# Final Code | PID Tuning

*Manual - 5*

This manual gives you **complete working code** for a self-balancing robot using **Arduino UNO + L293D + Adafruit MPU6050 library**.

---

# Installing the Adafruit Library

1. Open Arduino IDE → **Tools → Manage Libraries**
2. Search for **"Adafruit MPU6050"**
3. Install:
   - **Adafruit MPU6050**
   - It will also install **Adafruit BusIO**
4. Include in your sketch:

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

---

# Complementary Filter + PID Architecture

Same principles as before:

```
1. Read accelerometer + gyro
2. Compute tilt angle (complementary filter)
3. Run PID controller
4. Map PID output → motor speed/direction
5. Send PWM to motors
```

**Complementary filter equation:**

```
angle = 0.98 * (angle + gyroRate * dt) + 0.02 * accelAngle;
```

---

# Arduino Pin Setup (L293D)

```
// Motor Pins
#define IN1 5
#define IN2 6
#define ENA 9  // PWM

#define IN3 10
#define IN4 11
#define ENB 3  // PWM
```

---

# Complete Arduino Code Using Adafruit MPU6050

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

Adafruit_MPU6050 mpu;

// PID constants
float Kp = 30.0, Ki = 1.0, Kd = 1.0;

// PID variables
float error, prevError = 0, I = 0, D, PIDout;

// Timing
unsigned long lastTime;
float dt;

// Angle
float angle;

// Motor pins
#define IN1 5
#define IN2 6
#define ENA 9

#define IN3 10
#define IN4 11
#define ENB 3

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

  // Motor pins
  pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);

  // Initialize MPU6050
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) { delay(10); }
  }

  // Configure accelerometer and gyro
  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  lastTime = micros();
}

void loop() {
  // Calculate dt
  unsigned long now = micros();
  dt = (now - lastTime) / 1000000.0;
  lastTime = now;
```

```
  // Read sensor
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  // Accelerometer angle (pitch)
  float accAngle = atan2(a.acceleration.y, a.acceleration.z) * 57.2958;

  // Gyro rate
  float gyroRate = g.gyro.x * 57.2958; // rad/s → deg/s

  // Complementary filter
  angle = 0.98 * (angle + gyroRate * dt) + 0.02 * accAngle;

  // PID computation
  error = 0 - angle; // target = 0°
  I += Ki * error * dt;
  D = Kd * (error - prevError) / dt;
  PIDout = Kp * error + I + D;
  prevError = error;

  // Motor control
  moveMotor(PIDout);

  Serial.println(angle);
}

// Motor control function
void moveMotor(float pid) {
  int speed = constrain(abs(pid), 0, 255);

  if (pid > 0) {
    // Forward
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    analogWrite(ENA, speed);

    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
    analogWrite(ENB, speed);
  } else {
    // Backward
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    analogWrite(ENA, speed);

    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
    analogWrite(ENB, speed);
  }
}
```

---

# PID Tuning Guide (Adafruit MPU Version)

## Step 1 — Tune P

- Start with `Ki = 0, Kd = 0`
- Increase Kp slowly until robot stands but oscillates slightly

## Step 2 — Tune D

- Add damping with Kd
- Reduce oscillation without slowing response

### Step 3 — Tune I

- Correct slow lean
- Increase Ki until long-term drift is corrected

Always start small values and increase gradually.
Adafruit MPU6050 library gives **stable readings**, so tuning is smoother.

---

# Troubleshooting

| Symptom | Cause | Fix |
|---|---|---|
| Robot falls forward/back | PID output too low | Increase Kp |
| Shaking / jitter | Kp too high / frame shakes | Add damping (Kd), rigid frame |
| Drifting slowly | Ki too low | Increase Ki |
| Motors not moving | Wrong wiring / ENA/B PWM | Check wiring, polarity |
| Angle readings wrong | MPU orientation wrong | Mount MPU correctly, verify axes |

---

# Final Checklist

- MPU6050 powered by Arduino 5V
- MPU mounted rigidly, away from motors
- Motors low-backlash, wheels same size
- Battery fully charged (7.4V Li-ion recommended)
- All grounds common
- Control loop runs fast (~200 Hz)
- PID tuned gradually

---

# Summary

- Using **Adafruit MPU6050 library** simplifies setup and data reading
- Complementary filter gives a stable angle
- PID control stabilizes the robot

- L293D drives motors based on PID output
- Tuning PID gradually ensures robot balances smoothly

After following this manual, your self-balancing robot should **stand upright and respond to tilts reliably**.

---