#include <PID\_v1.h>

#include <LMotorController.h>

#include "I2Cdev.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

#include <MPU6050.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(12, 11);

int16\_t ax, ay, az; //ivme tanımlama

int16\_t gx, gy, gz; //gyro tanımlama

MPU6050 ivme\_sensor;

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

#include "Wire.h"

#endif

#define MIN\_ABS\_SPEED 1 // Minumum Başlangıç Hızı

MPU6050 mpu;

// MPU control/status vars

bool dmpReady = false;  //DMP init başarılı olmuşsa doğru olarak ayarlayın.

uint8\_t mpuIntStatus;   // MPU'dan gerçek kesme durumu baytı tutar

uint8\_t devStatus;      // Her cihaz çalışmasından sonra dönüş durumu (0 = başarı,! 0 = hata) (0 = success, !0 = error)

uint16\_t packetSize;    // Beklenen DMP paket boyutu (varsayılan 42 bayt)

uint16\_t fifoCount;     // Şu anda FIFO'daki tüm baytların sayısı

uint8\_t fifoBuffer[128]; // FIFO depolama arabelleği

// yönlendirme / hareket var

Quaternion q; // [w, x, y, z]  Kuarterna konteyner

VectorFloat gravity; // [x, y, z] Yerçekimi vektörü

float ypr[3]; // [yaw, pitch, roll] Yaw / pitch Adım/ roll rulo container ve yerçekimi vektörü

//PID

double originalSetpoint = 173;

double setpoint = originalSetpoint;

double movingAngleOffset = 0.3;

double input, output;

//bu değerleri ben seçtim

double Kp = 50;

double Kd = 1;

double Ki = 65;

PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

double motorSpeedFactorLeft = 0.53;

double motorSpeedFactorRight = 0.43;

//MOTOR SURUCU

int ENA = 5;

int IN1 = 6;

int IN2 = 7;

int IN3 = 8;

int IN4 = 9;

int ENB = 10;

LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);

volatile bool mpuInterrupt = false; // MPU kesme pininin yüksek olup olmadığını gösterir.

void dmpDataReady()

{

mpuInterrupt = true;

}

void setup()

{

Serial.begin(9600);

// I2C veriyoluna katıl (I2Cdev kütüphanesi bunu otomatik olarak yapmaz)

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

Wire.begin();

TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)

#elif I2CDEV\_IMPLEMENTATION == I2CDEV\_BUILTIN\_FASTWIRE

Fastwire::setup(400, true);

#endif

mpu.initialize();

devStatus = mpu.dmpInitialize();

// // burada kendi gyro ofsetlerini, minimum hassasiyet için ölçeklendirdim

mpu.setXGyroOffset(150);

mpu.setYGyroOffset(50);

mpu.setZGyroOffset(-100);

mpu.setZAccelOffset(1688); // 1688 factory default for my test chip

// Çalıştığından emin ol (eğer öyleyse 0 döndürür)

if (devStatus == 0)

{

// turn on the DMP, now that it's ready Hazır olduğunda, DMP'yi aç

mpu.setDMPEnabled(true);

// enable Arduino interrupt detection Arduino kesme algılamasını etkinleştir

attachInterrupt(0, dmpDataReady, RISING);

mpuIntStatus = mpu.getIntStatus();

// Ana döngü () işlevi onu kullanmanın iyi olduğunu bilmesi için DMP Hazır bayrağını ayarla

use it

dmpReady = true;

// get expected DMP packet size for later comparison  Daha sonra karşılaştırmak için beklenen DMP paket boyutunu elde edin

packetSize = mpu.dmpGetFIFOPacketSize();

//setup PID

pid.SetMode(AUTOMATIC);

pid.SetSampleTime(5);

pid.SetOutputLimits(-255, 255);

}

else

{

// HATA !

// 1 = başlangıç ​​bellek yükü başarısız

       // 2 = DMP yapılandırma güncellemeleri başarısız oldu

       // (eğer kırılırsa, genellikle kod 1 olur)

Serial.print(F("DMP Initialization failed (code "));

Serial.print(devStatus);

Serial.println(F(")"));

}

}

void loop()

{

char u=Serial.read();

if(u=='a'){

output=255;

motorController.move(output,120);

delay(10);

pid.Compute();

motorController.move(output, MIN\_ABS\_SPEED);

}

if(ax<15000){

if(ax>-15000){

ivme\_sensor.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

Serial.println(input-172);

// MPU kesmesi veya mevcut ek paket (ler) i bekle

while (!mpuInterrupt && fifoCount < packetSize)

{

if(u=='a'){

output=255;

motorController.move(output,120);

delay(1);

}

else{

pid.Compute();

motorController.move(output, MIN\_ABS\_SPEED);

delay(1);

}

}

// // kesme işaretini sıfırla ve INT\_STATUS baytını al

mpuInterrupt = false;

mpuIntStatus = mpu.getIntStatus();

// mevcut FIFO sayısını al

fifoCount = mpu.getFIFOCount();

// taşma olup olmadığını kontrol et (bu bizim kodumuz çok verimsiz olduğu sürece asla gerçekleşmemelidir)

if ((mpuIntStatus & 0x10) || fifoCount == 1024)

{

// sıfırla, böylece temiz bir şekilde devam edebiliriz

mpu.resetFIFO();

Serial.println(F("Devrildi!"));

// Aksi takdirde, DMP veri hazır interrupt'ını kontrol edin (bu sık sık yapılmalıdır)

}

else if (mpuIntStatus & 0x02)

{

// Doğru kullanılabilir veri uzunluğunu bekle, ÇOK kısa bir bekleyiş olmalı

while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

// FIFO'dan bir paket oku

mpu.getFIFOBytes(fifoBuffer, packetSize);

// > 1 paketin mevcut olması halinde FIFO sayısını buradan takip edin

// (Bu, bir kesinti beklemeden hemen okuyabilmemizi sağlar)

fifoCount -= packetSize;

mpu.dmpGetQuaternion(&q, fifoBuffer);

mpu.dmpGetGravity(&gravity, &q);

mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

input = ypr[1] \* 180/M\_PI + 180;

}

if(u=='a'){

output=255;

motorController.move(output,120);

delay(1);

}

}

else{

digitalWrite(5,LOW);

digitalWrite(10,LOW);

Serial.println(input-172);

Serial.println("DEVRİLDİ !");

delay(4000);

ax=0;

}

}

else{

digitalWrite(5,LOW);

digitalWrite(10,LOW);

Serial.println(input-172);

Serial.println("DEVRİLDİ !");

delay(4000);

ax=0;

}

if(u=='a'){

output=255;

motorController.move(output,120);

delay(1);

}

}