Low-Cost Self-Balancing Quadcopter Project Using Arduino

---

1. Overview

This document describes a minimal, cost-effective self-balancing quadcopter project using Arduino, designed as a summer side project to strengthen a technical resume.

---

2. Components List

- **Arduino Nano or Uno**
- **MPU6050 Gyroscope/Accelerometer**
- **HC-05 Bluetooth Module**
- **4 x Brushless Motors (e.g., 1000KV)**
- **4 x ESCs (Electronic Speed Controllers)**
- **3S LiPo Battery (11.1V)**
- **Propellers (2 CW + 2 CCW)**
- **Power Distribution Board or DIY wiring**
- **DIY frame (wood/PVC/aluminum)**
- **Zip ties, screws, glue**

---

3. Schematic Diagram

Description:

- MPU6050:
- VCC → 5V
- GND → GND
- SDA → A4
- SCL → A5
- ESC Signal Pins:
- ESC1 → D3 (Front Left - CW)
- ESC2 → D5 (Front Right - CCW)
- ESC3 → D6 (Back Right - CW)
- ESC4 → D9 (Back Left - CCW)
- HC-05 Bluetooth:

- VCC → 5V

- GND → GND
- TX → RX (via voltage divider)

- RX → TX

- Battery Powers ESCs & Arduino (via regulator or BEC)

---

4. Frame Design

- X-shaped layout using two 30 cm sticks
- Central acrylic/cardboard platform for electronics
- Components zip-tied securely
- Keep center of gravity at the center

---

5. Arduino Code (4-Motor PID Stabilization)

```
#include <Wire.h>
#include <PID_v1.h>
#include <MPU6050.h>

MPU6050 mpu;
double pitchInput, rollInput, pitchOutput, rollOutput;
double pitchSetpoint = 0, rollSetpoint = 0;

// PID gains (tune for your setup)
double Kp = 1.3, Ki = 0.04, Kd = 18.0;
PID pitchPID(&pitchInput, &pitchOutput, &pitchSetpoint, Kp, Ki, Kd, DIRECT);
PID rollPID(&rollInput, &rollOutput, &rollSetpoint, Kp, Ki, Kd, DIRECT);

int motorFL = 3;
int motorFR = 5;
int motorRR = 6;
int motorBL = 9;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
  pitchPID.SetMode(AUTOMATIC);
  rollPID.SetMode(AUTOMATIC);
  pitchPID.SetOutputLimits(-255, 255);
  rollPID.SetOutputLimits(-255, 255);
  pinMode(motorFL, OUTPUT);
  pinMode(motorFR, OUTPUT);
  pinMode(motorRR, OUTPUT);
  pinMode(motorBL, OUTPUT);
}

void loop() {
  int16_t ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  pitchInput = atan2(ax, az) * 180.0 / PI;
  rollInput = atan2(ay, az) * 180.0 / PI;

  pitchPID.Compute();
  rollPID.Compute();

  int baseThrottle = 1100;

  int m1 = constrain(baseThrottle + pitchOutput - rollOutput, 1000, 2000);
  int m2 = constrain(baseThrottle + pitchOutput + rollOutput, 1000, 2000);
  int m3 = constrain(baseThrottle - pitchOutput + rollOutput, 1000, 2000);
  int m4 = constrain(baseThrottle - pitchOutput - rollOutput, 1000, 2000);

  analogWrite(motorFL, map(m1, 1000, 2000, 0, 255));
  analogWrite(motorFR, map(m2, 1000, 2000, 0, 255));
  analogWrite(motorRR, map(m3, 1000, 2000, 0, 255));
  analogWrite(motorBL, map(m4, 1000, 2000, 0, 255));

  delay(10);
}
```

---

6. Resume Value

This project demonstrates:

- Embedded systems knowledge
- PID control implementation
- MPU6050 sensor integration
- Arduino development skills
- Power and motor control

---

7. Next Steps (Optional)

- Add Bluetooth-based smartphone control
- Introduce altitude hold with barometer
- GPS navigation (with Neo-6M)

Diagram file is available separately: [Drone Schematic Image]