**Project: ESP32 Control Panel with Mobile App**

**Objective**: The task was to create a control panel in a mobile app, which sends direction values via Bluetooth to an ESP32. The ESP32 then receives the direction value through Bluetooth or WiFi and performs the corresponding action, such as moving a motor or a robot.

**Task Breakdown:**

1. **Mobile App Control Panel**: A simple mobile app interface was created using MIT App Inventor. This app features directional control buttons (Forward, Backward, Left, Right) that allow the user to send control signals to the ESP32 via Bluetooth or WiFi.
2. **Sending Direction Values**: When the user presses one of the control buttons in the mobile app, the corresponding direction value (e.g., "Forward", "Backward") is sent via Bluetooth or WiFi to the ESP32.
3. **Receiving Direction Values by ESP32**: The ESP32 is programmed to receive the direction values sent from the mobile app. It uses the Bluetooth or WiFi connection to communicate with the app.
4. **Using WiFi Instead of Bluetooth**: While the task initially mentioned Bluetooth, WiFi was used instead for better performance and easier setup. The ESP32 is connected to the local WiFi network, and the mobile app communicates with it using the ESP32’s local IP address.

**Steps to Execute the Task:**

1. **Set up the Mobile App**:
   * Create the control panel with directional buttons using MIT App Inventor.
   * Add Bluetooth or WiFi functionality to send direction values when a button is pressed.
2. **Configure the ESP32**:
   * Connect the ESP32 to the local WiFi network.
   * Program the ESP32 to receive direction values via the Bluetooth or WiFi connection.
   * Process the received direction value and take appropriate action (e.g., motor control).
3. **WiFi Setup**:
   * The ESP32 is connected to the WiFi network and receives direction values via HTTP requests.
   * The IP address of the ESP32 device can be found in the serial monitor after it successfully connects to the WiFi. For reference: "Use the device's local IP address, which can be found in the ESP32 serial monitor after successful connection to the WiFi network."
4. **Testing**:
   * Once the mobile app and ESP32 are properly set up, testing was conducted by sending different control commands from the app to the ESP32 and ensuring the motor or robot responded accordingly.
5. Code:

#include <WiFi.h>

#include <WebServer.h>

const char\* ssid = "YOUR\_SSID";

const char\* password = "YOUR\_PASSWORD";

WebServer server(80);

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

Serial.println();

Serial.print("Connecting to WiFi...");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println();

Serial.print("Connected! IP Address: ");

Serial.println(WiFi.localIP());

server.on("/move", HTTP\_GET, handleMove);

server.begin();

}

void handleMove() {

String direction = server.arg("direction");

Serial.print("Received Direction: ");

Serial.println(direction);

if (direction == "forward") {

Serial.println(" Forward");

} else if (direction == "backward") {

Serial.println(" Backward");

} else if (direction == "left") {

Serial.println(" Left");

} else if (direction == "right") {

Serial.println("Right");

} else {

Serial.println("Unknown Direction");

}

server.send(200, "text/plain", "Direction Received: " + direction);

}

void loop() {

server.handleClient();

}

1. **Output**:

1. Mobile App Interface:

The mobile app displays a simple control panel with the following buttons:

- Forward

- Backward

- Left

- Right

When any of these buttons is pressed, the app sends a direction value (e.g. "Forward", "Backward") to the ESP32.

2. ESP32 Connects to WiFi:

The ESP32 successfully connects to the local WiFi network. The output in the serial monitor will show:

Connecting to WiFi...

Connected! IP Address: 192.xxx.xx.xx

This indicates that the ESP32 is connected to the network and ready to receive commands.

3. Receiving Direction Value by ESP32:

Once the mobile app sends the direction command, the ESP32 receives the value via WiFi (or Bluetooth, depending on the configuration) and processes it. The serial monitor will display the received command.

For example:

Received Direction: Forward

This output will be displayed on the serial monitor of ESP32’s every time when a direction value is received from the mobile app.

4.Response Action:

Based on the received direction value, ESP32 performs the appropriate action. For example:

-If the direction is “Forward”, robot moves forward.

-If the direction is “Backward”, robot moves to back.

Response on serial monitor may be like:

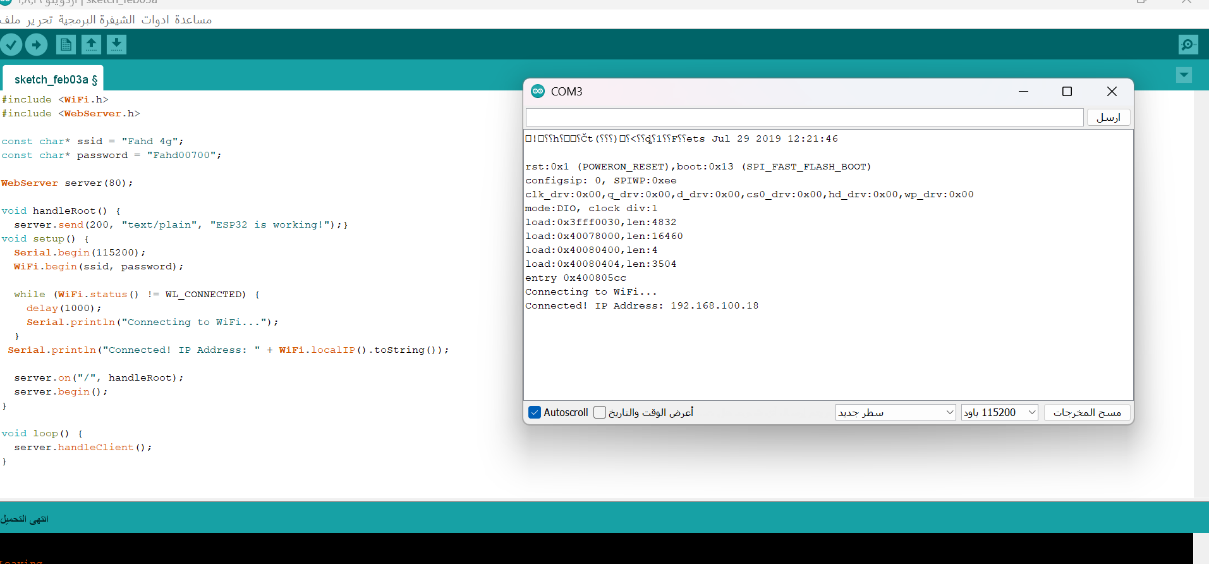
```

Moving Forward

```

**Result:**

* The mobile app successfully sends direction values to the ESP32.
* The ESP32 receives the commands and performs the corresponding action (e.g., moves a motor or robot).
* The system operates smoothly using WiFi instead of Bluetooth for better connectivity and ease of use.

A computer screen shot of a computer

AI-generated content may be incorrect.



