//newwwwww

// include the library code:

#include <LiquidCrystal.h>

#include <Keypad.h>

#include <stdlib.h>

#include <map>

#include <MPU6050\_light.h>

#include <Math.h>

MPU6050 mpu(Wire);

std::map <int, String> directions;

#define I2C\_SLAVE\_ADDR 0X04

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(19, 23, 18, 17, 16, 15);

const byte ROWS = 4; //four rows

const byte COLS = 3; //four columns

//define the cymbols on the buttons of the keypads

char keys[ROWS][COLS] = {

{'1','2','3'},

{'4','5','6'},

{'7','8','9'},

{'\*','0','#'}

};

byte rowPins[ROWS] = {33, 5, 26, 25}; //connect to the row pinouts of the keypad

byte colPins[COLS] = {4, 0, 2}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {

// set up the LCD's number of columns and rows:

Serial.begin(9600);

lcd.begin(16, 2);

// Print a message to the LCD.

directions[1]= 'F';

directions[2]= 'B';

directions[3]= 'L';

directions[4]= 'R';

/\*byte status = mpu.begin();

Serial.print(F("MPU6050 STATUS: "));

Serial.println(status);

while (status!= 0){}

Serial.println(F("Calculating offsets, do not move MPU"));

delay(1000);

mpu.calcOffsets();

Serial.println("done!\n");\*/

}

void loop()

{

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.clear();

lcd.setCursor(0, 0);

// print the number of seconds since reset:

lcd.print("Enter no. of ");

lcd.setCursor(0, 1);

lcd.print("commands:");

String numOfCommands = "";

char key;

do {

key = keypad.waitForKey();

if ( key != NO\_KEY){

Serial.println(key);

lcd.print(key);

if (numOfCommands.toInt()>20 && key == '\*'){

numOfCommands = "";

}

else{

numOfCommands += key ;

}

}

}

while ((numOfCommands.toInt()>20) ||(numOfCommands.toInt()==0) || (key!='\*') );

int n = numOfCommands.toInt();

lcd.clear();

lcd.print(n);

delay(1000);

// creating arrays

String Commands[n];

int quanitities[n] ;

for ( int i = 0 ; i<n; i++) {

lcd.setCursor(0,0);

lcd.print("Enter command ");

lcd.print(i+1);

String commands = "";

while (commands == "")

{

char key = keypad.waitForKey();

if ((key!=NO\_KEY) && (key=='1' || key=='2'|| key=='3'||key=='4'))

{lcd.clear();

commands += key;

lcd.print(directions[commands.toInt()]);

delay(1000);

}

else {

lcd.clear();

lcd.print("invalid command");

}

}

Commands[i] = directions[commands.toInt()] ;

lcd.clear();

lcd.print("Enter quantity ");

lcd.print(i+1);

lcd.setCursor(0,1);

String quantity = "";

char key;

do{

key=keypad.waitForKey();

if (((key!=NO\_KEY) && (Commands[i]== "L" || Commands[i]=="R" )) && (key!='5' && key!='6')){

lcd.clear();

lcd.print("invalid quantity");

lcd.setCursor(0,1);

lcd.print("5->90, 6->180");

}

else if (key!= NO\_KEY){

lcd.clear();

quantity += key;

if (quantity.toInt()==5 && (Commands[i]=="L"|| Commands[i]=="R")){

lcd.print(90);

}

else if (quantity.toInt()==6 && (Commands[i]=="L"|| Commands[i]=="R")){

lcd.print(180);

}

else{

lcd.print(quantity.toInt());

}

}

}while(key!='\*');

quanitities[i]= quantity.toInt();

lcd.clear();

}

lcd.print("[");

for ( int i=0 ; i<n; i++) {

lcd.print(Commands[i]);

lcd.print(quanitities[i]);

if (i < n -1) {

lcd.print(",");}}

lcd.print("]");

lcd.setCursor(0,1);

lcd.print("\* to confirm");

while(keypad.waitForKey()!='\*') {

lcd.clear();

lcd.print("\* to confirm");

}

for (int i=0 ; i<n; i++) {

if (Commands[i] == "F" ){

forward(quanitities[i]);

}

else if (Commands[i] == "B" ){

backward(quanitities[i]);

}

else if (Commands[i] == "R" ){

right(quanitities[i]);

}

else if (Commands[i] == "L" ){

left(quanitities[i]);

}

}

}

//setting up keypad

float getDistance(void){

int16\_t encoder1 =0 ;

int16\_t encoder2 =0 ;

Wire.requestFrom(I2C\_SLAVE\_ADDR, 4);

while (Wire.available()){

uint8\_t enC1\_16\_9 = Wire.read();

uint8\_t enC1\_8\_1 = Wire.read();

uint8\_t enC2\_16\_9 = Wire.read();

uint8\_t enC2\_8\_1 = Wire.read();

encoder1 = (enC1\_16\_9 << 8)| enC1\_8\_1;

encoder2 = (enC2\_16\_9 << 8)| enC2\_8\_1;

}

float distance = (encoder1 \*(0.06\*M\_PI)/24)\*100 ; //0.06 IS DIAMETER OF WHEEL 24 IS n and 100 is for conversion

Serial.print(distance);

delay (100);

return distance;

}

void forward(int distance){

lcd.setCursor(0, 1);

lcd.print("going forward");

float currentDistance = getDistance();

int setpoint = mpu.getAngleZ();

while ((getDistance()- (currentDistance))<(distance\*10)){

int leftMotor\_speed = 255;

int rightMotor\_speed = 253;

int servoAngle = 85;

straightLineFollowing(leftMotor\_speed, rightMotor\_speed, servoAngle, setpoint);

}

int leftMotor\_speed= 0;

int rightMotor\_speed= 0;

int servoAngle = 85;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

delay(1000);

lcd.clear();

}

void backward(int distance){

lcd.setCursor(0, 1);

lcd.print("going backward");

float currentDistance = getDistance();

int setpoint = mpu.getAngleZ();

while ((getDistance()- currentDistance)>(distance\*10\*-1)){

int leftMotor\_speed = -255;

int rightMotor\_speed = -253;

int servoAngle = 85;

straightLineFollowing(leftMotor\_speed, rightMotor\_speed, servoAngle, setpoint);

}

int leftMotor\_speed= 0;

int rightMotor\_speed= 0;

int servoAngle = 85;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

delay(1000);

lcd.clear();

}

void right(int angle){

lcd.setCursor(0, 1);

lcd.print("going right");

if (angle == 5){

angle = 90;

}

else {angle = 180;}

float currentAngle = mpu.getAngleZ();

while ((currentAngle - mpu.getAngleZ())<(angle)){

int leftMotor\_speed = 255;

int rightMotor\_speed = 253;

int servoAngle = 130;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

}

int leftMotor\_speed= 0;

int rightMotor\_speed= 0;

int servoAngle = 85;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

delay(1000);

lcd.clear();

}

void left(int angle){

lcd.setCursor(0, 1);

lcd.print("going left");

if (angle == 5){

angle = 90;

}

else {angle = 180;}

float currentAngle = mpu.getAngleZ();

while ((currentAngle - mpu.getAngleZ())>(angle)){

int leftMotor\_speed = 255;

int rightMotor\_speed = 253;

int servoAngle = 60;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

}

int leftMotor\_speed= 0;

int rightMotor\_speed= 0;

int servoAngle = 85;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

delay(1000);

lcd.clear();

}

void straightLineFollowing ( int leftMotor\_speed, int rightMotor\_speed, int servoAngle, int setpoint ){

double error;

double prev\_error;

double Kp = 0.5;

double Ki = 0;

double Kd = 0;

double K=0.6;

double cumulative\_error;

int baseSpeed = 100;

mpu.update();

error = mpu.getAngleZ() - setpoint;

double PID = (Kp\*error)+(Ki\*cumulative\_error)+(Kd\*(error-prev\_error));

servoAngle = 85 +PID;

leftMotor\_speed = baseSpeed + K\*PID;

rightMotor\_speed = baseSpeed - K\*PID;

cumulative\_error=cumulative\_error+error;

prev\_error = error;

cumulative\_error = cumulative\_error+error;

prev\_error = error;

Transmit\_to\_arduino(leftMotor\_speed, rightMotor\_speed, servoAngle);

delay(1000);

}

void Transmit\_to\_arduino(int leftMotor\_speed, int rightMotor\_speed, int servoAngle)

{

Wire.beginTransmission(I2C\_SLAVE\_ADDR); // transmit to device

Wire.write((byte)((leftMotor\_speed & 0x0000FF00) >> 8)); // first byte of x, containing bits 16 to 9

Wire.write((byte)(leftMotor\_speed & 0x000000FF)); // second byte of x, containing the 8 LSB - bits 8 to 1

Wire.write((byte)((rightMotor\_speed & 0x0000FF00) >> 8)); // first byte of y, containing bits 16 to 9

Wire.write((byte)(rightMotor\_speed & 0x000000FF)); // second byte of y, containing the 8 LSB - bits 8 to 1

Wire.write((byte)((servoAngle & 0x0000FF00) >> 8)); // first byte of x, containing bits 16 to 9

Wire.write((byte)(servoAngle & 0x000000FF));

Wire.endTransmission(); // stop transmitting

}