Basic Level Flight Control Code for this ESP32-S3 Programmable DIY Drone Kit, Ready-to-Fly FPV Coding Drone for Beginners & Makers | STEM Educational Quad-Copter with Wi-Fi/Bluetooth, Open-Source Flight Control, Solder-Free Assembly:-

Click to see full view

ESP32-S3 Programmable DIY Drone Kit, Ready-to-Fly FPV Coding Drone for Beginners & Makers | STEM Educational Quad-Copter with Wi-Fi/Bluetooth, Open-Source Flight Control, Solder-Free Assembly

Brand: ANU ELECTRONICS

Returns Policy | Secure transaction

Currently unavailable.

We don't know when or if this item will be back in stock.

Brand	ANU ELECTRONICS
Special Feature	Open-source platform for custom firmware development and programming, IMU for stable flight, supports Arduino and ESP-IDF programming
Colour	Black
Connectivity Technology	Wi-Fi
Item Weight	400 Grams

About this item

- Fully assembled and ready to fly, no soldering required
- Brainboard: ESP32-S3 with WiFi & plenty of GPIO for expansion
- Sensors: Integrated MPU6050 IMU (3-axis gyro + 3-axis accel) for stable flight
- Frame: Lightweight PCB structure doubles as the circuitry board — simple, elegant and hack-friendly

Below is a **very basic flight control sketch** you can use as a *starting point* for an ESP32-S3-based quadcopter like the LiteWing ESP32-S3 DIY Drone Kit (programmable kit using ESP32-S3). This code doesn't include advanced stabilization, but shows how to:

1. **Read IMU (MPU6050)** angles
2. **Convert to roll/pitch**
3. **Control motors with a simple PID loop**

 Important: Real drone flight control is safety-critical. This is *experimental* beginner code — test with props off until you're confident. The kit uses an MPU6050 IMU for attitude sensing (gyro+accel) and ESP32-S3 open-source flight firmware support is available via Crazyflie/ESP-Drone platforms.

---

1) Required Libraries & Wiring

Install these libraries in **Arduino IDE**:

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
```

Wiring:

- MPU6050 → ESP32-S3: SDA → GXIO21, SCL → GPIO22 (default I2C)
- 4 motors connected to PWM pins via proper motor drivers/ESCs
- Shared GND

---

2) Basic Flight Control Code (Arduino IDE)

This example reads IMU angles and runs a *simple PID stabilization loop* to control motor PWM.

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>

// ================ SETTINGS ================
const int MOTOR_FL = 14;  // front-left PWM pin
const int MOTOR_FR = 27;  // front-right
const int MOTOR_BL = 26;  // back-left
const int MOTOR_BR = 25;  // back-right

// PID tuning (very basic, needs real tuning per frame)
float Kp = 1.0;
float Ki = 0.0;
float Kd = 0.1;

// ================ GLOBALS ================
Adafruit_MPU6050 mpu;
float rollSetpoint = 0;
float pitchSetpoint = 0;
float rollErrorInt = 0, pitchErrorInt = 0;
float lastRollErr = 0, lastPitchErr = 0;

// simple PWM write (convert 0-255)
void setMotor(int pin, int val){
  ledcWrite(pin, constrain(val, 0, 255));
}

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // setup IMU
```

```
  if (!mpu.begin()) {
    Serial.println("MPU6050 not found!");
    while (1);
  }
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);

  // configure PWM channels
  ledcSetup(MOTOR_FL, 5000, 8);
  ledcSetup(MOTOR_FR, 5000, 8);
  ledcSetup(MOTOR_BL, 5000, 8);
  ledcSetup(MOTOR_BR, 5000, 8);
  ledcAttachPin(MOTOR_FL, MOTOR_FL);
  ledcAttachPin(MOTOR_FR, MOTOR_FR);
  ledcAttachPin(MOTOR_BL, MOTOR_BL);
  ledcAttachPin(MOTOR_BR, MOTOR_BR);

  Serial.println("Ready");
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // convert accel to simple roll/pitch
  float roll  = atan2(a.acceleration.y, a.acceleration.z) * 57.3;
  float pitch = atan2(-a.acceleration.x, a.acceleration.z) * 57.3;

  // — PID Controller for roll
  float rollErr = rollSetpoint - roll;
  rollErrorInt += rollErr * 0.01;
  float rollDeriv = (rollErr - lastRollErr)/0.01;
  float rollOutput = Kp*rollErr + Ki*rollErrorInt + Kd*rollDeriv;
  lastRollErr = rollErr;

  // — PID Controller for pitch
  float pitchErr = pitchSetpoint - pitch;
  pitchErrorInt += pitchErr * 0.01;
  float pitchDeriv = (pitchErr - lastPitchErr)/0.01;
  float pitchOutput = Kp*pitchErr + Ki*pitchErrorInt + Kd*pitchDeriv;
  lastPitchErr = pitchErr;

  // Motor mixing (simple plus config)
  int baseSpeed = 120;
  int mFL = baseSpeed + rollOutput + pitchOutput;
  int mFR = baseSpeed - rollOutput + pitchOutput;
  int mBL = baseSpeed + rollOutput - pitchOutput;
  int mBR = baseSpeed - rollOutput - pitchOutput;

  setMotor(MOTOR_FL, mFL);
  setMotor(MOTOR_FR, mFR);
  setMotor(MOTOR_BL, mBL);
  setMotor(MOTOR_BR, mBR);

  delay(10); // 100Hz loop
}
```

---

3) How It Works (High-Level)

 IMU: Reads accelerometer + gyro to estimate current roll/pitch.
 PID: Computes error between desired angle (hover) and measured angle.
 Motor Mix: Adjusts PWM on each motor to correct attitude.

---

4) Next Steps

This code is *very basic* and should be improved with:

- **True sensor fusion** (e.g., complementary or Kalman filter) to combine gyro+accel.
- **ESC calibration & proper PWM frequency** matching ESC/driver specs.
- **Safety checks / arming routine** before spinning motors.
- **Integration with radio control or Wi-Fi commands.**

For more advanced flight stacks, you can explore the **ESP-Drone open-source firmware** from Espressif that supports ESP32-S3 with stable/assistive modes based on Crazyflie logic.

How To Upload This Code To This Kit

For your **ESP32-S3 DIY Drone Kit**, uploading code is basically the same as any ESP32 Arduino project, but there are a few kit-specific details. Here's a **step-by-step guide**:

---

1. Install Arduino IDE / PlatformIO

- **Arduino IDE** (simpler for beginners):
  1. Download from Arduino.cc.
  2. Open **Preferences** → Additional Board Manager URLs → add:

     ```
     https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
     ```

  3. Open **Tools → Board → Board Manager**, search **ESP32**, install the latest "esp32 by Espressif Systems".

- **PlatformIO** (VSCode-based, better for advanced users):
  - Install **VSCode**, then install **PlatformIO IDE extension**.
  - Create a new ESP32-S3 project (select your ESP32-S3 board).

---

2. Connect the Drone to PC

1. Use **USB-C cable** (usually included) to connect the drone to your computer.
2. The ESP32-S3 has a built-in USB bootloader, so it will appear as a COM port (Windows) or `/dev/ttyUSB0` (Linux/macOS).

---

3. Configure Arduino IDE for ESP32-S3

- Go to **Tools → Board → ESP32 Arduino → ESP32S3 Dev Module** (or the exact model of your kit).
- Select **Tools → Port → COMx** (your device port).
- Set **Upload Speed** to 115200 (default works).

---

4. Upload the Code

1. Copy the flight control code I provided into a new **Arduino sketch**.
2. Click the **Upload button (→)** in Arduino IDE.
3. The ESP32-S3 will compile and flash your code.
4. Open **Serial Monitor** (115200 baud) to see debug prints (`Serial.println("Ready");` etc.).

 Important: Remove propellers or have the drone securely restrained when testing the first upload. Motors may spin immediately.