**LAMPIRAN**

```
#include <PID_v1.h>

//#include "I2Cdev.h" //library allows communication with I2C / TWI devices

#include "math.h" //library includes mathematical functions

#include <Servo.h>    // memasukkan library servo

#include <PID_v1.h>

Servo myservo;

int sensorpin = A0;

int y = 1;

float rata_coba, sampling_coba;

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);



float sp_rpm = 1500;

double Input, Output, pul;

double Setpoint;
```

```
double Kp=2, Ki=1, Kd=0;

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);



const int MPU=0x68; //I2C address of the MPU-6050

int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; //16-bit integers

int AcXcal,AcYcal,AcZcal,GyXcal,GyYcal,GyZcal,tcal; //calibration variables

double t,tx,tf,pitch,roll;



int encoder_pin = 2; // pulse output from the module

unsigned int rpm; // rpm reading

volatile byte pulses; // number of pulses

unsigned long timeold;

//unsigned int pulsesperturn = 20;

void counter()

{

  //Update count

pulses++;

}
```

```
void setup()

{

Wire.begin(); //initiate wire library and I2C

Wire.beginTransmission(MPU); //begin transmission to I2C slave device

Wire.write(0x6B); // PWR_MGMT_1 register

Wire.write(0); // set to zero (wakes up the MPU-6050)

Wire.endTransmission(true); //ends transmission to I2C slave device

Serial.begin(9600); //serial communication at 9600 bauds

//      lcd.begin();

//      lcd.backlight();

//

//      //Serial.print(" PWM = ");

//    //  Serial.print(pul) ;

//      lcd.setCursor(0,0);

//      lcd.print(pul);

//     // Serial.print(" Pitch = ");

//     // Serial.println(pitch);

//      lcd.setCursor(0,1);
```

```
//      lcd.print(pitch);

//     // Serial.print(" RPM = ");

//     // Serial.print(rpm);

//      lcd.setCursor(7,1);

//      lcd.print(rpm);

////   pinMode(encoder_pin, INPUT);

////   //Interrupt 0 is digital pin 2

////   //Triggers on Falling Edge (change from HIGH to LOW)

////   attachInterrupt(0, counter, FALLING);

////   // Initialize

////   pulses = 0;

//   rpm = 0;

////   timeold = 0;

myservo.attach(9);

   //Setpoint = (225/5)*sp_rpm ; // ubah RPM ke PWM

//myservo.write(Setpoint);

myPID.SetMode(AUTOMATIC);

}

void loop()
```

```
{
Wire.beginTransmission(MPU); //begin transmission to I2C slave device
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false); //restarts transmission to I2C slave device
Wire.requestFrom(MPU,14,true); //request 14 registers in total
   //Acceleration data correction
   AcXcal = -950;    AcYcal = -300;    AcZcal = 0;
   //Gyro correction
   GyXcal = 480;    GyYcal = 170;    GyZcal = 210;
   //read accelerometer data
   AcX=Wire.read()<<8|Wire.read(); /* 0x3B (ACCEL_XOUT_H) 0x3C
(ACCEL_XOUT_L) */ AcY=Wire.read()<<8|Wire.read(); /* 0x3D
(ACCEL_YOUT_H) 0x3E (ACCEL_YOUT_L) */
AcZ=Wire.read()<<8|Wire.read(); /* 0x3F (ACCEL_ZOUT_H) 0x40
(ACCEL_ZOUT_L) */
   //read gyroscope data
   GyX=Wire.read()<<8|Wire.read(); /* 0x43 (GYRO_XOUT_H) 0x44
(GYRO_XOUT_L)*/ GyY=Wire.read()<<8|Wire.read(); /* 0x45
(GYRO_YOUT_H) 0x46 (GYRO_YOUT_L)*/
```

```
GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) 0x48
(GYRO_ZOUT_L)
   //get pitch/roll
getAngle(AcX,AcY,AcZ);


   //printing values to serial port
   //Serial.print("Angle: ");
   //Serial.print("Pitch = "); Serial.print(pitch);
   //Serial.println(" Roll = "); Serial.println(roll);


if (pitch < -10) {
    Setpoint = 229 ;
   }
else if (pitch > 10) {
    Setpoint = 229 ;
   }
else { Setpoint = 229 ;}
if (pitch < -30) {
    Setpoint = 20;
```

```
    }
else if (pitch > 30) {
     Setpoint = 20;}
else { Setpoint = 229 ;}

   // pwm = 1.046* Setpoint - 163.95 ;
   //int pwm = map(pitch, 0, 90, 70, 150) ;
   //myservo.write(pwm);

for ( int y =0; y < 200; y++){
int baca_coba = analogRead (A0);
   sampling_coba = sampling_coba + baca_coba ;
delay(1);
   }
   rata_coba = sampling_coba / 200.0;
   sampling_coba = 0;
rpm = 0.649 *rata_coba -14.27;
```

```
    //delay(1000);


if (millis() - timeold >= 1000) {
      //Don't process interrupts during calculations
      //detachInterrupt(0);
     // rpm = (60 * 1000 / pulsesperturn )/ (millis() - timeold)* pulses;
timeold = millis();
pulses = 0;
lcd.begin();
lcd.backlight();


Serial.print(" | PWM = ");
Serial.print(pul) ;
     // lcd.setCursor(0,0);
     // lcd.print(pul);
Serial.print(" | Pitch = ");
Serial.println(pitch);
```

```
    // lcd.setCursor(0,1);
    // lcd.print(pitch);
Serial.print("RPM = ");
Serial.print(rpm);
   //  lcd.setCursor(7,1);
   //  lcd.print(rpm);
delay(10);
     }

     Input = rpm ;
myPID.Compute();
pul = 0.649 * Output + -14.27 ;
myservo.write(pul);
}
//function to convert accelerometer values into pitch and roll
void getAngle(int Ax,int Ay,int Az)
{
double x = Ax;
```

```
double y = Ay;

double z = Az;



pitch = atan(x/sqrt((y*y) + (z*z))); //pitch calculation

roll = atan(y/sqrt((x*x) + (z*z))); //roll calculation

   //converting radians into degrees

pitch = pitch * (180.0/3.14);

roll = roll * (180.0/3.14) ;

}
```