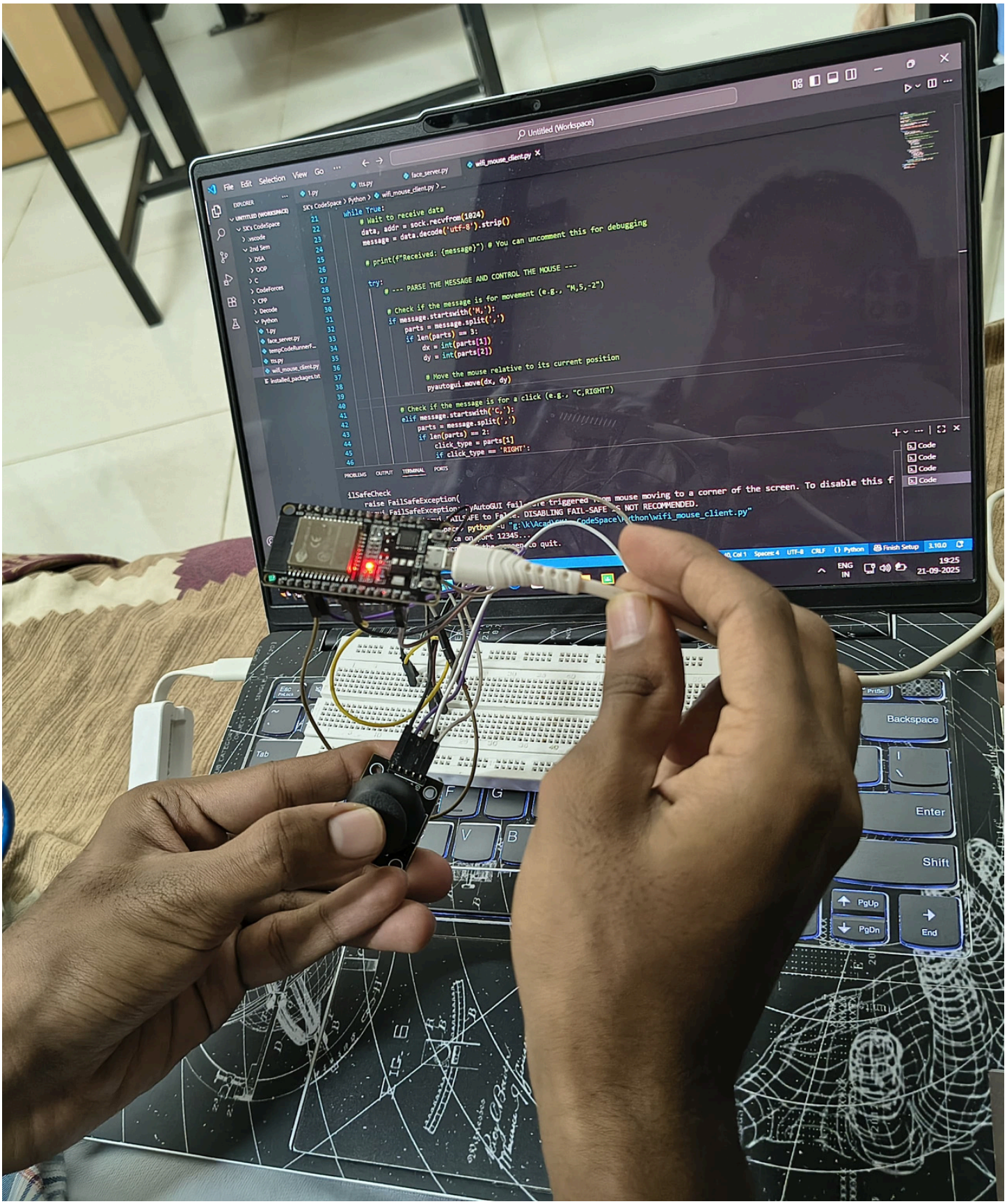


```
#include <WiFi.h>

#include <WiFiUdp.h>


// --- Wi-Fi Settings ---
```

```
const char* ssid = "ESP32_AirMouse_AP";
const char* password = "12345678";

// --- UDP Settings ---
WiFiUDP udp;
const int udpPort = 12345;

// Joystick and button pins
const int joyX = 34;
const int joyY = 35;
const int rightClickBtn = 25;

// --- Calibration Variables ---
// We will store the joystick's actual center position here
int centerX = 0;
int centerY = 0;

void setup() {
  Serial.begin(115200);
  pinMode(rightClickBtn, INPUT_PULLUP);

  // --- CALIBRATION ROUTINE ---
  Serial.println("Calibrating joystick... Do not touch it!");
```

```
long totalX = 0;
long totalY = 0;
// Read the joystick 100 times to get a stable average
for (int i = 0; i < 100; i++) {
  totalX += analogRead(joyX);
  totalY += analogRead(joyY);
  delay(5);
}
centerX = totalX / 100;
centerY = totalY / 100;
Serial.print("Calibration complete. Center X: ");
Serial.print(centerX);
Serial.print(", Center Y: ");
Serial.println(centerY);
// --- END CALIBRATION ---

Serial.print("Setting up Access Point...");
WiFi.softAP(ssid, password);

Serial.println("\nAP Started!");
Serial.print("IP Address: ");
Serial.println(WiFi.softAPIP());
```

```
  udp.begin(udpPort);
  Serial.print("Broadcasting UDP on port ");
  Serial.println(udpPort);
}

void loop() {
  int xRaw = analogRead(joyX);
  int yRaw = analogRead(joyY);

  // Normalize using the CALIBRATED center values
  int x = xRaw - centerX;
  int y = yRaw - centerY;

  // Apply a deadzone to prevent any minor remaining jitter
  if (abs(x) < 200) x = 0;
  if (abs(y) < 200) y = 0;

  // Scale and send
  int dx = x / 200;
  int dy = -y / 200; // Invert Y-axis

  char packetBuffer[32];
  if (dx != 0 || dy != 0) {
```

```
    snprintf(packetBuffer, sizeof(packetBuffer), "M,%d,%d\n", dx, dy);
  } else if (digitalRead(rightClickBtn) == LOW) {
    snprintf(packetBuffer, sizeof(packetBuffer), "C,RIGHT\n");
    delay(200); // Debounce
  } else {
    delay(20);
    return;
  }

  udp.beginPacket(WiFi.softAPBroadcastIP(), udpPort);
  udp.write((uint8_t*)packetBuffer, strlen(packetBuffer));
  udp.endPacket();


  delay(20);
}
```