#define BLYNK\_TEMPLATE\_ID "TMPL3iM-O34wW"

#define BLYNK\_TEMPLATE\_NAME "MAJOR PROJECT RC PLANE"

#define BLYNK\_AUTH\_TOKEN "WBKIzKfAe1L5-6o4Z3p0oqFfKstouXvI"

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#include <Wire.h>

#include <ESP32Servo.h>

#include <Adafruit\_BMP280.h>

#include <MPU6050.h>

#include <TinyGPS++.h>

#include <HardwareSerial.h>

// WiFi credentials

char ssid[] = "Mini-Project";

char pass[] = "uday8dewas";

// Virtual Pins

#define VPIN\_THROTTLE V1

#define VPIN\_RUDDER\_LEFT V2

#define VPIN\_RUDDER\_RIGHT V3

#define VPIN\_ELEVATOR V4

#define VPIN\_AILERON V5

#define VPIN\_GPS V6

// Servo & Sensors

Servo esc, elevator, aileron, rudder;

TinyGPSPlus gps;

HardwareSerial gpsSerial(1);

TwoWire bmpWire(1);

Adafruit\_BMP280 bmp(&bmpWire);

MPU6050 mpu;

// GPIO Pins

#define GPS\_RX 16

#define GPS\_TX 17

#define MPU\_SDA 21

#define MPU\_SCL 22

#define BMP\_SDA 25

#define BMP\_SCL 26

#define ESC\_PIN 32

#define ELEVATOR\_PIN 33

#define AILERON\_PIN 27

#define RUDDER\_PIN 14

// Control Variables

int throttleValue = 1000;

int elevValue = 90;

int ailValue = 90;

int elevInput = 0;

int ailInput = 0;

bool rudderLeft = false, rudderRight = false;

// Gyro Variables

float pitch = 0, roll = 0;

// Timing for auto/manual switch

unsigned long lastElevatorInputTime = 0;

unsigned long lastAileronInputTime = 0;

const unsigned long manualTimeout = 1000; // ms

// ========== BLYNK WRITE FUNCTIONS ==========

BLYNK\_WRITE(VPIN\_THROTTLE) {

  int input = param.asInt();

  throttleValue = map(input, 0, 100, 1000, 2000);

}

BLYNK\_WRITE(VPIN\_ELEVATOR) {

  int input = param.asInt();

  elevInput = input;

  elevValue = map(input, -50, 50, 60, 100);

  lastElevatorInputTime = millis();

}

BLYNK\_WRITE(VPIN\_AILERON) {

  int input = param.asInt();

  ailInput = input;

  ailValue = map(input, -50, 50, 65, 115);

  lastAileronInputTime = millis();

}

BLYNK\_WRITE(VPIN\_RUDDER\_LEFT) {

  rudderLeft = param.asInt();

}

BLYNK\_WRITE(VPIN\_RUDDER\_RIGHT) {

  rudderRight = param.asInt();

}

// ========== GYRO READING FUNCTION ==========

void readGyro() {

  int16\_t ax, ay, az, gx, gy, gz;

  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  pitch = atan2(ax, az) \* 180 / PI;

  roll = atan2(ay, az) \* 180 / PI;

}

// ========== GPS READING AND SENDING FUNCTION ==========

void sendGPS() {

  while (gpsSerial.available() > 0) {

    gps.encode(gpsSerial.read());

  }

  if (gps.location.isUpdated()) {

    float latitude = gps.location.lat();

    float longitude = gps.location.lng();

    String gpsData = String(latitude, 6) + "," + String(longitude, 6);

    Blynk.virtualWrite(VPIN\_GPS, gpsData);

    Serial.print("Latitude: ");

    Serial.print(latitude, 6);

    Serial.print(" | Longitude: ");

    Serial.println(longitude, 6);

  }

}

// ========== SETUP ==========

void setup() {

  Serial.begin(115200);

  Blynk.begin(BLYNK\_AUTH\_TOKEN, ssid, pass);

  // GPS

  gpsSerial.begin(9600, SERIAL\_8N1, GPS\_RX, GPS\_TX);

  // Sensors

  Wire.begin(MPU\_SDA, MPU\_SCL);

  mpu.initialize();

  bmpWire.begin(BMP\_SDA, BMP\_SCL);

  bmp.begin(0x76);

  // Servo/ESC setup

  esc.attach(ESC\_PIN);

  elevator.attach(ELEVATOR\_PIN);

  aileron.attach(AILERON\_PIN);

  rudder.attach(RUDDER\_PIN);

  // ESC arming sequence

  esc.writeMicroseconds(1000);

  delay(100);

  esc.writeMicroseconds(2000);

  delay(100);

  esc.writeMicroseconds(1000);

  delay(100);

}

// ========== LOOP ==========

void loop() {

  Blynk.run();

  sendGPS();

  readGyro();

  // Throttle control

  esc.writeMicroseconds(throttleValue);

  // === Elevator Control ===

  if ((millis() - lastElevatorInputTime > manualTimeout) && (abs(elevInput) < 5)) {

    int correction = map(pitch, -20, 20, 100, 80);

    correction = constrain(correction, 60, 100);

    elevator.write(correction);

  } else {

    elevator.write(elevValue);

  }

  // === Aileron Control ===

  if ((millis() - lastAileronInputTime > manualTimeout) && (abs(ailInput) < 5)) {

    int correction = map(roll, -20, 20, 105, 75);

    correction = constrain(correction, 65, 115);

    aileron.write(correction);

  } else {

    aileron.write(ailValue);

  }

  // Rudder Control

  if (rudderLeft) rudder.write(60);

  else if (rudderRight) rudder.write(120);

  else rudder.write(90);

  delay(50);

}