#include <Adafruit\_PWMServoDriver.h>

#include <Wire.h>

Adafruit\_PWMServoDriver board1 = Adafruit\_PWMServoDriver(0x40); // called this way, it uses the default address 0x40

#define SERVOMIN 100 // Adjust according to your servo

#define SERVOMAX 600 // Adjust according to your servo

#define Muro\_trig 5

#define Muro\_echo 6

#define Ostacolo\_trig 3

#define Ostacolo\_echo 4

#define IN\_A0 A1

long durata,cm,durata2,cm2,rifrazione,err;

int Deviazione,c=0,last\_error = 0;

int timer = millis();

bool ag = false;

void stop()

{

AvviaServoAvanti(10,10);

AvviaServoIndietro(10,10);

}

void setup() {

Serial.begin(9600);

cm = 1000;

board1.begin();

board1.setPWMFreq(60);

AvviaServoAvanti(10,10);

AvviaServoIndietro(10,10);

pinMode (IN\_A0,INPUT);

pinMode(Muro\_trig, OUTPUT);

pinMode(Muro\_echo, INPUT);

pinMode(Ostacolo\_trig,OUTPUT);

pinMode(Ostacolo\_echo,INPUT);

impostaservo(145,4); //TILT (160 per il seguilinea,115 per le palline)

impostaservo(95,15);

gestiscibraccio(200); //braccio sopra

apribracci();

}

void prendipallina()

{

AvviaServoAvanti(-30,50);

AvviaServoIndietro(-30,50);

delay(1500);

calcolapotenzamotori(10,25);

gestiscibraccio(85);

apribracci();

delay(4000);

chiudibracci();

delay(500);

gestiscibraccio(200);

delay(1000);

AvviaServoAvanti(-30,50);

AvviaServoIndietro(-30,50);

delay(1000);

}

void depositapallina()

{

calcolapotenzamotori(0,25);

gestiscibraccio(130);

delay(2000);

apribracci();

gestiscibraccio(200);

delay(1000);

AvviaServoAvanti(-30,50);

AvviaServoIndietro(-30,50);

delay(2000);

}

void apribracci()

{

impostaservo(20,13); //20 aperto 70 chiuso

impostaservo(180,14); //180 aperto 135 chiuso

}

void chiudibracci()

{

impostaservo(60,13); //20 aperto 70 chiuso

impostaservo(60,14); //180 aperto 135 chiuso

}

void impostaservo(int gradi,int pin){board1.setPWM(pin, 0, angleToPulse(gradi) );}

void gestiscibraccio(int gradi)

{

impostaservo(180-gradi,9);

impostaservo(gradi,8);

}

int sw = -1;

int pallinegradi = 110;

String Message;

//PID PER IL MURO//

float Kp = 6.0; // Costante proporzionale

float Ki = 0.0; // Costante integrale

float Kd = 0.0; // Costante derivativa

float setpoint = 4.5; // Distanza desiderata dal muro

float prevError = 0.0;

float integral = 0.0;

float calculatePID(float input) {

if (input < 5)

input\*=-1;

float error = setpoint - input;

integral += error;

float derivative = error - prevError;

float output = Kp \* error + Ki \* integral + Kd \* derivative;

prevError = error;

output = constrain(output,-25,25);

if (CheckDistanza(Ostacolo\_trig,Ostacolo\_echo) < 7)

{

calcolapotenzamotori(-100,40);

delay(1750);

}

return output;

}

//FINE PID PER IL MURO//

int limite = 27;

int agvisto = 0;

int timer1 = millis();

void loop() {

/\*

LA PALLINA DEVE ESSERE PRESA QUANDO E' A 12 CM DI DISTANZA

\*/

//Serial.println(CheckDistanza(Muro\_trig,Muro\_echo));

//Serial.println(CheckDistanza(Ostacolo\_trig,Ostacolo\_echo));

if (Serial.available() > 0) {

cm = CheckDistanza(Ostacolo\_trig,Ostacolo\_echo);

if(agvisto == 0)

if((cm==5)&&(cm>0)){

obsostacolo();

Serial.println(1000);

while (Deviazione != 1000)

{

Message = Serial.readStringUntil('\n'); // Legge i dati disponibili fino a quando non viene ricevuto un carattere di nuova riga

Deviazione = Message.toInt(); // Converte il messaggio in un numero intero

cm2 = CheckDistanza(Muro\_trig,Muro\_echo);

err = calculatePID(cm2);

err = constrain(err,-20,20);

calcolapotenzamotori(err, limite); // Esegue il calcolo con il valore ricevuto

delay(100);

}

calcolapotenzamotori(100,40);

delay(500);

calcolapotenzamotori(0,40);

delay(1000);

}

// Leggi direttamente un valore intero dalla porta seriale

Message = Serial.readStringUntil('\n');

Deviazione = Message.toInt();

//entra nella zona delle palline

//se perde il cassonetto

if ((Deviazione!=1000))

{

calcolapotenzamotori(Deviazione,limite);

}

else

{

stop();

impostaservo(145,4);

limite = 27;

}

if((Deviazione>3000)&&(Deviazione < 4000))

{ limite = 20;

impostaservo(Deviazione-3000,4);

}

if((Deviazione>2000)&&(Deviazione < 3000))

esegui\_operazione(Deviazione);

}

}

void esegui\_operazione(int n)

{

int codice = n-2000;

delay(100);

switch(codice)

{

case 1:

Serial.println(CheckDistanza(Ostacolo\_trig,Ostacolo\_echo));

break;

case 2:

Serial.println(CheckDistanza(Muro\_trig,Muro\_echo));

break;

case 3:

prendipallina();

Serial.println(0);

break;

case 4:

depositapallina();

Serial.println(0);

break;

}

}

void AvviaServoAvanti(int speed,int speed2) {

// Set PWM to control the speed and direction of the continuous rotation servo

board1.setPWM(0, 0, map(speed, -100, 100, SERVOMIN, SERVOMAX));

board1.setPWM(1, 0, map(speed2, -100, 100, SERVOMIN, SERVOMAX));

}

void AvviaServoIndietro(int speed,int speed2)

{

board1.setPWM(2, 0, map(speed, -100, 100, SERVOMIN, SERVOMAX));

board1.setPWM(3, 0, map(speed2, -100, 100, SERVOMIN, SERVOMAX));

}

void calcolapotenzamotori(int deviazione,int lim)

{

int dev2;

int potenzaDX = lim+deviazione;

int potenzaSX = lim-deviazione;

potenzaDX = constrain(potenzaDX, -lim, lim);

potenzaSX = constrain(potenzaSX,-lim,lim);

AvviaServoAvanti(potenzaDX+10,-1\*(potenzaSX)+10);

dev2 = constrain(deviazione,-100,100); //da modificare a -lim a lim

potenzaDX = lim+dev2;

potenzaSX = lim-dev2;

potenzaDX = constrain(potenzaDX, 0, lim);

potenzaSX = constrain(potenzaSX,0,lim);

AvviaServoIndietro(potenzaDX+10,-1\*(potenzaSX)+10);

}

int CheckDistanza(int trig,int echo)

{

digitalWrite(trig, LOW);

delayMicroseconds(2);

digitalWrite(trig, HIGH);

delayMicroseconds(10);

digitalWrite(trig, LOW);

int durata = pulseIn(echo, HIGH);

int cm = durata / 58;

return cm;

}

void obsostacolo()

{

AvviaServoAvanti(-30,50);

AvviaServoIndietro(-30,50);

delay(1250);

calcolapotenzamotori(100,40);

delay(1200);

AvviaServoAvanti(50,-30);

AvviaServoIndietro(50,-30);

delay(1600);

}

int angleToPulse(int ang){

int pulse = map(ang,0, 180, SERVOMIN,SERVOMAX);// map angle of 0 to 180 to Servo min and Servo max

return pulse;

}