Documentatie Proiect

Componente:

* KS0556 Keyestudio 8833 Motor Driver Expansion Board
* Keyestudio V4.0 Board
* 2xUltrasonic Sensor
* 4xMotoare
* 4xRoti
* Fotorezistenta
* 2x LED
* 3x Rezistente 220
* Joystick

Functionalitati:

-miscare fata-spate, stanga-dreapta (din motiv al calitatii proaste al motoarelor miscarea e mai dificila si a facut lucrarea cu Infra-rosu imposibila)

-oprire in cazul in care e un obstacol in fata/spatele masinii

-aprindere de LED la nivel scazut de luminozitate

Folosinta in viata de zi cu zi:

O masina teleghidata (RC - remote-controlled) poate fi utilizata intr-o varietate de aplicatii practice si de divertisment. In context utilitar, o astfel de masina poate fi folosita pentru inspectii de siguranta, transport de obiecte mici sau monitorizare video in zone cu acces restrictionat. Controlata printr-un joystick, aceasta permite deplasarea precisa in orice directie, iar senzorii de proximitate opresc automat masina in fata obstacolelor pentru a preveni coliziunile. De asemenea, sistemul de iluminare automata, activat de o fotorezistenta, asigura navigarea in medii intunecate, extinzand aria sa de utilizare.

Diagrama Bloc:

Joystick

Obiect

Photoresistor

Ultrasound sensor

Motor

LED

Cod Folosit:  
// Pin Definitions

const int X\_AXIS = A3;

const int Y\_AXIS = A5;

const int LIGHT\_SENSOR = A0;

const int LED = A4;

const int RIGHT\_MOTOR\_CTRL = 2;

const int RIGHT\_MOTOR\_PWR = 5;

const int LEFT\_MOTOR\_CTRL = 4;

const int LEFT\_MOTOR\_PWR = 6;

const int FRONT\_TRIGGER\_PIN = 10;

const int FRONT\_ECHO\_PIN = 9;

const int REAR\_TRIGGER\_PIN = 12;

const int REAR\_ECHO\_PIN = 13;

// Thresholds and constants

const int DEADZONE = 50;

const int LIGHT\_THRESHOLD = 500;

const int MAX\_SPEED = 255;

const long OBSTACLE\_THRESHOLD\_CM = 15;

// Variables

int joystickX = 0;

int joystickY = 0;

// MotorMovement class

class MotorMovement {

public:

void moveForward(int speed) {

digitalWrite(RIGHT\_MOTOR\_CTRL, HIGH);

digitalWrite(LEFT\_MOTOR\_CTRL, HIGH);

analogWrite(RIGHT\_MOTOR\_PWR, speed);

analogWrite(LEFT\_MOTOR\_PWR, speed);

}

void moveBackward(int speed) {

digitalWrite(RIGHT\_MOTOR\_CTRL, LOW);

digitalWrite(LEFT\_MOTOR\_CTRL, LOW);

analogWrite(RIGHT\_MOTOR\_PWR, speed);

analogWrite(LEFT\_MOTOR\_PWR, speed);

}

void turnRight(int speed) {

digitalWrite(RIGHT\_MOTOR\_CTRL, LOW);

digitalWrite(LEFT\_MOTOR\_CTRL, HIGH);

analogWrite(RIGHT\_MOTOR\_PWR, speed);

analogWrite(LEFT\_MOTOR\_PWR, speed);

}

void turnLeft(int speed) {

digitalWrite(RIGHT\_MOTOR\_CTRL, HIGH);

digitalWrite(LEFT\_MOTOR\_CTRL, LOW);

analogWrite(RIGHT\_MOTOR\_PWR, speed);

analogWrite(LEFT\_MOTOR\_PWR, speed);

}

void stopMotors() {

analogWrite(RIGHT\_MOTOR\_PWR, 0);

analogWrite(LEFT\_MOTOR\_PWR, 0);

digitalWrite(RIGHT\_MOTOR\_CTRL, LOW);

digitalWrite(LEFT\_MOTOR\_CTRL, LOW);

}

};

// UltrasonicSensor class

class UltrasonicSensor {

public:

long getDistance(int triggerPin, int echoPin) {

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

long duration = pulseIn(echoPin, HIGH);

return duration \* 0.034 / 2;

}

};

// Create objects

MotorMovement motor;

UltrasonicSensor ultrasonic;

void setup() {

// Initialize pins

pinMode(LED, OUTPUT);

pinMode(RIGHT\_MOTOR\_CTRL, OUTPUT);

pinMode(RIGHT\_MOTOR\_PWR, OUTPUT);

pinMode(LEFT\_MOTOR\_CTRL, OUTPUT);

pinMode(LEFT\_MOTOR\_PWR, OUTPUT);

pinMode(FRONT\_TRIGGER\_PIN, OUTPUT);

pinMode(FRONT\_ECHO\_PIN, INPUT);

pinMode(REAR\_TRIGGER\_PIN, OUTPUT);

pinMode(REAR\_ECHO\_PIN, INPUT);

Serial.begin(9600); // For debugging

}

void loop() {

joystickX = analogRead(X\_AXIS) - 512;

joystickY = analogRead(Y\_AXIS) - 512;

int lightLevel = analogRead(LIGHT\_SENSOR);

digitalWrite(LED, lightLevel < LIGHT\_THRESHOLD ? HIGH : LOW);

if (abs(joystickX) < DEADZONE && abs(joystickY) < DEADZONE) {

motor.stopMotors(); // Full stop

return;

}

long frontDistance = ultrasonic.getDistance(FRONT\_TRIGGER\_PIN, FRONT\_ECHO\_PIN);

long rearDistance = ultrasonic.getDistance(REAR\_TRIGGER\_PIN, REAR\_ECHO\_PIN);

// Miscare fata-spate

if (joystickY > DEADZONE && frontDistance > OBSTACLE\_THRESHOLD\_CM) {

motor.moveForward(map(joystickY, DEADZONE, 512, 0, MAX\_SPEED));

} else if (joystickY < -DEADZONE && rearDistance > OBSTACLE\_THRESHOLD\_CM) {

motor.moveBackward(map(abs(joystickY), DEADZONE, 512, 0, MAX\_SPEED));

}

// Miscare rotativa

else if (joystickX > DEADZONE) {

motor.turnRight(map(joystickX, DEADZONE, 512, 0, MAX\_SPEED));

} else if (joystickX < -DEADZONE) {

motor.turnLeft(map(abs(joystickX), DEADZONE, 512, 0, MAX\_SPEED));

} else {

motor.stopMotors();

}

}