// Updated MPU6050 IMU code for ROS

#include <Wire.h>

#include <math.h>

const int MPU6050\_ADDR = 0x68;

int16\_t accelX, accelY, accelZ;

int16\_t gyroX, gyroY, gyroZ;

float angleX = 0, angleY = 0;

float biasX = 0, biasY = 0;

float P[2][2] = {{1, 0}, {0, 1}};

float Q\_angle = 0.001, Q\_bias = 0.003, R\_measure = 0.03;

unsigned long lastTime = 0;

void setup() {

Serial.begin(115200);

Wire.begin();

Wire.beginTransmission(MPU6050\_ADDR);

Wire.write(0x6B);

Wire.write(0);

Wire.endTransmission(true);

delay(100);

lastTime = millis();

}

float kalmanFilter(float newAngle, float newRate, float dt, float &angle, float &bias) {

float rate = newRate - bias;

angle += dt \* rate;

P[0][0] += dt \* (dt \* P[1][1] - P[0][1] - P[1][0] + Q\_angle);

P[0][1] -= dt \* P[1][1];

P[1][0] -= dt \* P[1][1];

P[1][1] += Q\_bias \* dt;

float S = P[0][0] + R\_measure;

float K[2] = {P[0][0] / S, P[1][0] / S};

float y = newAngle - angle;

angle += K[0] \* y;

bias += K[1] \* y;

float P00\_temp = P[0][0];

float P01\_temp = P[0][1];

P[0][0] -= K[0] \* P00\_temp;

P[0][1] -= K[0] \* P01\_temp;

P[1][0] -= K[1] \* P00\_temp;

P[1][1] -= K[1] \* P01\_temp;

return angle;

}

void loop() {

unsigned long currentTime = millis();

float dt = (currentTime - lastTime) / 1000.0;

lastTime = currentTime;

Wire.beginTransmission(MPU6050\_ADDR);

Wire.write(0x3B);

Wire.endTransmission(false);

Wire.requestFrom(MPU6050\_ADDR, 14, true);

accelX = (Wire.read() << 8 | Wire.read());

accelY = (Wire.read() << 8 | Wire.read());

accelZ = (Wire.read() << 8 | Wire.read());

Wire.read(); Wire.read(); // Skip temp

gyroX = (Wire.read() << 8 | Wire.read());

gyroY = (Wire.read() << 8 | Wire.read());

gyroZ = (Wire.read() << 8 | Wire.read());

float ax = accelX / 16384.0;

float ay = accelY / 16384.0;

float az = accelZ / 16384.0;

float gx = gyroX / 131.0;

float gy = gyroY / 131.0;

float gz = gyroZ / 131.0;

float accelAngleX = atan2(ay, az) \* 180 / PI;

float accelAngleY = atan2(-ax, az) \* 180 / PI;

angleX = kalmanFilter(accelAngleX, gx, dt, angleX, biasX);

angleY = kalmanFilter(accelAngleY, gy, dt, angleY, biasY);

// Format: ax,ay,az,gx,gy,gz,roll,pitch

Serial.print("D,");

Serial.print(ax); Serial.print(",");

Serial.print(ay); Serial.print(",");

Serial.print(az); Serial.print(",");

Serial.print(gx); Serial.print(",");

Serial.print(gy); Serial.print(",");

Serial.print(gz); Serial.print(",");

Serial.print(angleX); Serial.print(",");

Serial.println(angleY);

delay(50);

}