```
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

#define MIN_ABS_SPEED 20

MPU6050 mpu;

bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];

Quaternion q;
VectorFloat gravity;
float ypr[3];


double originalSetpoint = 175.8;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
int moveState = 0;
double Kp = 50;
double Kd = 1.4;
double Ki = 60;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);

double motorSpeedFactorLeft = 0.6;
double motorSpeedFactorRight = 0.5;

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);



long time1Hz = 0;
long time5Hz = 0;

volatile bool mpuInterrupt = false;
void dmpDataReady()
{
  mpuInterrupt = true;
}



void setup()
{

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
```

```
  Wire.begin();
  TWBR = 24;
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif


  Serial.begin(115200);
  while (!Serial);


  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();


  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));


  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();


  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);

  if (devStatus == 0)
  {
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();



    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;



    packetSize = mpu.dmpGetFIFOPacketSize();



    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
  }
  else
  {

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



void loop()
```

```
{

 if (!dmpReady) return;


 while (!mpuInterrupt && fifoCount < packetSize)
 {


  pid.Compute();
  motorController.move(output, MIN_ABS_SPEED);

 }


 mpuInterrupt = false;
 mpuIntStatus = mpu.getIntStatus();


 fifoCount = mpu.getFIFOCount();


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

  mpu.resetFIFO();
  Serial.println(F("FIFO overflow!"));


 }
 else if (mpuIntStatus & 0x02)
 {

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


   mpu.getFIFOBytes(fifoBuffer, packetSize);

   fifoCount -= packetSize;

   mpu.dmpGetQuaternion(&q, fifoBuffer);
   mpu.dmpGetGravity(&gravity, &q);
   mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
#if LOG_INPUT
   Serial.print("ypr\t");
   Serial.print(ypr[0] * 180 / M_PI);
   Serial.print("\t");
   Serial.print(ypr[1] * 180 / M_PI);
   Serial.print("\t");
   Serial.println(ypr[2] * 180 / M_PI);
#endif
   input = ypr[1] * 180 / M_PI + 180;
 }
}
```