Let's build a **WiFi-controlled ESP32 drone with full flight control** step by step. I'll break it down carefully so it's safe, understandable, and complete.

Step 0: Safety First

Before doing anything:

- Always test **without propellers first**.
- Keep drone on a **secure mount** when testing PID loops.
- Use **LiPo safety rules** (don't short-circuit, don't overcharge, don't handle with bare hands).
- Wear **goggles** if you test motors spinning.

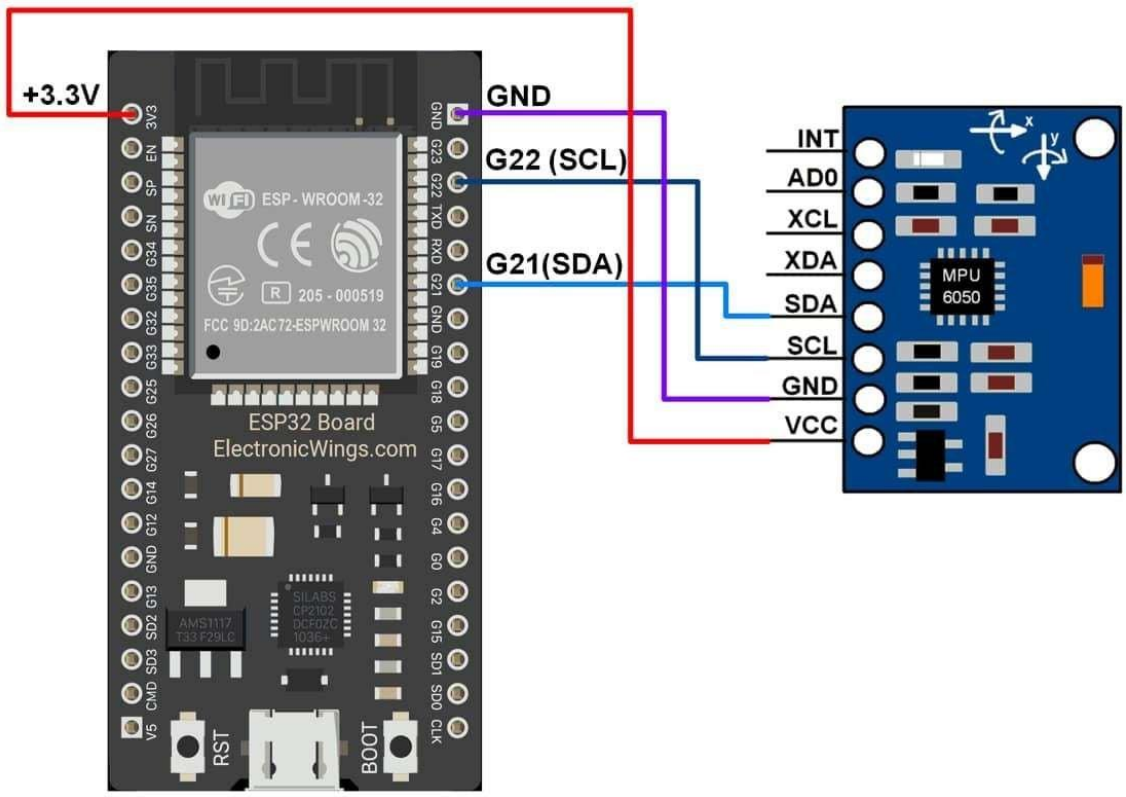
Step 1: Components

Component	Notes
ESP32 board (WROOM 32 / S3)	Brain
MPU6050	Gyro + accelerometer
4× Brushless motors	1000–1400KV for small drone
4× ESC (30A recommended)	PWM controlled
F450 frame (or similar)	Quadcopter frame
Propellers (2 CW, 2 CCW)	Spin direction matters
LiPo battery 3S 11.1V	Power supply
Buck converter 5V	Powers ESP32
Jumper wires / PCB / connectors	For connections

Step 2: Pin Connections

MPU6050 → ESP32 (I2C)

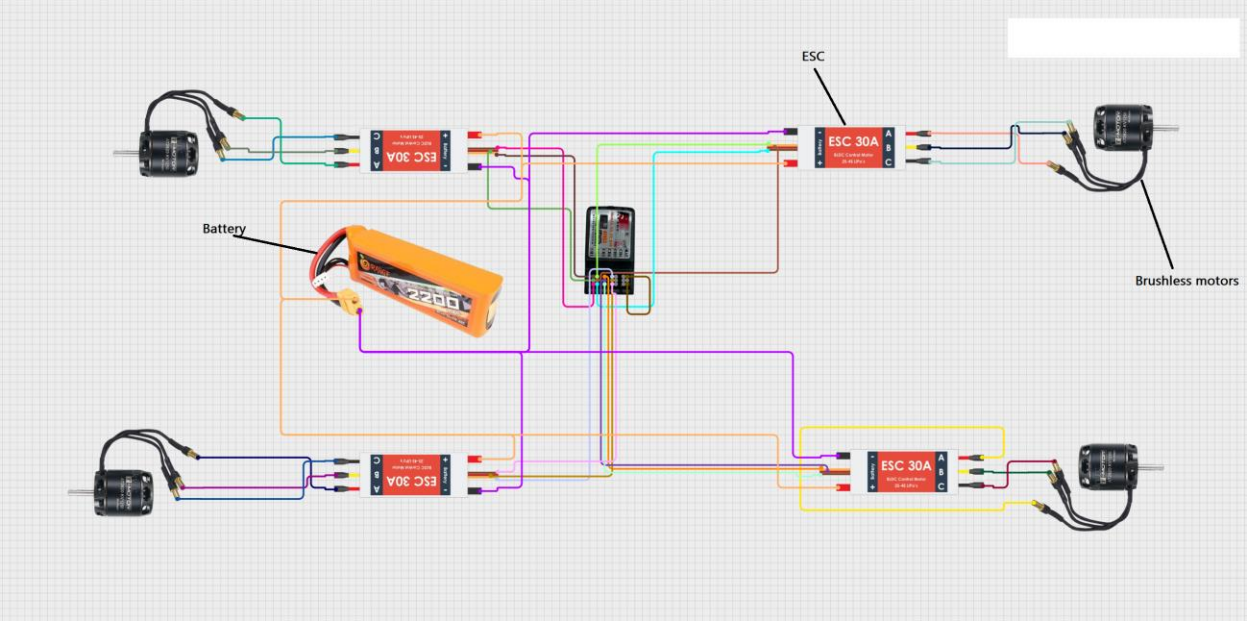
MPU6050	ESP32
VCC	3.3V
GND	GND
SDA	GPIO21
SCL	GPIO22



ESCs → ESP32 (PWM output)

ESC	ESP32 GPIO
ESC1	25
ESC2	26
ESC3	27
ESC4	14

 Important: Connect all grounds together: ESC + ESP32 + battery.



Power Setup

- LiPo → ESCs directly
- Buck converter → 5V → ESP32 VIN

Never connect LiPo directly to ESP32.

Step 3: Software Setup

1. Install **Arduino IDE**.
2. Add **ESP32 board support**:
   `File → Preferences → Additional Boards URL →`
   `https://dl.espressif.com/dl/package_esp32_index.json`
3. Install libraries:
   - `MPU6050` by Electronic Cats or Jeff Rowberg
   - `Wire.h` (built-in)
   - `WiFi.h` (built-in)
   - `ESP32Servo.h` (for ESC PWM signals)

Step 4: PID Control Concept

- Pitch (front/back), Roll (left/right), Yaw (rotation)
- PID formula:

```
Output = Kp*error + Ki*integral + Kd*derivative
```

- Error = desired angle − current angle

Example:
If drone tilts forward 5° → increase back motors, decrease front motors → stabilize.

Step 5: Full Working Code (ESP32)

Complete ESP32 Drone Code

```
#include <Wire.h>
#include <MPU6050.h>
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <ESP32Servo.h>
```

```
// -------------------- WiFi --------------------
const char* ssid = "ESP32_DRONE";
const char* password = "12345678";
AsyncWebServer server(80);

// -------------------- MPU6050 --------------------
MPU6050 mpu;

// -------------------- Motors --------------------
Servo m1, m2, m3, m4;
int motorPins[4] = {25, 26, 27, 14};

// -------------------- PID Variables --------------------
float kp = 5.0, ki = 0.0, kd = 2.0;
float rollSetpoint = 0, pitchSetpoint = 0, yawSetpoint = 0;
float rollError, pitchError, yawError, rollPrev=0, pitchPrev=0;
float rollIntegral=0, pitchIntegral=0;

// -------------------- Throttle --------------------
int baseThrottle = 1300; // Adjust for hover
int minSignal = 1000;    // ESC µs
int maxSignal = 2000;

// -------------------- HTML Page --------------------
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
  <head>
    <title>ESP32 Drone Control</title>
    <style>
      body { text-align:center; font-family: Arial; }
      button { width:80px; height:80px; font-size:20px; margin:10px; }
    </style>
  </head>
  <body>
    <h2>Drone Control</h2>
    <div>
      <button onclick="sendCommand('PITCH_UP')">↑ Pitch</button><br>
      <button onclick="sendCommand('ROLL_LEFT')">← Roll</button>
      <button onclick="sendCommand('THROTTLE_UP')">↑ Throttle</button>
      <button onclick="sendCommand('THROTTLE_DOWN')">↓ Throttle</button>
      <button onclick="sendCommand('ROLL_RIGHT')">→ Roll</button><br>
      <button onclick="sendCommand('PITCH_DOWN')">↓ Pitch</button><br>
      <button onclick="sendCommand('YAW_LEFT')">↺ Yaw</button>
      <button onclick="sendCommand('YAW_RIGHT')">↻ Yaw</button>
    </div>
    <script>
      function sendCommand(cmd){
        fetch("/"+cmd).then(resp=>resp.text()).then(data=>console.log(data));
      }
    </script>
  </body>
</html>
)rawliteral";

// -------------------- Setup --------------------
```

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

  // Attach motors
  m1.attach(motorPins[0], minSignal, maxSignal);
  m2.attach(motorPins[1], minSignal, maxSignal);
  m3.attach(motorPins[2], minSignal, maxSignal);
  m4.attach(motorPins[3], minSignal, maxSignal);

  // Initialize MPU6050
  Wire.begin();
  mpu.initialize();
  if(!mpu.testConnection()) Serial.println("MPU6050 connection failed");

  // WiFi AP
  WiFi.softAP(ssid, password);
  Serial.println("AP Started: Connect to " + String(ssid));

  // Serve HTML page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
  });

  // Command handlers
  server.on("/THROTTLE_UP", HTTP_GET, [](AsyncWebServerRequest *request){
    baseThrottle += 20; request->send(200,"text/plain","OK");
  });
  server.on("/THROTTLE_DOWN", HTTP_GET, [](AsyncWebServerRequest *request){
    baseThrottle -= 20; request->send(200,"text/plain","OK");
  });
  server.on("/PITCH_UP", HTTP_GET, [](AsyncWebServerRequest *request){
    pitchSetpoint += 2; request->send(200,"text/plain","OK");
  });
  server.on("/PITCH_DOWN", HTTP_GET, [](AsyncWebServerRequest *request){
    pitchSetpoint -= 2; request->send(200,"text/plain","OK");
  });
  server.on("/ROLL_LEFT", HTTP_GET, [](AsyncWebServerRequest *request){
    rollSetpoint -= 2; request->send(200,"text/plain","OK");
  });
  server.on("/ROLL_RIGHT", HTTP_GET, [](AsyncWebServerRequest *request){
    rollSetpoint += 2; request->send(200,"text/plain","OK");
  });
  server.on("/YAW_LEFT", HTTP_GET, [](AsyncWebServerRequest *request){
    yawSetpoint -= 2; request->send(200,"text/plain","OK");
  });
  server.on("/YAW_RIGHT", HTTP_GET, [](AsyncWebServerRequest *request){
    yawSetpoint += 2; request->send(200,"text/plain","OK");
  });

  server.begin();
}

// -------------------- Loop --------------------
void loop() {
  // 1. Read MPU6050
  int16_t ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

```
  // Convert to degrees (simplified)
  float roll  = atan2(ay, az) * 57.3;
  float pitch = atan2(-ax, sqrt(ay*ay + az*az)) * 57.3;

  // 2. PID calculations
  rollError = rollSetpoint - roll;
  pitchError = pitchSetpoint - pitch;

  rollIntegral += rollError * 0.01; // dt ~10ms
  pitchIntegral += pitchError * 0.01;

  float rollDerivative = (rollError - rollPrev)/0.01;
  float pitchDerivative = (pitchError - pitchPrev)/0.01;

  float rollOutput = kp*rollError + ki*rollIntegral + kd*rollDerivative;
  float pitchOutput = kp*pitchError + ki*pitchIntegral + kd*pitchDerivative;

  rollPrev = rollError;
  pitchPrev = pitchError;

  // 3. Motor mixing
  int m1Signal = constrain(baseThrottle + pitchOutput - rollOutput,
minSignal, maxSignal);
  int m2Signal = constrain(baseThrottle + pitchOutput + rollOutput,
minSignal, maxSignal);
  int m3Signal = constrain(baseThrottle - pitchOutput + rollOutput,
minSignal, maxSignal);
  int m4Signal = constrain(baseThrottle - pitchOutput - rollOutput,
minSignal, maxSignal);

  // 4. Send signals to motors
  m1.writeMicroseconds(m1Signal);
  m2.writeMicroseconds(m2Signal);
  m3.writeMicroseconds(m3Signal);
  m4.writeMicroseconds(m4Signal);

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

□ How to Test

1. Upload code to ESP32.
2. Connect phone to **ESP32_DRONE WiFi**.
3. Open browser → go to **192.168.4.1** → see joystick page.
4. Test **without propellers** first.
5. Watch serial monitor → PID values change as you press buttons.
6. Slowly attach propellers → test throttle **incrementally**.
7. Tune **kp/kd/ki** and `baseThrottle` for stable hover.

Let's go **step by step** so your Android phone can control the ESP32 drone via the browser joystick page.

Step 1: Power on the ESP32 Drone

1. Make sure your **ESP32 is powered via 5V** (from buck converter).
2. Make sure the **code we uploaded** (with WiFi softAP + joystick page) is running.
3. Ensure **ESCs are connected** but **propellers are not attached yet** for safety.

Step 2: Connect Your Phone to ESP32 WiFi

1. Open **Settings → WiFi** on your Android phone.
2. Look for the network:

```
ESP32_DRONE
```

3. Tap it and enter the password:

```
12345678
```

4. Wait until your phone **connects successfully**.

 Important: Disconnect your phone from any other WiFi; you must be on the ESP32 hotspot.

Step 3: Open the Browser

1. Open Chrome, Firefox, or any browser on your phone.
2. In the address bar, type:

```
192.168.4.1
```

This is the default IP for ESP32 softAP.

3. Press **Enter**.

Step 4: Load the Joystick Page

- If everything is working, you should see a **webpage with buttons**:

```
↑ Pitch   ← Roll   ↑ Throttle   ↓ Throttle   → Roll
↓ Pitch   ↺ Yaw   ↻ Yaw
```

- This is the **joystick page** we embedded in ESP32 code.

Step 5: Test Commands

1. Press ↑ **Throttle** → `baseThrottle` increases → motors should spin up (without props first).
2. Press ↓ **Throttle** → motors slow down.
3. Press ↑/↓ **Pitch** → changes pitch setpoint.
4. Press ←/→ **Roll** → changes roll setpoint.
5. Press ↺/↻ **Yaw** → changes yaw setpoint.

The PID loop in ESP32 code will automatically adjust motor speeds to stabilize the drone.

Step 6: Safety Testing

1. **No propellers first** → just hear motors respond to joystick.
2. Once confirmed, **attach propellers carefully**.
3. Start with **very low throttle**.
4. Observe movement → tune `kp`, `kd`, `baseThrottle` for stable hover.

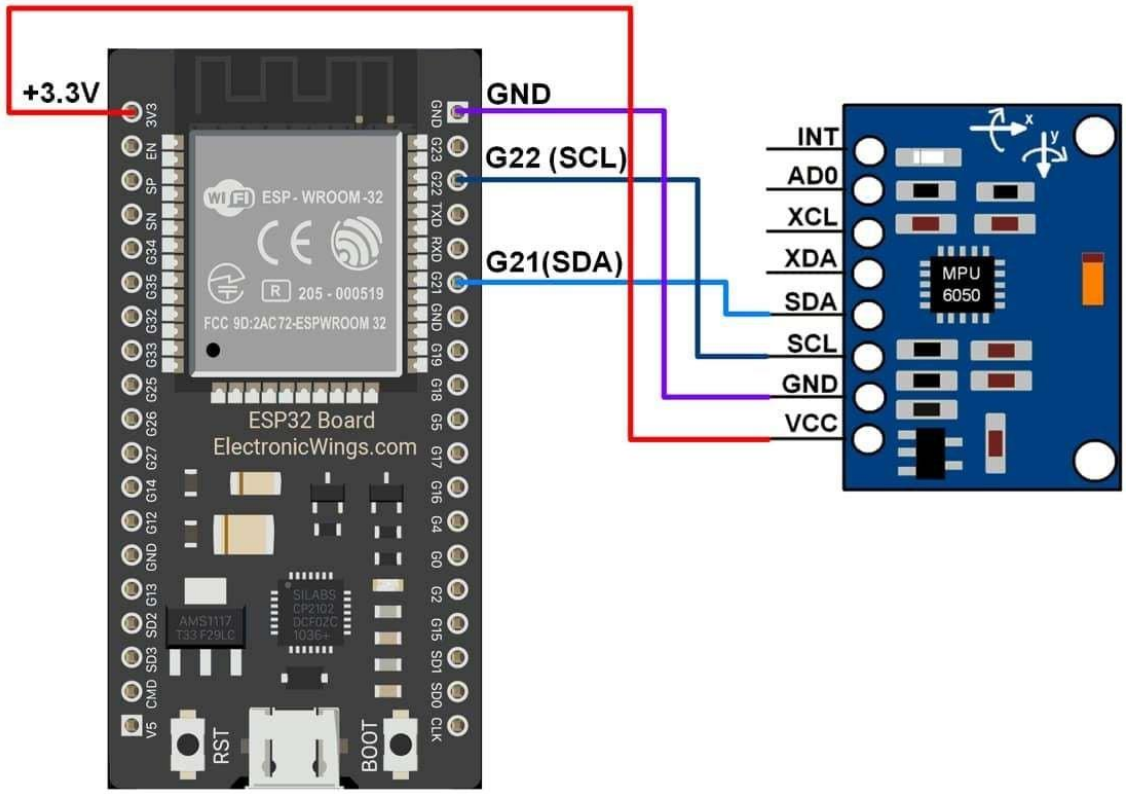
 Tip:
You can also **open browser in landscape mode** to make buttons easier to press.

Full wiring diagram for your ESP32 drone so you can connect everything safely and correctly. I'll explain each connection clearly.

ESP32 Drone Wiring Guide

1. ESP32 → MPU6050 (I2C Sensor for Stabilization)

MPU6050 Pin	ESP32 Pin	Notes
VCC	3.3V	Do **not** use 5V
GND	GND	Common ground
SDA	GPIO21	I2C Data
SCL	GPIO22	I2C Clock

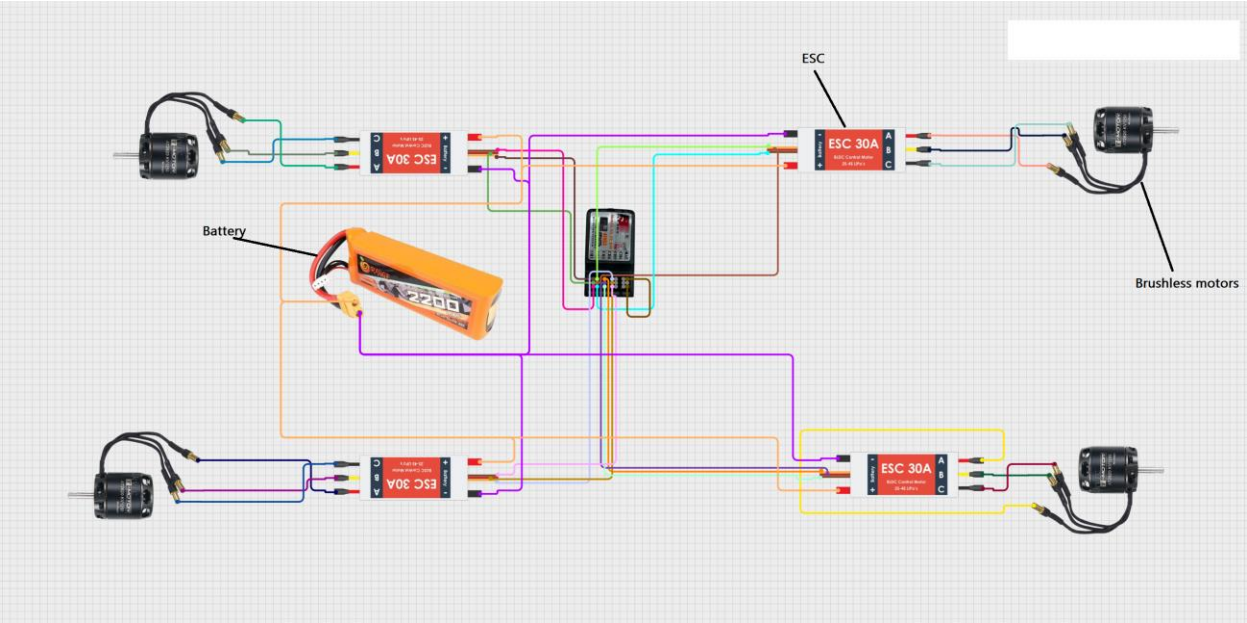


Purpose: Reads tilt angles (roll, pitch, yaw) for PID stabilization.

---

2. ESP32 → ESCs → Brushless Motors

ESC	ESP32 GPIO	Notes
ESC1 (Motor 1)	GPIO25	Front-left motor
ESC2 (Motor 2)	GPIO26	Front-right motor
ESC3 (Motor 3)	GPIO27	Back-right motor
ESC4 (Motor 4)	GPIO14	Back-left motor



Wiring Notes:

- Each ESC has **3 wires to motor** → connect to brushless motor.
- **Signal wire** → ESP32 GPIO (PWM).
- **Red wire (5V)** from ESC → leave disconnected if using buck converter.
- **Black wire (GND)** → connect to common ground with ESP32 and battery.

- **ESC to ESP32 connection**

- The **ESC "signal" wire** (usually **white or yellow**) goes to an **ESP32 PWM GPIO pin** like 25, 26, 27, 14, etc.
- **ESC "ground" wire** goes to ESP32 **GND** so both share the same reference.
- ESC "power" from battery **should not power the ESP32 directly** — use a buck converter for 5 V to the ESP32's Vin or 5 V pin.

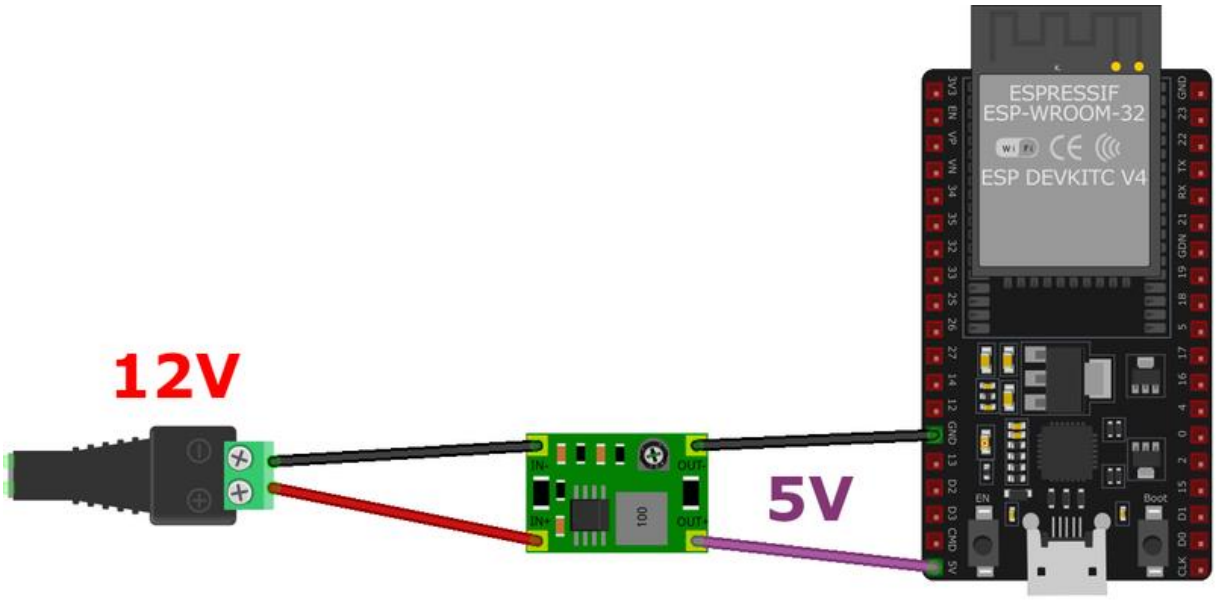
- **ESC output to BLDC motor**
  - The three ESC output wires go directly to the brushless motor (phases). The order determines direction — swapping any two reverses it.
- **Power Distribution**
  - Your LiPo battery powers the ESCs and motors.
  - A **buck converter** steps that battery voltage down to 5 V for your ESP32.

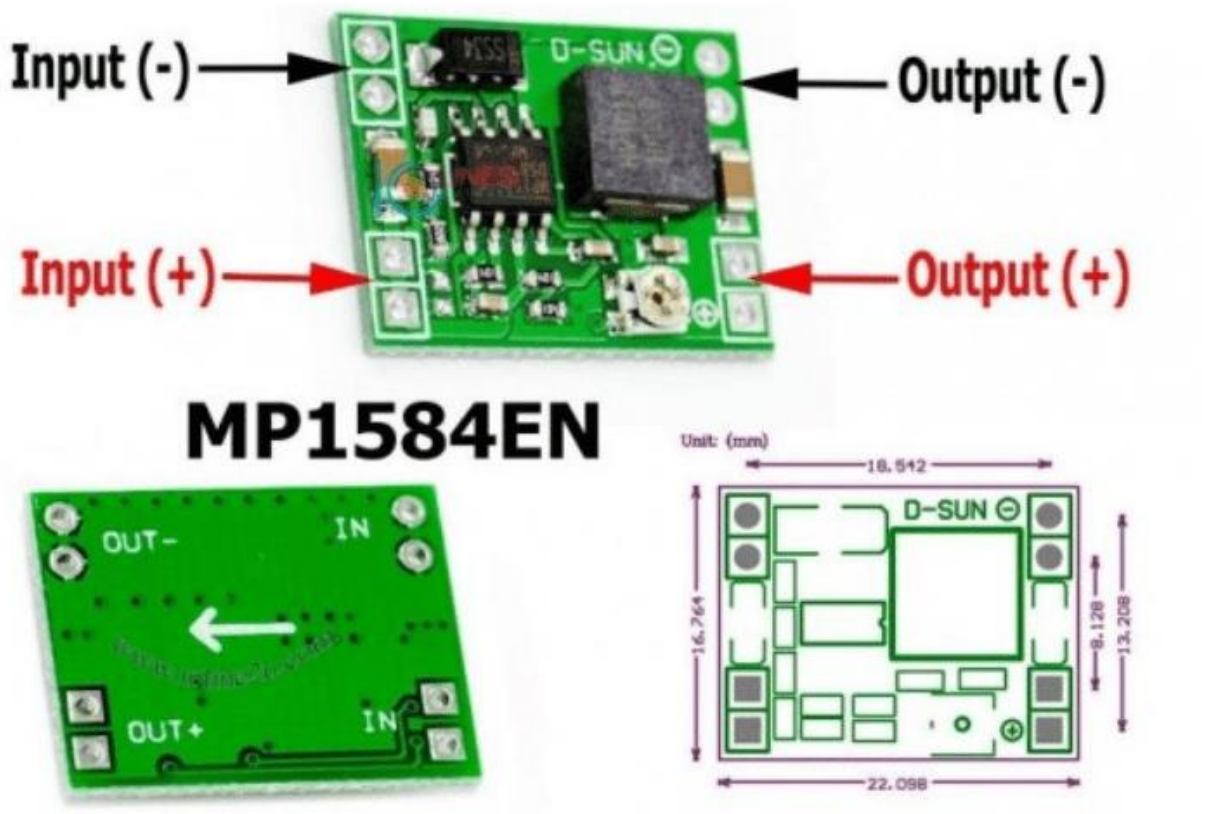
Typical ESC 3-Wire

- **Red = VCC (Battery)**
- **Black = GND**
- **White/Yellow = Signal (PWM)**

Connect **Signal** → ESP32 PWM GPIO
Connect **GND** → ESP32 GND
Do **not connect VCC directly to ESP32** from ESC (use buck converter)

Buck converter :-





- **MP1584 / LM2596-based DC-DC buck converter modules** — they take a higher DC voltage (like ~11–12 V from a LiPo) and output a stable **5 V** you can safely power your ESP32 with.
- They have **input and output terminals** clearly marked (IN+, IN– → source battery; OUT+, OUT– → ESP32 Vin and GND).
- The tiny screw potentiometer lets you **adjust the output voltage** precisely to 5 V before connecting your ESP32.

Quick Reminder on Wiring

1. **Battery (LiPo 11.1 V) → Buck Converter IN**
   - Battery `+` → Buck `IN+`
   - Battery `–` → Buck `IN–`
2. **Buck Converter OUT → ESP32 Power**
   - Buck `OUT+` (set to **5 V**) → ESP32 `Vin` or `5V` pin
   - Buck `OUT–` → ESP32 `GND`
3. **All Grounds Common**
   - Connect ESC ground, ESP32 ground, battery ground, and buck converter ground together for

---

3. Power Setup

LiPo Battery → ESCs → Motors

- Connect **battery directly to ESC power input**.
- **LiPo 3S (11.1V)** recommended.

Buck Converter → ESP32

- Vin of ESP32 → 5V output from buck converter.
- GND of ESP32 → GND from battery/ESC.

Important: All grounds (ESP32, ESCs, battery) must be connected.

---

4. Summary of Connections

```
Battery 3S
  ├─> ESC1 –> Motor1
  ├─> ESC2 –> Motor2
  ├─> ESC3 –> Motor3
  └─> ESC4 –> Motor4

ESC Signal Wires → ESP32 PWM
  ├─ ESC1 Signal → GPIO25
  ├─ ESC2 Signal → GPIO26
  ├─ ESC3 Signal → GPIO27
  └─ ESC4 Signal → GPIO14

MPU6050 → ESP32 I2C
  ├─ VCC → 3.3V
  ├─ GND → GND
  ├─ SDA → GPIO21
  └─ SCL → GPIO22

Buck Converter 5V → ESP32 Vin
GNDs connected together
```

---

5. Motor Orientation & Propellers

Motor	Position	Propeller
M1	Front-left	CW
M2	Front-right	CCW
M3	Back-right	CW
M4	Back-left	CCW

This ensures the drone can stabilize and rotate properly.

---

6. Safety Tips for Wiring

1. Double-check **all grounds are common**.
2. Keep **ESC power separate from ESP32 5V**.
3. Test **motors without propellers** first.
4. Ensure **propellers match motor rotation directions**.

Key Notes Before Wiring

✔ Use PWM pins on ESP32 for ESC signals (e.g., GPIO25, GPIO26, GPIO27, GPIO14).
✔ ESC expects **50 Hz servo-style PWM (1–2 ms pulse)** for throttle control.
✔ All grounds (ESC, ESP32, battery) **must be common**.
✔ Test without propellers first.