// Portas driver motor

#define PININ1 2

#define PININ2 4

#define PININ3 5

#define PININ4 7

#define PINENA 3

#define PINENB 6

// Portas led rgb

#define PINLEDR 11

#define PINLEDB 10

#define PINLEDG 9

// Portas sensor QTR

#define SENSOR1 A0

#define SENSOR2 A1

#define SENSOR3 A2

#define SENSOR4 A3

#define SENSOR5 A4

#define SENSOR6 A5

// Valores para o cálculo de PD

int err, prv\_err=0;

int speed=255, left\_speed, right\_speed;

float Kp=40, Kd=2, PID=0;

// Valores de ajustes para o seguidor de linha MIF

#define TRESHOLD 512 // Valor de referencia para cor da linha branca

#define RUNTIME 25000 // Valor para executar o percurso

void setup() {

Serial.begin(9600);

rgbControl(255, 0, 0, 1000);

rgbControl(0, 255, 0, 1000);

rgbControl(0, 0, 255, 1000);

}

void loop() {

followLinePD();

//readSensors();

}

// -------- Apenas controle Bluetooth ---------

void advancedOption(char option) {

// Função para controle de opções avançadas

switch (option) {

case 'a':

readSensors();

break;

case 'b':

followLinePD(); // Seguidor de linha com Máquina de Estados Finitos

break;

}

}

void motorControl(int speedLeft, int speedRight) {

// Função para controle do driver de motor

// Definições das portas digitais

pinMode(PININ1, OUTPUT);

pinMode(PININ2, OUTPUT);

pinMode(PININ3, OUTPUT);

pinMode(PININ4, OUTPUT);

pinMode(PINENA, OUTPUT);

pinMode(PINENB, OUTPUT);

// Ajustes motor da esquerda

if (speedLeft < 0) {

speedLeft = -speedLeft;

digitalWrite (PININ3, HIGH);

digitalWrite (PININ4, LOW);

} else {

digitalWrite (PININ3, LOW);

digitalWrite (PININ4, HIGH);

}

// Ajustes motor da direita

if (speedRight < 0) {

speedRight = -speedRight;

digitalWrite (PININ1, LOW);

digitalWrite (PININ2, HIGH);

} else {

digitalWrite (PININ1, HIGH);

digitalWrite (PININ2, LOW);

}

analogWrite (PINENA, speedLeft);

analogWrite (PINENB, speedRight);

}

void motorOption(char option, int speedLeft, int speedRight) {

// Função para controle de motor com pre definições

switch (option) {

case '6': // Esquerda

motorControl(-speedLeft, speedRight);

break;

case '4': // Direita

motorControl(speedLeft, -speedRight);

break;

case '2': // Trás

motorControl(-speedLeft, -speedRight);

break;

case '8': // Frente

motorControl(speedLeft, speedRight);

break;

case '0': // Parar

motorControl(0, 0);

break;

}

}

bool motorStop(long runtime, long currentTime) {

// Função de parada do robô

if (millis() >= (runtime + currentTime)) {

motorOption('0', 0, 0);

int cont = 0;

while (cont < 5) {

rgbControl(255, 0, 0, 500);

rgbControl(0, 0, 0, 500);

cont++;

}

return false;

}

return true;

}

void rgbControl(int red, int green, int blue, long rumtime) {

// Função para controle do led rgb

pinMode(PINLEDR, OUTPUT);

pinMode(PINLEDG, OUTPUT);

pinMode(PINLEDB, OUTPUT);

digitalWrite(PINLEDR, HIGH);

digitalWrite(PINLEDG, HIGH);

digitalWrite(PINLEDB, HIGH);

analogWrite(PINLEDR, 255 - red);

analogWrite(PINLEDG, 255 - green);

analogWrite(PINLEDB, 255 - blue);

delay(rumtime);

}

void readSensors(void) {

// Função para leitura dos sensores

Serial.print(analogRead(A0));

Serial.print(' ');

Serial.print(analogRead(A1));

Serial.print(' ');

Serial.print(analogRead(A2));

Serial.print(' ');

Serial.print(analogRead(A3));

Serial.print(' ');

Serial.print(analogRead(A4));

Serial.print(' ');

Serial.println(analogRead(A5));

Serial.print(' ');

}

void calc\_pid(void){

// Função para o cálculo de PD

int deriv;

deriv = err - prv\_err;

PID = Kp\*err + Kd\*deriv;

prv\_err = err;

}

void calc\_speed(void){

// Função para o cálculo das velocidades

if(err && err!=7 && err!=-7){

left\_speed = speed + PID;

right\_speed = speed - PID;

}else if(err==7){

left\_speed = speed - PID;

right\_speed = speed - PID;

}else{

left\_speed = speed + PID;

right\_speed = speed + PID;

}

// Definição dos limites de velocidades

if(left\_speed>255) left\_speed = 255;

if(left\_speed<0) left\_speed = 0;

if(right\_speed>255) right\_speed = 255;

if(right\_speed<0) right\_speed = 0;

}

void followLinePD(void) {

// Função para controle do seguidor de linha com cálculo de PID

bool flag = true;

long currentTime = millis();

while (flag) {

flag = motorStop(RUNTIME, currentTime);

// leitura do sensor (1 1 1 1 1 1)

if (analogRead(A0) <= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) <= TRESHOLD) {

err=0;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 1 1 1 1 0)

} else if ( analogRead(A0) >= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=0;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 1 1 0 0)

} else if ( analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=0;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 1 1 1 0 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=1;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 1 1 1 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) >= TRESHOLD ) {

err=-1;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 1 0 0 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=2;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 0 1 0 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD ) {

err=-2;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 1 1 0 0 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=3;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 0 1 1 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=-3;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (1 1 1 0 0 0)

} else if (analogRead(A0) <= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) <= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=4;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 0 1 1 1)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) <= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) <= TRESHOLD) {

err=-4;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 1 0 0 0 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=5;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 0 0 1 0)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=-5;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (1 1 0 0 0 0)

} else if (analogRead(A0) <= TRESHOLD && analogRead(A1) <= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=6;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (0 0 0 0 1 1)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) <= TRESHOLD && analogRead(A5) <= TRESHOLD) {

err=-6;

calc\_pid();

calc\_speed();

motorOption('8', left\_speed, right\_speed);

// leitura do sensor (1 0 0 0 0 0)

} else if (analogRead(A0) <= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) >= TRESHOLD) {

err=7;

calc\_pid();

calc\_speed();

motorOption('6', left\_speed, right\_speed);

// leitura do sensor (0 0 0 0 0 1)

} else if (analogRead(A0) >= TRESHOLD && analogRead(A1) >= TRESHOLD && analogRead(A2) >= TRESHOLD && analogRead(A3) >= TRESHOLD && analogRead(A4) >= TRESHOLD && analogRead(A5) <= TRESHOLD) {

err=-7;

calc\_pid();

calc\_speed();

motorOption('4', left\_speed, right\_speed);

}

}

motorOption('0', 0, 0);

}