**Documentație proiect**

**Proiectare cu microprocesoare**

**Proiect realizat de:** Péntek Tamás, Podaru Bogdan-Mihai

**Profesor îndrumător:** Radu Dănescu

**Grupa:** 30238

**Data:** 13.01.2020

**Cuprins :**

1. **Rezumat**
2. **Detalii de implementare**
3. **Concluzii**
4. **Dezvoltări ulterioare**
5. **Referințe**
6. **Rezumat:**

Proiectul are în vedere consolidarea cunoștiințelor dobândite pe parcursul sesiunilor de laborator. Propunem un program în care cu ajutorul unei aplicații Android controlăm kitul robotic care preia date din mediu printr-un modul Ultrasonic și le procesează prin microcontrollerul Arduino Uno, după care le trimite printr-un modul de Bluetooth HC-05 către calculator, unde sunt afișate cu ajutorul mediului Processing.

Montajul realizat conține :

1. Placă microcontroller compatibilă Arduino Uno

2. Driver motoare L298N Dual H-Bridge

3. 2x Motor DC

4. 1 Motor Servo

5. Carcasă baterii 4xAA (R6)

6. 2 roţi conectate la motoare, 1 roată suport

7. Suport plexiglas

8. Două plăci de prototipizare (Breadboard)

9. 1 senzor sonar (neconectat)

10. Buzzer

11. Modul Bluetooth HC-05

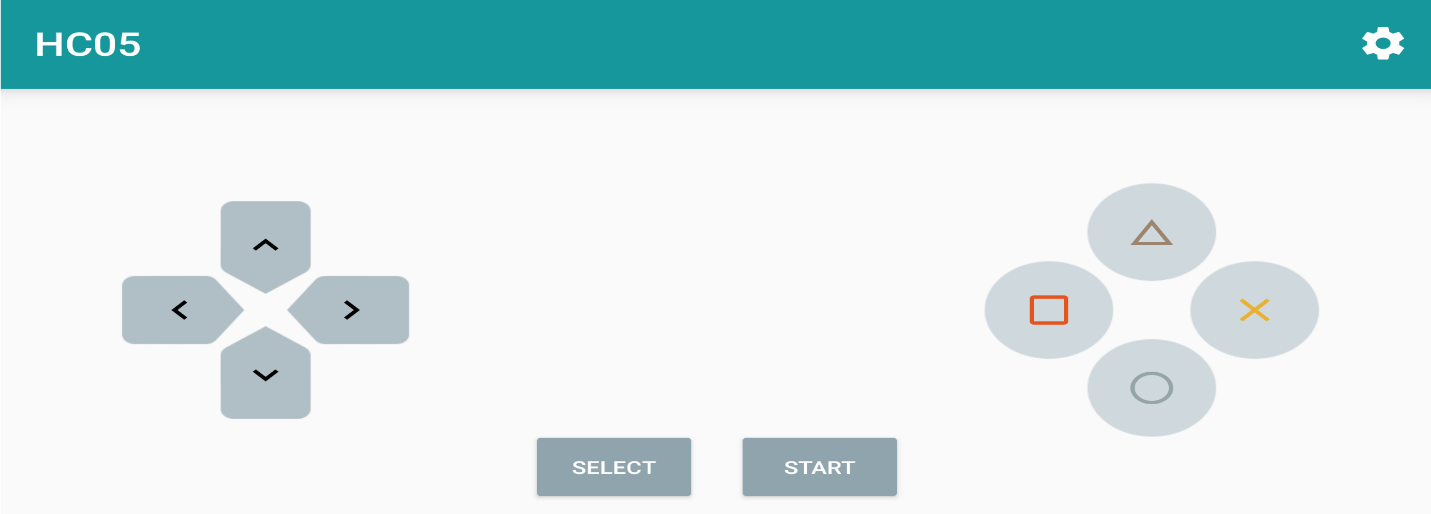
12. Alte resurse necesare: aplicația Android Arduino bluetooth controller, un laptop, cablu USB, alimentator, mediul de dezvoltare Processing, mediul de dezvoltare Arduino

**Exemplu de rulare:**

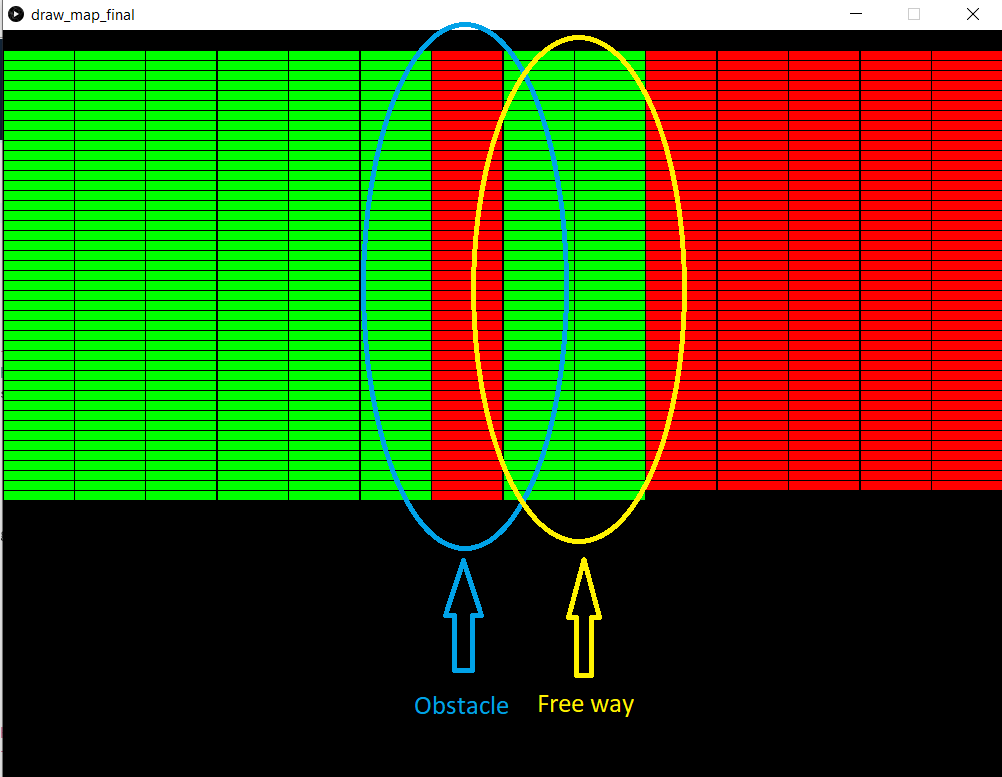
1. Realizarea montajului:



1. Conectarea ansamblului la aplicație:



1. Rezultatele obținute:



1. **Detalii de implementare**

În acest capitol sunt prezentate niște detalii din codul programului. Prima dată este descris codul sursă al programului rulat pe placa Arduino Uno, după care este prezentat codul sursă scris în mediul Processing care desenează mediul perceput de robot.

***Controller robot (mediul Arduino):***

În următoarea secțiune de cod este prezentat partea de **inițialiare** a programului, declararea variabilelor folosite în funcții și includerea bibliotecii Servo.h folosită pentru motorul de tip servo și bibliotecii SoftwareSerial.h care este utilizată în comunicarea între modulul Bluetooth și aplicația Android de pe telefon.

#include <Servo.h>

#include <SoftwareSerial.h>

// Pinii motor 1

#define mpin00 5

#define mpin01 6

// Pinii motor 2

#define mpin10 3

#define mpin11 11

Servo srv;

int angle=-1;

const int trigPin = 9;

const int echoPin = 10;

const int piezoPin = 7;

long duration;

int distance;

SoftwareSerial mySerial(12,2); //RX, TX

Metoda **setup** este folosită pentru a declara tipul fiecărui pin, tot aici sunt apelate niște metode de inițializare, de exemplu apelul Serial.begin(9600); pornește comunicația serială cu cu baud rate 9600.

void setup() {

// configurarea pinilor motor ca iesire, initial valoare 0

digitalWrite(mpin00, 0);

digitalWrite(mpin01, 0);

digitalWrite(mpin10, 0);

digitalWrite(mpin11, 0);

pinMode (mpin00, OUTPUT);

pinMode (mpin01, OUTPUT);

pinMode (mpin10, OUTPUT);

pinMode (mpin11, OUTPUT);

// pin LED

pinMode(13, OUTPUT);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

pinMode(piezoPin, OUTPUT); // Pin folosit de Buzzer

Serial.begin(9600); // Starts the serial communication

mySerial.begin(9600);

}

Funcția **StartMotor** are 4 parametrii: doi pini motorului DC, direcția și viteza motorului. Această funcție este folosită pentru a controla un motor DC.

void StartMotor (int m1, int m2, int forward, int speed)

{

if (speed==0) // oprire

{

digitalWrite(m1, 0);

digitalWrite(m2, 0);

}

else

{

if (forward)

{

digitalWrite(m2, 0);

analogWrite(m1, speed); // folosire PWM

}

else

{

digitalWrite(m1, 0);

analogWrite(m2, speed);

}

}

}

Metoda **delayStopped** are ca și parametru numărul de milisecunde cât motorul este oprit, aceasta este o metodă de siguranță și execută oprirea ambelor motoare.

void delayStopped(int ms)

{

StartMotor (mpin00, mpin01, 0, 0);

StartMotor (mpin10, mpin11, 0, 0);

delay(ms);

}

**TestServo** are ca și funcționalitate mișcarea motorului de tip servo cu un unghi dat. Pin reprezintă pinul la care este conectat motorul servo, iar angle este unghiul cu care se rotește motorul servo.

void testServo(int pin){

srv.attach(pin);

for(int i=30; i<=170; i=i+10){

srv.write(i);

angle=i;

// Serial.print("Angle: ");

Serial.print(angle);

readUltraSonic();

delay(100);

}

delay(400);

srv.detach(); }

Această metodă citește valoarea returnată de senzorul **Ultrasonic** și după aplicarea unei formule returnează distanța până la obiectul din fața senzorului. Tot această metodă pornește Buzzerul, atunci când distanța este mai mică decât 40, care emită un sunet în funcție de distanță până la obiect.

int readUltraSonic(){

// Clears the trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance= duration\*0.034/2;

//Serial.print("Distance: ");

Serial.print(" ");

Serial.println(distance);

if (distance < 40) {

tone(piezoPin,10000/distance,500);

}}

Funcția **startMotors** este destul de complexă. Citește într-o variabilă caracterul trimis de aplicația Android prin modulul de Bluetooth. În funcție de caracterul citit pornește cele două motoare: caracterul ’w’ – înainte, caracterul ’s’ – înapoi, , caracterul ’a’ – stânga , caracterul ’d’ – dreapta. În cazul altor caractere motorul este oprit cu metoda delayStopped.

void startMotors(){

if (mySerial.available()){

char com=mySerial.read();

Serial.write(com);

if( com=='w' || com=='W'){

StartMotor (mpin00, mpin01, 1, 128);

StartMotor (mpin10, mpin11, 1, 128);

delay (500); // Cât timp e motorul pornit

}

if( com=='s' || com=='S'){

StartMotor (mpin00, mpin01, 0, 128);

StartMotor (mpin10, mpin11, 0, 128);

delay (500);

}

if (com=='d' || com=='D'){

StartMotor (mpin00, mpin01, 1, 128);

StartMotor (mpin10, mpin11, 0, 128);

delay (100);

}

if( com=='a' || com=='A'){

StartMotor (mpin00, mpin01, 0, 128);

StartMotor (mpin10, mpin11, 1, 128);

delay (100);

}

delayStopped(100);}}

Metoda **loop** este cea care se rulează continuu, în această metodă este apelată testServo care controlează motorul de tip servo și metoda startMotors() care prelucrează datele primite prin modulul Bluetooth și controlează motoarele de tip DC.

void loop() {

// Pornirea motorului Servo

testServo(8);

//delay(250);

startMotors();}

***Desenarea hărții (mediul Processing) :***

***1.*** Inițializarea portului serial pentru citire (citire până la “new line”). Crearea ferestrei :

import processing.serial.\*;

Serial myPort;

void **setup**() {

size(1000, 1000); // dimensiunea ferestrei

lastAngle=map(lastAngle,30,170,0,width); //mapare valoare in //dimensiunile ferestrei

int lastport = Serial.list().length; //evita porturile seriale de //pe placa de baza

String portName = Serial.list()[lastport-1];

myPort=new Serial(this, portName, 9600); //baudRate 9600

myPort.bufferUntil(’\n’); //citeste pana intalneste ‘newline’

background(0);

}

***2.***  Sunt inițializate variabilele care descriu trasarea. Trasarea e constituită prin alipirea mai multor dreptunghiuri de diferite culori :

int y=10,dim;

float realDist; //valoarea primita de la senzor nemapata la dim ferestrei

float rectangleWidth; // lungimea unui dreptunghi alipit

float lastAngle=20; // incepe de la unghiul cu care incepe

// servo motorul

boolean kept=false; //evita desenarea primul dreptunghi(latime=0)

***3.*** Trasarea propriu zisă. Se preiau pe portul serial perechi de valori (unghi=orientarea senzorului ultrasonic dată de servo motor, distanța=valoarea măsurată de ultrasonic la un eșantion). Dacă distanța depășește o valoare impusă (exemplul aici este de 50 cm) dreptunghiul desenat va fi verde cu semnificația că în fața senzorului nu este obstacol și este cale liberă pentru deplasare. În caz contrar dreptunghiul desenat este roșu cu semnificația că în fața se află un obstacol la o distanța mai mică de 40 cm. Estimarea a cât de aproape de obstacol se află nu e dată de această hartă ci de un buzzer care îsi modifică intensitatea sonoră în funcție de valoarea din intervalul [0,50] .

void **draw**() {

String inString = myPort.readStringUntil(’\n’);

if (inString != null) {

String[] list=split(inString, ’ ’); // se primeste o // pereche (unghi, valoare)

float angle = float(list[0]);

float dist = float(list[1]);

realDist=dist; //pastreaza pentru o comparatie viitoare

if(int(angle)==30){ // s-a desenat o linie completa ,

// pregateste urmatoarea linie

y=y+10;

}

dist=map(dist, 3, 400, 0, height); // mapare corespunzatoare

angle=map(angle, 30, 170 , 0,width);

dim=(int)(abs(lastAngle-angle)); // dimensiune dreptunghi

if(kept==false){

kept=true; //primul dreptunghi va avea lungime 0

rectangleWidth=dim;

}

if(realDist > 40)

fill(0, 255, 0); // cale libera => dreptunghi verde

else

fill(255, 0, 0); // obstacol in fata => dreptunghi rosu

rect((int)angle,y,rectangleWidth,10);

}

}

1. **Concluzii**

În cadrul acestui proiect s-a realizat implementarea unui program, care împreună cu kitul robotic formează un sistem controlat de o aplicație Android prin care se realizează mișcarea unui robot pe baza informațiilor citite de senzori. Lucrând la acest proiect am dobândit următoarele cunoștințe: înțelegerea modului de funcționare a interfeței seriale, controlul motorului de tip servo și DC, controlul Buzzer-ului, preluarea de date de la un senzor Ultrasonic, transmiterea de date folosind modulul Bluetooth HC-05 și realizarea unei interfețe grafice pentru afișarea traseului robotului folosind mediul Processing.

1. **Dezvoltări ulterioare**

Propunem urmatoarele dezvoltari :

• Implementarea unei aplicații proprii Android de comunicare cu microncontroller-ul de pe kitul de dezvoltare prin intermediul modulului bluetooth HC-05.

• Înlocuirea cablului USB printr-un alt modul bluetooth HC-05 sau HC-06 datorită limitărilor date de lungimea cablului.

• Crearea unei logici de stabilire a poziției robotului raportată la harta creată în mediul Processing.

• Afișarea hărții pe ecranul telefonului în locul ecranului de la calculator prin aplicația dezvoltată anterior

• Salvarea traseului robotului și afișarea acestuia

• Navigare în două moduri : autonom(traversarea autonomă a unui traseu și realizarea tuturor pașilor de mai sus) sau controlat (prin aplicație)

**Referințe:**

* <https://www.arduino.cc/en/Reference/softwareSerial>
* <https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/336-3.pdf>
* <https://processing.org/reference/rect_.html>
* <https://arduinobasics.blogspot.com/2012/11/hc-sr04-ultrasonic-sensor.html>