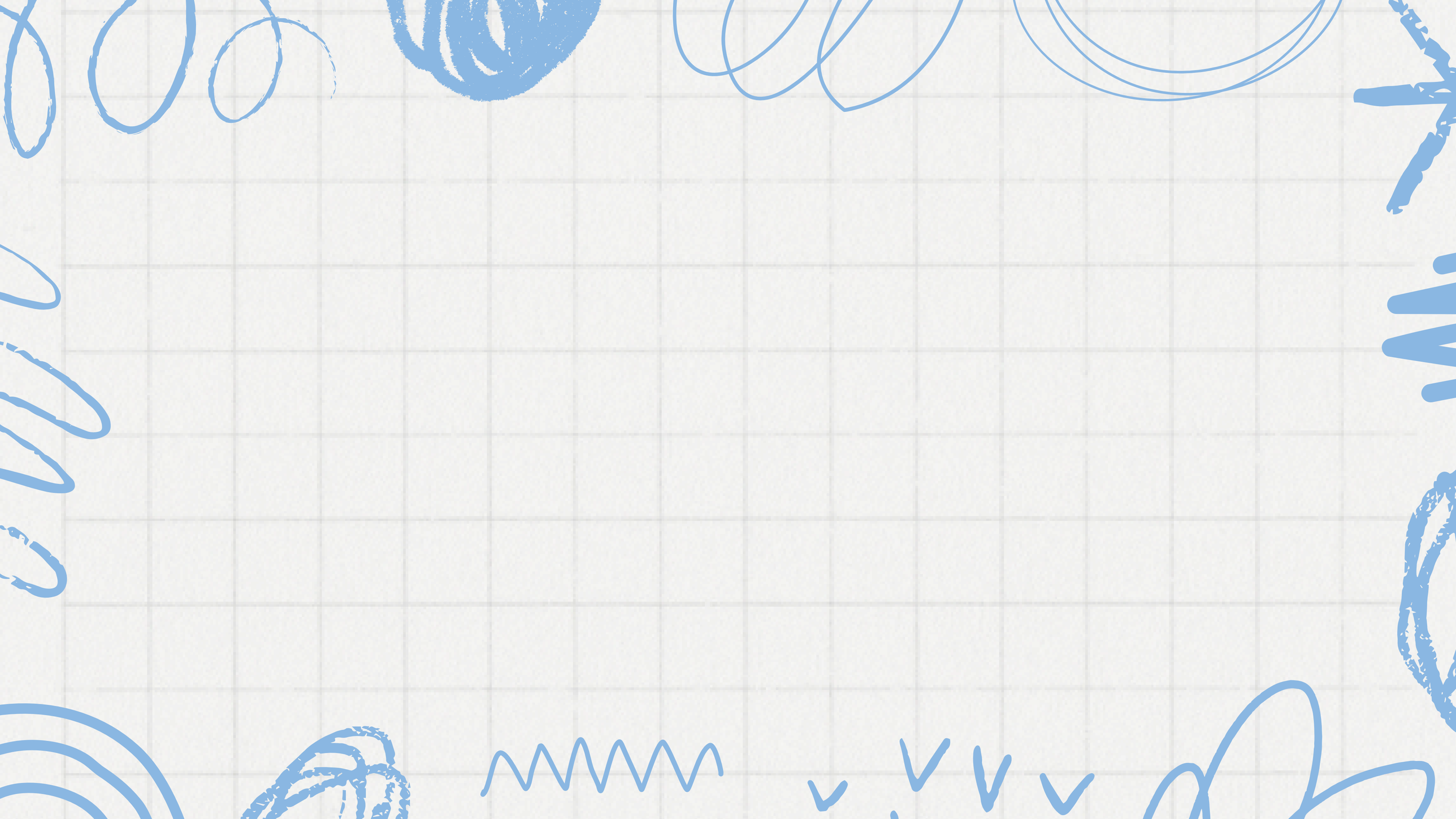
Contributors

Mentors : Devansh Bansal , Devansh Agrawal , Dhruv Mittal and Kanika



Mentees:

Akshit
Aniket
Aurav
Divyaman
Durbasmriti
Pratul
Priyanka
Ronav
Samarth
Shaurya
Shital
Vidhi



Serial Communication Between Two ESP8266 NodeMCU Boards (Wired)

Sender

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(" Sender Here");
  delay(1000);
}
```

Receiver

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String receivedData = Serial.readStringUntil('\n');
    Serial.println("Received: " + receivedData);
  }
}
```

WIRELESS COMMUNICATION BETWEEN TWO NODEMCUs

CLIENT CODE :

```
#include <ESP8266WiFi.h>
const char* ssid = "USERNAME";
const char* password = "12345678";
WiFiServer server(80);
WiFiClient client;
```

```
void setup() {
    Serial.begin(115200);
    Serial.println("\nConnecting to Wi-Fi...");
    Serial.print("SSID: ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi connected");
    Serial.print("NodeMCU IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println("Connecting to server...");
    while (!client.connect(serverIP, 80)) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nConnected to server");
}
```

```
void loop() {
    if (client.available()) {
        String message = client.readStringUntil('\r');
        Serial.println("Received message from server:");
        Serial.println(message);
    }

    if (Serial.available()) {
        String message = Serial.readStringUntil('\n');
        Serial.println("Sending message to server:");
        Serial.println(message);
        client.println(message);
    }
```

SERVER CODE:

```
#include <ESP8266WiFi.h>
const char* ssid = "USERNAME";
const char* password = "12345678";
WiFiServer server(80);
WiFiClient client;
```

```
void setup() {
    Serial.begin(115200);
    Serial.println("\nConnecting to Wi-Fi...");
    Serial.print("SSID: ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("\nWiFi connected");
    Serial.print("NodeMCU IP address: ");
    Serial.println(WiFi.localIP());
    server.begin();
    Serial.println("Server started\nReady to receive and send messages");
}
```

```
void loop() {
    if (!client || !client.connected()) {
        client = server.available();
        return;
    }
    if (client.available()) {
        String message = client.readStringUntil('\r');
        Serial.println("Received message from client:");
        Serial.println(message);

        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("");
        client.println("<h1>Message received by server!</h1>");
        Serial.println("Response sent to client\nWaiting for next message...");
    }
    if (Serial.available()) {
        String message = Serial.readStringUntil('\n');
        Serial.println("Sending message to client:");
        Serial.println(message);
        client.println(message);
    }
}
```