

UNIVERSITATEA TITU MAIORESCU

FACULTATEA: INFORMATICĂ

DEPARTAMENT: INFORMATICĂ

Programa de studii: INFORMATICĂ

DISCIPLINA: INTELIGENȚĂ ARTIFICIALĂ

IA - Testul de evaluare nr. 1

Robot autonom terestru – 4WD (ATV)

Nr.crt	Grupa	Numele și prenumele	Semnătură student	Notă evaluare
1				
2				
3				
4				
5				
6				

Data: ____ / ____ / ____

CS I dr.ing.

Lucian Ștefăniță GRIGORE

Conf.dr.ing.

Ș.l. dr.ing.

Iustin PRIESCU

Dan-Laurențiu GRECU



Cuprins

1.	INTRODUCERE	3
2.	SOFTWARE 4WD (ATV) AUTONOM – OCOLIRE OBSTACOLE	6
2.1	Senzor de distanță sharp_GP2Y0D810Z0F_10cm.....	7
3.	SOFTWARE 4WD (ATV) – TELECOMANDAT în INFRAROȘU.....	8
4.	SOFTWARE 4WD (ATV) – BLUETHOOT TELEFON MOBIL.....	11
5.	SOFTWARE 4WD (ATV) – WIRELESS.....	18
5.1	WiFi car.....	18
5.2	WiFi car Reloaded - I.....	21
5.3	WiFi car Reloaded - II.....	24
6.	INTERFAȚA de COMUNICARE cu PIC Microcontroler	25

1. INTRODUCERE

Progresele recente ale bazelor computaționale ale roboticii au condus la dezvoltarea de noi algoritmi și metodologii eficiente pentru abordarea mai multor probleme fundamentale în robotică.

Îmbunătățirea susținută a performanței procesorului și a tehnologiei senzorilor au făcut posibilă construirea de vehicule care sunt în mod substanțial autonome și capabile să realizeze o mare varietate de sarcini în lumea reală.

O mare parte din efortul investit pentru a crește abilitatea roboților mobili de a-și lua propriile decizii a abordat aspectele esențiale ale mobilității, și anume localizarea și cartografierea, care sunt puternic dependente de interpretarea datelor și de fuziune.

Localizarea constă în estimarea poziției și orientării robotului mobil în raport cu un cadru inerțial dat, este considerată o problemă foarte dificilă din cauza incertitudinii inerente și a non-determinismului lumii reale.

Maparea este o problemă înrudită, care poate fi descrisă ca fiind sarcina de a construi o reprezentare a lumii pe baza informațiilor senzorilor. În timp ce se deplasează într-un mediu interior, de cele mai multe ori trebuie să fie cunoscute numai punctele robotului pe plan.

Poziția poate fi specificată de un vector de configurație:

$$V(t) = [x(t), y(t), \psi(t)] \cdot, \quad 1.1$$

în care: x și y reprezintă poziția centrului de greutate al robotului și ψ reprezintă unghiul față de un anumit cadru de referință inerțial.

Teoria care stipulează cele de mai sus consideră o dreaptă oarecare $\{\Delta\}$ și un vector (\bar{a}) necoplanar cu aceasta (Fig. 1-1). Prin punctul A de aplicație al vectorului se construiește dreapta $\Delta_1 \parallel \Delta$, iar din vârful B se duce $BB_1 \perp \Delta_1$. Se duc apoi perpendicularele comune $\{AA_1\}$ și $\{B'B_1\}$. Lungimea segmentului $|AA_1| \equiv \{a_\Delta\}$, reprezintă proiecția vectorului (\bar{a}) pe direcția $\{\Delta\}$.

Astfel, cu observația că $\bar{u}_\Delta = \text{versor}\Delta$ și $\bar{u}_\Delta = 1$, se poate scrie:

$$a_\Delta = \text{pr}_\Delta \bar{a} = |\bar{a}| \cdot \cos \alpha = |\bar{a}| \cdot |\bar{u}_\Delta| \cdot \cos \alpha = \bar{a} \cdot \bar{u}_\Delta [m]. \quad 1.2$$

$$\begin{cases} a_\Delta = 0, & \alpha = \frac{\pi}{2} [-] \\ a_\Delta > 0, & 0 < \alpha < \frac{\pi}{2} [-] \\ a_\Delta < 0, & \frac{\pi}{2} < \alpha < \pi [-] \end{cases} \quad 1.3$$

$$\begin{cases} a_x = \text{pr}_{Ox} \bar{a} = \bar{a} \cdot \bar{i} = |\bar{a}| \cdot \cos \alpha [-] \\ a_y = \text{pr}_{Oy} \bar{a} = \bar{a} \cdot \bar{j} = |\bar{a}| \cdot \cos \beta [-] \\ a_z = \text{pr}_{Oz} \bar{a} = \bar{a} \cdot \bar{k} = |\bar{a}| \cdot \cos \gamma [-] \end{cases} \quad 1.4$$

unde: $\{\bar{i}, \bar{j}, \bar{k}\}$ versorii axelor de coordonate.

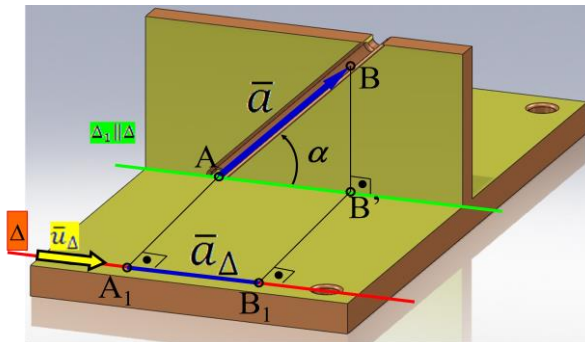


Fig. 1-1 Proiecția unui vector pe o axă

Direcția unui vector față de un sistem de coordonate cartezian (Fig. 1-2) și față de un sistem de coordonate sferice (Fig. 1-3).

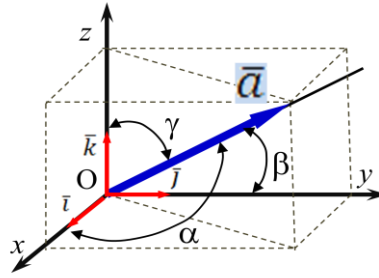


Fig. 1-2 Proiecția unui vector pe un sistem de coordonate cartezian

$$\begin{cases} \cos^2 \alpha + \cos^2 \beta + \cos^2 \gamma = 1 & [-] \\ |\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} & [-] \end{cases} \quad 1.5$$

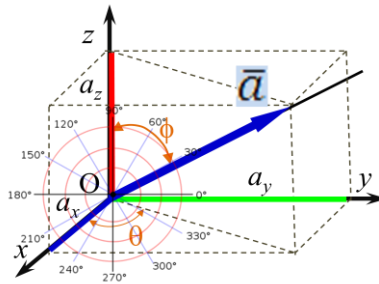


Fig. 1-3 Proiecția unui vector pe un sistem de coordonate sferice

$$\begin{cases} a_x = |\vec{a}| \cdot \cos \phi \cdot \cos \theta & [-] \\ a_y = |\vec{a}| \cdot \cos \phi \cdot \sin \theta & [-] \\ a_z = |\vec{a}| \cdot \sin \phi & [-] \end{cases} \quad 1.6$$

Ipoteza de simplificare în cazul general este că terenul traversat de robot este orizontal și plat și de obicei caracteristicile pavajului, cum ar fi tipurile de suprafață și altele sunt neglijate. Cu toate acestea, nu este cazul în mediile în aer liber, în care topografia și tipurile de suprafețe ale terenului au o influență majoră asupra deplasării și navigării roboților și ar trebui să fie luate în considerare.

Neregularitățile tipice întâlnite în medii în aer liber sunt: denivelările, panta și caracteristica solului (solid, mâlos, nisip etc.).

În astfel de medii, robotul nu se mai va deplasa pe o suprafață plană orizontală, dar atitudinea sa va fi în continuă schimbare din cauza neregularităților de pe teren și, prin urmare, poziția sa va avea șase grade de libertate. Una dintre metodele de planificare a dinamicii vehiculelor mobile care se deplasează pe terenuri generale introduce ca variabile: geometria căii de rulare, viteza, timpul, topografia terenului, obstacolele și caracteristica de mobilitate.

Calea de rulare este reprezentată în modelul analitic printr-o curbă Spline. Se pleacă de la un traseu optim (fără obstacole), iar în timp de la iterație la iterație se introduc elementele amintite. S-a ales ca teren unul din zona muntoasă, întrucât acoperă o geometrie foarte variată.

Planificarea mișcării vehiculelor mobile constă în selectarea geometriei căii de rulare, a vitezei necesare evitării obstacolelor în condițiile unei funcții cost cât mai reduse (economia de energie).

Se are în vedere că o planificare mai puțin fericită a vitezelor de deplasare poate conduce la pierderea traseului și implicit la un consum de energie mai mare pentru a reveni pe traseul impus sau determinat.

Problema planificării mișcării este formulată ca o optimizare în etape.

- optimizarea vitezei de-a lungul unei căi sunt calculate pentru a minimiza timpul de mișcare;
- minimizarea timpului de deplasare, algoritmul selectează cele mai mari viteze posibile fără a introduce constrângeri dinamice suplimentare cu privire la alunecarea tangențială și laterală, trecerea peste obstacol, coeficientul de aderență;
- evitarea obstacolelor și reducerea timpului necesar efectuării manevrelor de ocolire.

Toate etapele converg către o optimizare globală, în care se introduce un factor de mobilitate care amplifică coeficientul inițial de frecare dintre vehicul și sol. Mobilitatea „0” este asociată regiunilor inaccesibile, iar valoarea „1” este asociată drumurilor perfecte. Căile de rulare normale: păduri, teren accidentat etc. au valori cuprinse între $0 \div 1$.

Curba de viteză se obține în urma combinării ecuațiilor de cinematică, dinamică și de topografie a terenului. De asemenea sunt introduse constrângerile legate de economia de energie, distanța până la obstacole, timpul de rulare pe un tip de teren.

2. SOFTWARE 4WD (ATV) AUTONOM – OCOLIRE OBSTACOLE

Funcția "go(int speedLeft, int speedRight)" nu trebuie modificată. Cei doi parametri reprezintă vitezele celor două motoare (între $-255 \div +255$, cu o zonă moartă între $-100 \div +100$). Astfel, pentru ca robotul să meargă cu viteza maximă:

- **înainte**, trebuie apelat codul "go(+255, +255)";
- **înapoi**, se apelează "go(-255, -255)";
- **rotire**, se apelează "go(-255, +255)".

```
int MOTOR2_PIN1 = 3;
int MOTOR2_PIN2 = 5;
int MOTOR1_PIN1 = 6;
int MOTOR1_PIN2 = 9;
```

```
void setup() {
  pinMode(MOTOR1_PIN1, OUTPUT);
  pinMode(MOTOR1_PIN2, OUTPUT);
  pinMode(MOTOR2_PIN1, OUTPUT);
  pinMode(MOTOR2_PIN2, OUTPUT);
  Serial.begin(9600);
}
```

```
void loop() {
  go(255,-255);
  delay(1000);
  go(-255,-255);
  delay(1000);
  go(-255,255);
  delay(1000);
  go(255,255);
  delay(1000);
}
```

```
void go(int speedLeft, int speedRight) {
  if (speedLeft > 0) {
    analogWrite(MOTOR1_PIN1, speedLeft);
    analogWrite(MOTOR1_PIN2, 0);
  }
  else {
    analogWrite(MOTOR1_PIN1, 0);
    analogWrite(MOTOR1_PIN2, -speedLeft);
  }

  if (speedRight > 0) {
    analogWrite(MOTOR2_PIN1, speedRight);
    analogWrite(MOTOR2_PIN2, 0);
  }
  else {
    analogWrite(MOTOR2_PIN1, 0);
    analogWrite(MOTOR2_PIN2, -speedRight);
  }
}
```

2.1 Senzor de distanță sharp_GP2Y0D810Z0F_10cm

```
void setup() {  
    Serial.begin(9600);  
    pinMode(7, INPUT);  
    pinMode(8, INPUT);  
}  
  
void loop() {  
    int valoareSenzor10cm = digitalRead(7);  
    int valoareSenzor5cm = digitalRead(8);  
    Serial.print("Senzor 10cm: ");  
    Serial.print(valoareSenzor10cm, DEC);  
    Serial.print(" Senzor 5cm: ");  
    Serial.println(valoareSenzor5cm, DEC);  
}
```

3. SOFTWARE 4WD (ATV) – TELECOMANDAT în INFRAROȘU¹

```
#include <IRremote.h>

#define SPEED 255

int MOTOR2_PIN1 = 3;
int MOTOR2_PIN2 = 5;
int MOTOR1_PIN1 = 6;
int MOTOR1_PIN2 = 9;

int IR_RECV_PIN = 4;

IRrecv irrecv(IR_RECV_PIN);

decode_results results;

long key = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(MOTOR1_PIN1, OUTPUT);
  pinMode(MOTOR1_PIN2, OUTPUT);
  pinMode(MOTOR2_PIN1, OUTPUT);
  pinMode(MOTOR2_PIN2, OUTPUT);
  irrecv.enableIRIn(); // Start the receiver
}

void dump(decode_results *results) {
  int count = results->rawlen;
  if (results->decode_type == UNKNOWN) {
    Serial.print("Unknown encoding: ");
  }
  else if (results->decode_type == NEC) {
    Serial.print("Decoded NEC: ");
  }
  else if (results->decode_type == SONY) {
    Serial.print("Decoded SONY: ");
  }
  else if (results->decode_type == RC5) {
    Serial.print("Decoded RC5: ");
  }
  else if (results->decode_type == RC6) {
    Serial.print("Decoded RC6: ");
  }
  else if (results->decode_type == PANASONIC) {
    Serial.print("Decoded PANASONIC - Address: ");
    Serial.print(results->panasonicAddress, HEX);
    Serial.print(" Value: ");
  }
  else if (results->decode_type == JVC) {
    Serial.print("Decoded JVC: ");
  }
```

¹ <https://www.robofun.ro/docs/flexybot-mare-4motoareir.ino>


```
}

key = results->value;

}

void loop() {

    //reverse direction key
    if (key == 0x10EF10EF) {
        go(-SPEED, SPEED);
    }

    //reverse direction key
    if (key == 0x10EF807F) {
        go(SPEED, -SPEED);
    }

    //speed increase key
    if (key == 0x10EFA05F) {
        go(SPEED, SPEED);
    }

    //speed decrease key
    if (key == 0x10EF00FF) {
        go(-SPEED, -SPEED);
    }

    //stop key
    if (key == 0x10EF20DF) {
        go(0, 0);
    }

    //OFF key
    if (key == 0x10EFD827) {
        go(0, 0);
    }

    key = 0;

    if (irrecv.decode(&results)) {
        dump(&results);
        irrecv.resume(); // Receive the next value
        Serial.println(key, HEX);
        Serial.println();
    }

}

void go(int speedLeft, int speedRight) {
    if (speedLeft > 0) {
        analogWrite(MOTOR1_PIN1, speedLeft);
        analogWrite(MOTOR1_PIN2, 0);
    }
    else {
        analogWrite(MOTOR1_PIN1, 0);
    }
}
```

```
    analogWrite(MOTOR1_PIN2, -speedLeft);
  }

  if (speedRight > 0) {
    analogWrite(MOTOR2_PIN1, speedRight);
    analogWrite(MOTOR2_PIN2, 0);
  } else {
    analogWrite(MOTOR2_PIN1, 0);
    analogWrite(MOTOR2_PIN2, -speedRight);
  }
}
```

Senzor activ: TSOP4838

```
#include "IRremote.h"

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

4. SOFTWARE 4WD (ATV) – BLUETHOOT TELEFON MOBIL²

```
#define VERSION    "\n\nAndroTest V2.0 - @kas2014\ndemo for V5.x App"

// V2.0 changed to pure ASCII Communication Protocol ** not backward compatible **
// V1.4 improved communication errors handling
// V1.3 renamed for publishing, posted on 09/05/2014
// V1.2 Text display ** not backward compatible **
// V1.1 Integer display
// V1.0 6 buttons + 4 data char implemented

// Demo setup:
// Button #1 controls pin #13 LED
// Button #4 toggle datafield display rate
// Button #5 configured as "push" button (momentary)
// Other buttons display demo message

// Arduino pin#2 to TX BlueTooth module
// Arduino pin#3 to RX BlueTooth module
// make sure your BT board is set @57600 bps
// better remove SoftSerial for PWM based projects

// For Mega 2560:
// remove #include "SoftwareSerial.h", SoftwareSerial mySerial(2,3);
// search/replace mySerial >> Serial1
// pin#18 to RX Bluetooth module, pin#19 to TX Bluetooth module

#include "SoftwareSerial.h"

#define STX        0x02
#define ETX        0x03
#define ledPin     13
#define SLOW       750           // Datafields refresh rate (ms)
#define FAST       250          // Datafields refresh rate (ms)
#define SPEEDLEFT  160
#define SPEEDRIGHT 128

SoftwareSerial mySerial(4,2);

int MOTOR2_PIN1 = 3;
int MOTOR2_PIN2 = 5;
int MOTOR1_PIN1 = 6;
int MOTOR1_PIN2 = 9;

byte cmd[8] = {0, 0, 0, 0, 0, 0, 0, 0}; // bytes received
byte buttonStatus = 0;                  // first Byte sent to Android device
long previousMillis = 0;                 // will store last time Buttons status was updated
long sendInterval = SLOW;                // interval between Buttons status transmission (milliseconds)
String displayStatus = "lol";           // message to Android device

void setup() {

    pinMode(MOTOR1_PIN1, OUTPUT);
```

² <https://www.robofun.ro/docs/flexybot-mare-4motoarebluetooth.ino>

```

pinMode(MOTOR1_PIN2, OUTPUT);
pinMode(MOTOR2_PIN1, OUTPUT);
pinMode(MOTOR2_PIN2, OUTPUT);

Serial.begin(9600);
mySerial.begin(115200);
mySerial.print("$");
mySerial.print("$");
mySerial.print("$");
delay(100);
mySerial.println("U,9600,N");
mySerial.begin(9600); // Start bluetooth serial at 9600           // 57600 = max value for softserial
pinMode(ledPin, OUTPUT);
Serial.println(VERSION);
while (mySerial.available()) mySerial.read();    // empty RX buffer
}

void loop() {
  if (mySerial.available()) {                      // data received from smartphone
    delay(2);
    cmd[0] = mySerial.read();
    if (cmd[0] == STX) {
      int i = 1;
      while (mySerial.available()) {
        delay(1);
        cmd[i] = mySerial.read();
        if (cmd[i] > 127 || i > 7) break; // Communication error
        if ((cmd[i] == ETX) && (i == 2 || i == 7)) break; // Button or Joystick data
        i++;
      }
      if (i == 2)    getButtonState(cmd[1]); // 3 Bytes ex: < STX "C" ETX >
      else if (i == 7)    getJoystickState(cmd); // 6 Bytes ex: < STX "200" "180" ETX >
    }
  }
  sendBlueToothData();
}

void sendBlueToothData() {
  static long previousMillis = 0;
  long currentMillis = millis();
  if (currentMillis - previousMillis > sendInterval) { // send data back to smartphone
    previousMillis = currentMillis;

    // Data frame transmitted back from Arduino to Android device:
    // < 0X02 Buttons state 0X01 DataField#1 0x04 DataField#2 0x05 DataField#3 0x03 >
    // < 0X02 "01011" 0X01 "120.00" 0x04 "-4500" 0x05 "Motor enabled" 0x03 > // example

    mySerial.print((char)STX);                      // Start of Transmission
    mySerial.print(getButtonStatusString()); mySerial.print((char)0x1); // buttons status feedback
    mySerial.print(GetdataInt1()); mySerial.print((char)0x4); // datafield #1
    mySerial.print(GetdataFloat2()); mySerial.print((char)0x5); // datafield #2
    mySerial.print(displayStatus); // datafield #3
    mySerial.print((char)ETX);                      // End of Transmission
  }
}

String getButtonStatusString() {

```

```
String bStatus = "";
for (int i = 0; i < 6; i++) {
    if (buttonStatus & (B100000 >> i))    bStatus += "1";
    else                                bStatus += "0";
}
return bStatus;
}

int GetDataInt1() {                // Data dummy values sent to Android device for demo purpose
    static int i = -30;            // Replace with your own code
    i ++;
    if (i > 0)    i = -30;
    return i;
}

float GetDataFloat2() {            // Data dummy values sent to Android device for demo purpose
    static float i = 50;           // Replace with your own code
    i -= .5;
    if (i < -50)    i = 50;
    return i;
}

void getJoystickState(byte data[8]) {
    int joyX = (data[1] - 48) * 100 + (data[2] - 48) * 10 + (data[3] - 48); // obtain the Int from the ASCII representation
    int joyY = (data[4] - 48) * 100 + (data[5] - 48) * 10 + (data[6] - 48);
    joyX = joyX - 200;                                // Offset to avoid
    joyY = joyY - 200;                                // transmitting negative numbers

    if (joyX < -100 || joyX > 100 || joyY < -100 || joyY > 100)    return; // communication error

    // Your code here ...
    Serial.print("Joystick position: ");
    Serial.print(joyX);
    Serial.print(", ");
    Serial.println(joyY);

    if (joyY >= 90) {

        //md.setM1Speed(SPEED);
        //stopIfFault();
        //md.setM2Speed(-SPEED);
        //stopIfFault();

        go(SPEEDLEFT,SPEEDRIGHT);

        Serial.println("inainte");

    } else if (joyX == 0 && joyY == 0) {
        //md.setM1Speed(0);
        //stopIfFault();
        //md.setM2Speed(0);
        //stopIfFault();

        go(0,0);
        Serial.println("stop");
    }
}
```

```
if (joyY < -90) {

    //md.setM1Speed(-SPEED);
    //stopIfFault();
    //md.setM2Speed(SPEED);
    //stopIfFault();
    go(-SPEEDLEFT,-SPEEDRIGHT);
    Serial.println("inapoi");

} else if (joyX == 0 && joyY == 0) {
    //md.setM1Speed(0);
    //stopIfFault();
    //md.setM2Speed(0);
    //stopIfFault();
    go(0,0);
    Serial.println("stop");

}

if (joyX >= 90) {

    //md.setM1Speed(SPEED);
    //stopIfFault();
    //md.setM2Speed(SPEED);
    //stopIfFault();

    go(SPEEDLEFT,-SPEEDRIGHT);
    Serial.println("dreapta");

} else if (joyX == 0 && joyY == 0) {
    //md.setM1Speed(0);
    //stopIfFault();
    //md.setM2Speed(0);
    //stopIfFault();
    go(0,0);
    Serial.println("stop");
}

if (joyX < -90) {

    //md.setM1Speed(-SPEED);
    //stopIfFault();
    //md.setM2Speed(-SPEED);
    //stopIfFault();

    go(-SPEEDLEFT,SPEEDRIGHT);
    Serial.println("stanga");

} else if (joyX == 0 && joyY == 0) {
    //md.setM1Speed(0);
    //stopIfFault();
    //md.setM2Speed(0);
    //stopIfFault();
    go(0,0);
    Serial.println("stop");
}

}
```

```
void getButtonState(int bStatus) {
  switch (bStatus) {
    // ----- BUTTON #1 -----
    case 'A':
      buttonStatus |= B000001;    // ON
      Serial.println("\n** Button_1: ON **");
      // your code...
      displayStatus = "LED <ON>";
      Serial.println(displayStatus);
      digitalWrite(ledPin, HIGH);
      break;
    case 'B':
      buttonStatus &= B111110;    // OFF
      Serial.println("\n** Button_1: OFF **");
      // your code...
      displayStatus = "LED <OFF>";
      Serial.println(displayStatus);
      digitalWrite(ledPin, LOW);
      break;

    // ----- BUTTON #2 -----
    case 'C':
      buttonStatus |= B000010;    // ON
      Serial.println("\n** Button_2: ON **");
      // your code...
      displayStatus = "Button2 <ON>";
      Serial.println(displayStatus);
      break;
    case 'D':
      buttonStatus &= B111101;    // OFF
      Serial.println("\n** Button_2: OFF **");
      // your code...
      displayStatus = "Button2 <OFF>";
      Serial.println(displayStatus);
      break;

    // ----- BUTTON #3 -----
    case 'E':
      buttonStatus |= B000100;    // ON
      Serial.println("\n** Button_3: ON **");
      // your code...
      displayStatus = "Motor #1 enabled"; // Demo text message
      Serial.println(displayStatus);
      break;
    case 'F':
      buttonStatus &= B111011;    // OFF
      Serial.println("\n** Button_3: OFF **");
      // your code...
      displayStatus = "Motor #1 stopped";
      Serial.println(displayStatus);
      break;

    // ----- BUTTON #4 -----
    case 'G':
      buttonStatus |= B001000;    // ON
      Serial.println("\n** Button_4: ON **");
```

```
// your code...
displayStatus = "Datafield update <FAST>";
Serial.println(displayStatus);
sendInterval = FAST;
break;
case 'H':
    buttonStatus &= B110111; // OFF
    Serial.println("\n** Button_4: OFF **");
    // your code...
    displayStatus = "Datafield update <SLOW>";
    Serial.println(displayStatus);
    sendInterval = SLOW;
    break;

// ----- BUTTON #5 -----
case 'I': // configured as momentary button
    // buttonStatus |= B010000; // ON
    Serial.println("\n** Button_5: ++ pushed ++ **");
    // your code...
    displayStatus = "Button5: <pushed>";
    break;
// case 'J':
//     buttonStatus &= B101111; // OFF
//     // your code...
//     break;

// ----- BUTTON #6 -----
case 'K':
    buttonStatus |= B100000; // ON
    Serial.println("\n** Button_6: ON **");
    // your code...
    displayStatus = "Button6 <ON>"; // Demo text message
    break;
case 'L':
    buttonStatus &= B011111; // OFF
    Serial.println("\n** Button_6: OFF **");
    // your code...
    displayStatus = "Button6 <OFF>";
    break;
}
// -----
}

void go(int speedLeft, int speedRight) {
    if (speedLeft > 0) {
        analogWrite(MOTOR1_PIN1, speedLeft);
        analogWrite(MOTOR1_PIN2, 0);
    }
    else {
        analogWrite(MOTOR1_PIN1, 0);
        analogWrite(MOTOR1_PIN2, -speedLeft);
    }
}

if (speedRight > 0) {
    analogWrite(MOTOR2_PIN1, speedRight);
    analogWrite(MOTOR2_PIN2, 0);
} else {
```



```
    analogWrite(MOTOR2_PIN1, 0);
    analogWrite(MOTOR2_PIN2, -speedRight);
  }
}
```

Modul Bluetooth

```
#include "SoftwareSerial.h";
int bluetoothTx = 2;
int bluetoothRx = 3;

SoftwareSerial bluetooth(bluetoothTx, bluetoothRx);

void setup()
{
    Serial.begin(9600);

    bluetooth.begin(115200);
    bluetooth.print("$$$");
    delay(100);
    bluetooth.println("U,9600,N");
    bluetooth.begin(9600);
}

void loop()
{
    if(bluetooth.available()) {
        char toSend = (char)bluetooth.read();
        Serial.print(toSend);
    }

    if(Serial.available()) {
        char toSend = (char)Serial.read();
        bluetooth.print(toSend);
    }
}
```

Exemplu:

```
void setup() {
    Serial.begin(9600);
}
long time = millis();
void loop() {
    if (Serial.available()){
        Serial.println(Serial.read());
    }
    if ((millis() - time) > 2000) {
        Serial.println(time);
        time = millis();
    }
}
```

5. SOFTWARE 4WD (ATV) – WIRELESS

5.1 WiFi car³

Programul ce va rula pe microcontrolerul ATmega32U4 al plăcii Arduino Yun va utiliza biblioteca software Bridge pentru a crea legătura dintre comanda motoarelor și comenzile web pe care le vom trimite prin conexiunea WiFi.

```
#include <Bridge.h>
#include <BridgeServer.h>
#include <BridgeClient.h>
BridgeServer server;
```

Configurarea și comanda motoarelor este similară cu utilizarea shield-ului în conjuncție cu o conexiune Bluetooth sau un algoritm local de tip evitare de obstacole sau urmărire de linie.

```
#define MOTOR2_PIN1 3
#define MOTOR2_PIN2 5
#define MOTOR1_PIN1 6
#define MOTOR1_PIN2 9
void setup() {
  Bridge.begin();
  pinMode(MOTOR1_PIN1, OUTPUT);
  pinMode(MOTOR1_PIN2, OUTPUT);
  pinMode(MOTOR2_PIN1, OUTPUT);
  pinMode(MOTOR2_PIN2, OUTPUT);
  server.listenOnLocalhost();
  server.begin();
}
void go(int speedLeft, int speedRight) {
  if (speedLeft > 0) {
    analogWrite(MOTOR1_PIN1, speedLeft);
    analogWrite(MOTOR1_PIN2, 0);
  } else {
    analogWrite(MOTOR1_PIN1, 0);
    analogWrite(MOTOR1_PIN2, -speedLeft);
  }
  if (speedRight > 0) {
    analogWrite(MOTOR2_PIN1, speedRight);
    analogWrite(MOTOR2_PIN2, 0);
  } else {
    analogWrite(MOTOR2_PIN1, 0);
    analogWrite(MOTOR2_PIN2, -speedRight);
  }
}
```

În cadrul secțiunii *loop()* se vor prelua, prin intermediul bibliotecii Bridge, conexiunile de comandă și se vor trimite către procedura *process()* pentru execuție.

```
void loop() {
  BridgeClient client = server.accept();
  if (client) {
```

³ <https://blog.robofun.ro/2017/02/07/proiect-wifi-car/>

```
        process(client);
        client.stop();
    }
    delay(50);
}

void process(BridgeClient client) {
    String command = client.readStringUntil('/');
    command.trim();
    if(command == "forward") {
        go(150,150);
        client.println("Forward");
        client.println("Vs: 150 Vd: 150");
    }
    if(command == "left") {
        go(100,150);
        client.println("Left");
        client.println("Vs: 100 Vd: 150");
    }
    if(command == "stop") {
        go(0,0);
        client.println("Stop");
    }
    if(command == "right") {
        go(150,100);
        client.println("Right");
        client.println("Vs: 150 Vd: 100");
    }
    if(command == "back") {
        go(-150,-150);
        client.println("Backward");
        client.println("Vs: -150 Vd: -150");
    }
}
```

După încărcarea programului se vor putea accesa comenzile WiFi Car de pe orice sistem ce deține un client web (browser) și este conectat la aceeași subrețea ca și placa Arduino Yun accesând următoarele URL-uri:

- mers înainte:
http://ip_placa_arduino_yun/arduino/forward/
- virare stânga:
http://ip_placa_arduino_yun/arduino/left/
- virare dreapta:
http://ip_placa_arduino_yun/arduino/right/
- oprire:
http://ip_placa_arduino_yun/arduino/stop/
- mers înapoi:
http://ip_placa_arduino_yun/arduino/back/

Comanda poate fi îmbunătățită prin introducerea unor comenzi de accelerare, frânare sau accentuare/încetinire a virării. Având în vedere faptul că placa Arduino Yun rulează implicit un server web (uhttp) se poate realiza o pagină html care să centralizeze accesul la URL-urile de comandă. Există două variante de copiere și localizare la nivelul sistemului de fișiere OpenWRT a fișierului html: https://create.arduino.cc/projecthub/Arduino_Scuola/arduino-yun-intro-to-web-server-6caf93

Utilizarea unui card microSD și includerea fișierului html în proiectul Arduino. Copierea fișierului html se va face automat la încărcarea programului. Această variantă este descrisă în următorul material: [Arduino Yún: Intro to web server](#).

Crearea manuală a fișierului html în linie de comandă. Aceasta este varianta propusă datorită simplității. Se va crea un fișier *wificar.html* în directorul */www* cu următorul conținut:

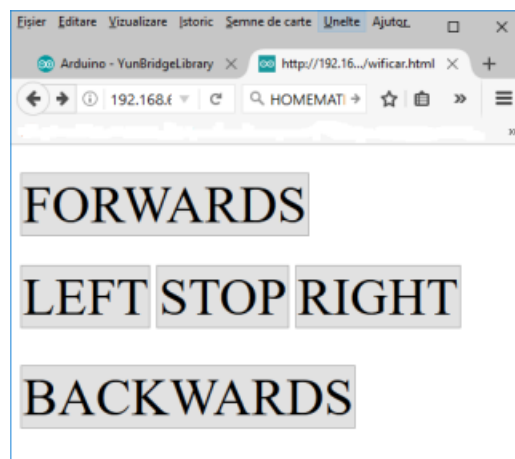
```
<html>
<head>
<style>
a.button {-webkit-appearance: button;
-moz-appearance: button;
appearance: button;
height:200px;
line-height:200px;
text-align:center;
text-decoration: none;
font-size: 50px;
color: initial;}
</style>
</head>
<body>
<a href="/arduino/forward/" class="button"
style="width:100%;">FORWARDS</a>
<br />
<a href="/arduino/left/" class="button"
style="width:35%;">LEFT</a>
<a href="/arduino/stop/" class="button"
style="width:30%;">STOP</a>
<a href="/arduino/right/" class="button"
style="width:35%;">RIGHT</a>
<br />
<a href="/arduino/back/" class="button"
style="width:100%;">BACKWARDS</a>
<br />
</body>
</html>
```

Se va verifica dacă în fișierul */etc/config/uhttp* variabila *option home* are valoarea */www*:

Server document root

option home /www

Se va accesa URL-ul: http://ip_placa_arduino_yun/wificar.html



Protocoloalele de comunicație TCP/IP sunt lente, mai ales protocolul HTTP. Nu vom avea aceeași viteză de răspuns ca în cazul unei mașini teleghidate RC obișnuite.

5.2 WiFi car Reloaded - I⁴

Programul ce va rula pe microcontrolerul ATmega32U4 este următorul (programarea plăcii se poate face, la fel ca și în cazul plăcii Arduino Yun, prin WiFi – Over The Air):

```
#define MOTOR2_PIN1 10
#define MOTOR2_PIN2 9
#define MOTOR1_PIN1 6
#define MOTOR1_PIN2 5
int vs = 0;
int vd = 0;
void setup() {
  pinMode(MOTOR1_PIN1, OUTPUT);
  pinMode(MOTOR1_PIN2, OUTPUT);
  pinMode(MOTOR2_PIN1, OUTPUT);
  pinMode(MOTOR2_PIN2, OUTPUT);
  Serial1.begin(9600); }
void loop() {
  if (Serial1.available()) {
    char c = (char)Serial1.read();
    switch (c) {
      case 'i':
        if (vs<245) vs=vs+10;
        if (vd<245) vd=vd+10;
        Serial1.print("Viteza: ");
        Serial1.print(vs);
        Serial1.print("/");
        Serial1.println(vd);
        break;
      case 'b':
        if (vs>-245) vs=vs-10;
        if (vd>-245) vd=vd-10;
        Serial1.print("Viteza: ");
        Serial1.print(vs);
        Serial1.print("/");
        Serial1.println(vd);
        break;
      case 's':
        if (vs<245) vs=vs+10;
        if (vd>-245) vd=vd-10;
        Serial1.print("Viteza: ");
        Serial1.print(vs);
        Serial1.print("/");
        Serial1.println(vd);
        break;
      case 'd':
        if (vs>-245) vs=vs-10;
        if (vd<245) vd=vd+10;
        Serial1.print("Viteza: ");
        Serial1.print(vs);
        Serial1.print("/");
        Serial1.println(vd);
        break;
```

⁴ <https://blog.robofun.ro/2017/02/14/proiect-wifi-car-reloaded-partea-i/>

```

    case 'x':
        vs=0;
        vd=0;
        Serial1.print("Viteza: ");
        Serial1.print(vs);
        Serial1.print("\n");
        Serial1.println(vd);
        break;
    }
}
Serial1.flush();
go(vs,vd);
delay(50);
}
void go(int speedLeft, int speedRight) {
    if (speedLeft > 0) {
        analogWrite(MOTOR1_PIN1, speedLeft);
        analogWrite(MOTOR1_PIN2, 0);
    }
    else {
        analogWrite(MOTOR1_PIN1, 0);
        analogWrite(MOTOR1_PIN2, -speedLeft);
    }
    if (speedRight > 0) {
        analogWrite(MOTOR2_PIN1, speedRight);
        analogWrite(MOTOR2_PIN2, 0);
    }
    else {
        analogWrite(MOTOR2_PIN1, 0);
        analogWrite(MOTOR2_PIN2, -speedRight);
    }
}
}

```

Programul preia de pe portul serial Serial1 (portul serial ce face legătura între microprocesor și microcontroler) comenzile de înainte (i), dreapta (d), stânga (s), înapoi (b) și stop (x) și le transmite către motoare. Comenzile constau în modificarea (incrementarea sau decrementarea cu 10) celor două viteze (a motoarelor de pe partea dreapta și a motoarelor de pe partea stângă).

Înainte de a implementa cea de a doua componentă software a sistemului, componenta de legătură între conexiunea WiFi și comunicația serială dintre cele două circuite programabile ale plăcii de dezvoltare, trebuie configurată placa LinkIt Smart 7688 Duo să poată să acceseze Internet prin intermediul unei rețele WiFi locale – configurația inițială a plăcii – similară cu partea de configurare inițială.

Pentru configurarea de bază/inițială a plăcii de dezvoltare LinkIt Smart 7688 Duo se poate consulta următorul material: http://wiki.seeed.cc/LinkIt_Smart_7688_Duo/

Testarea configurației următoare a fost realizată pe o placă LinkIt Smart 7688 Duo cu firmware 0.9.4. Componenta software ce va rula pe microprocesorul sistemului sub OpenWRT este aplicația **ser2net**:

```

opkg update
opkg install ser2net

```

Pentru pornirea automată a aplicației la resetarea sistemului se va crea fișierul `/etc/init.d/ser2net` cu următorul conținut:

```

#!/bin/sh /etc/rc.common
START=10


```

```
STOP=15
start(){
ser2net
}
stop(){
killall ser2net
}
```

După care se va activa pornirea aplicației:

```
chmod +x /etc/init.d/ser2net
/etc/init.d/ser2net enable
/etc/init.d/ser2net start
```

Pentru a transmite comenzile către mașină vom utiliza o aplicație client telnet („[putty](#)” de exemplu) și se va conecta la adresa IP locală a plăcii de dezvoltare pe portul 2001. În consola telnet se vor trimite comenzile descrise anterior (caracterele i, s, d, b, x).

 192.168.1.1 - PuTTY

```
i
Viteza: 10/10
i
Viteza: 20/20
i
Viteza: 30/30
i
Viteza: 40/40
i
Viteza: 50/50
x
Viteza: 0/0
s
Viteza: 10/-10
s
Viteza: 20/-20
i
Viteza: 30/-10
i
Viteza: 40/0
x
Viteza: 0/0
█
```

5.3 WiFi car Reloaded - II⁵

În cazul în care una dintre comenzile prezentate nu este recunoscută sau camera video nu este recunoscută ca dispozitiv USB sau video se va verifica dacă pachetele software necesare sunt instalate:

```
opkg update
opkg install kmod-video-uvc kmod-video-core
opkg install usbutils
opkg install mjpg-streamer
```

Pentru transmisia live de imagine se va utiliza programul **mjpg-streamer** (instalat de ultima comandă din blocul anterior). Comanda de pornire a acestuia este:

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -y" -o "output_http.so"
```

Accesarea imaginilor transmise se poate face la adresele:

- flux video:
http://adresa_IP_sistem:8080/?action=stream;
- imagini:
http://adresa_IP_sistem:8080/?action=snapshot.

Pentru ca aplicația **mjpg-streamer** să pornească în mod automat trebuie ca în fișierul */etc/config/mjpg_streamer* să fie configurat parametrul **enabled** (și alți parametri dacă este cazul):

```
config mjpg-streamer 'core'
    option enabled '1'
    option input 'uvc'
    option output 'http'
    option device '/dev/video0'
#   option resolution '320x240'
    option yuv '1'
    option quality '80'
    option fps '15'
#   option led 'auto'
#   option www '/www/webcam'
    option port '8080'
#   option username 'openwrt'
#   option password 'openwrt'

/etc/init.d/mjpg-streamer enable
```

WebCam with the Linux UVC driver [OpenWrt Wiki]: <https://wiki.openwrt.org/doc/howto/webcam>

Pentru a scădea încărcarea sistemului OpenWRT se va activa doar comunicația între portul **2001** și portul serial */dev/ttyS0*.

În fișierul */etc/ser2net.conf* se vor comenta toate liniile de după linia ce definește comunicația utilizată:

```
2001:raw:600:/dev/ttyS0:9600 NONE 1STOPBIT 8DATABITS XONXOFF LOCAL -RTSCTS
#2002:raw:600:/dev/ttyS1:9600 NONE 1STOPBIT 8DATABITS XONXOFF LOCAL -RTSCTS
#2003:raw:5:/dev/ttyS2:9600
#2004:raw:5:/dev/ttyS3:115200
...
```

⁵ <https://blog.robofun.ro/2017/02/21/proiect-wifi-car-reloaded-partea-a-ii-a/>

6. INTERFAȚA de COMUNICARE cu PIC Microcontroler

Interfațarea senzorului de distanță cu ultrasunete HC-SR04 cu microcontroler PIC, respectiv cu PIC 18F8680 în cazul aplicației noastre. PIC Microcontroler este conectat cu VDD, VSS și GND la + 5V pentru alimentare. Procesorul cu frecvența de lucru de 8MHz este conectat la pinii OSC1 și OSC2 ai PIC. Conectorii de 22 [pF] sunt conectați împreună cu procesorul și au rolul de a stabiliza oscilațiile de frecvență. De asemenea se poate conecta un ecran LCD 16×2 la PORTD, care este interfațat cu ajutorul modului de comunicație pe 4 biți. Presetarea de 10 [KΩ] este utilizată pentru reglarea contrastului ecranului LCD. Un rezistor de 100 [Ω] este folosit pentru a limita curentul prin LED-ul de back-light.

Pinul TRIGGER-ului este conectat la RB0 (pinul 33) al PIC care trebuie să fie configurat ca un PIN de ieșire (bitul TRIS este 0) și pinul ECHO este conectat la RB4 (pinul 37) care trebuie să fie configurat ca un PIN de intrare (bitul TRIS este 1).

Programarea se face conform algoritmului de mai jos:

1. se activează modulul TRIGGER al HC-SR04;
2. se înregistrează ECHO;
3. se pornește cronometrul când este recepționat ECHO HIGH;
4. se oprește cronometrul când ECHO recepționează LOW;
5. se citește valoarea timer-ului;
6. se convertește semnalul în distanță;
7. se afișează valorile.

Modulul Timer1 se utilizează ca un contor sau cronometru pe 16 biți. Se compune din două registre de 8 biți TMR1H și TMR1L care pot fi citite și scrise. Perechea de înregistrare, TMR1H: TMR1L crește de la 0000H la FFFFH și crește până la 0000H. Dacă este activată întreruperea fluxului de timp, Timer1 este generat în timpul rulărilor de peste 0000H, abia din acest moment modulul va fi folosit ca un timer pe 16 biți.

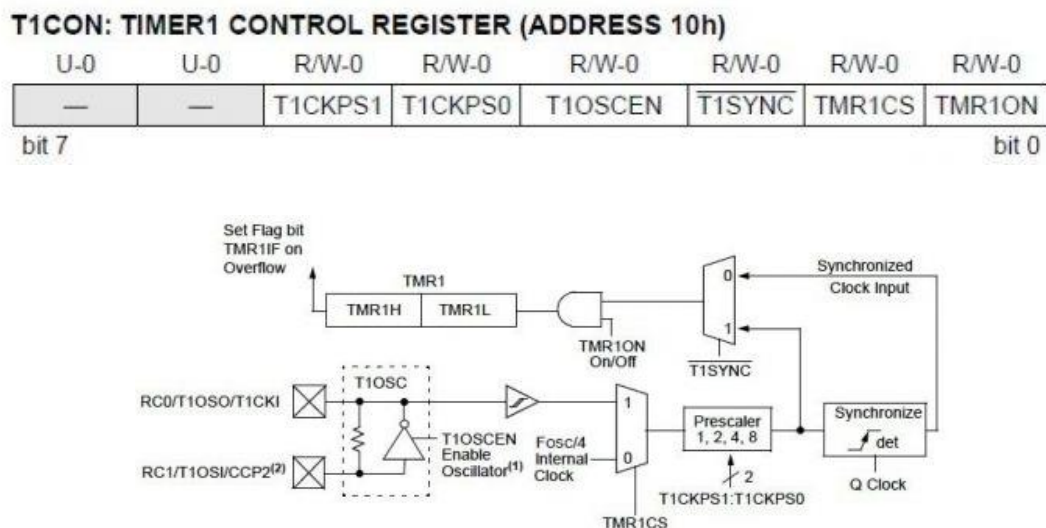


Fig. 6-1 Când T1OSCEN este ștersă, invertorul este oprit, fiind oprită alimentarea.

Deoarece se folosește modulul Timer1 ca un cronometru, va trebui să folosim ceasul intern ($F_{osc}/4$), adică $TMR1CS=0$, astfel încât se pot genera următoarele comenzi:

- scalarea pe două secvențe: $T1CKPS1=0$ și $T1CKPS0=1$;
- dacă $TMR1CS=0$, atunci $T1SYNC$ este ignorat;
- $T1OSEN=0$ are rolul de a dezactiva oscilatorul;
- bitul $TMR1ON$ permite utilizarea temporizatorului pe cele două poziții: ON sau OFF conform cerințelor utilizatorului;
- se poate inițializa temporizatorul ca: $T1CON=0x10$;

- pornirea cronometrului se face astfel: T1CON.F0=1 sau TMR1ON=1;
- oprirea cronometrului: T1CON.F0=0 sau TMR1ON=0;
- frecvența oscilatorului $F_{osc}=8\text{MHz}$, adică cea a cristalului.

$$\text{Time} = (\text{TMR1H}:\text{TMR1L}) \cdot (1/\text{Internal Clock}) \cdot \text{Prescaler}$$

$$\text{Internal Clock} = F_{osc}/4 = 8\text{MHz}/4 = 2\text{MHz}$$

$$\text{Time} = (\text{TMR1H}:\text{TMR1L}) \cdot 2/(2000000) = (\text{TMR1H}:\text{TMR1L})/1000000$$

Calculul distanței
- distanța = viteză * timp;
- d = distanța dintre senzorul cu ultrasunete și țintă;
- distanța totală parcursă de emisia cu ultrasunete = $2d$ (înainte și înapoi);
- viteza sunetului în aer: $340 \text{ [m/s]} \equiv 34000 \text{ [cm/s]}$;
- $d = (34000 * \text{Time}) / 2$, unde $\text{Time} = (\text{TMR1H}:\text{TMR1L}) / (1000000)$;
- $d = (\text{TMR1H}:\text{TMR1L}) / 58,82 \text{ cm}$;
- $\text{TMR1L}:\text{TMR1L} = \text{TMR1L} \mid (\text{TMR1H} \ll 8)$

Method – MikroC & MPLAB XC8 Program

MikroC Code

```
// LCD module connections
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;

sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections

void main()
{
    int a;
    char txt[7];
    Lcd_Init();
    Lcd_Cmd(_LCD_CLEAR);      // Clear display
    Lcd_Cmd(_LCD_CURSOR_OFF); // Cursor off

    TRISB = 0b00010000;      //RB4 as Input PIN (ECHO)

    Lcd_Out(1,1,"Developed By");
    Lcd_Out(2,1,"electroSome");

    Delay_ms(3000);
    Lcd_Cmd(_LCD_CLEAR);

    T1CON = 0x10;             //Initialize Timer Module

    while(1)
    {
        TMR1H = 0;           //Sets the Initial Value of Timer
```

```

TMR1L = 0;           //Sets the Initial Value of Timer

PORTB.F0 = 1;        //TRIGGER HIGH
Delay_us(10);        //10uS Delay
PORTB.F0 = 0;        //TRIGGER LOW

while(!PORTB.F4);     //Waiting for Echo
T1CON.F0 = 1;         //Timer Starts
while(PORTB.F4);      //Waiting for Echo goes LOW
T1CON.F0 = 0;         //Timer Stops

a = (TMR1L | (TMR1H<<8)); //Reads Timer Value
a = a/58.82;          //Converts Time to Distance
a = a + 1;            //Distance Calibration\
if(a>=2 && a<=400)    //Check whether the result is valid or not
{
    IntToStr(a,tst);
    Ltrim(tst);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1,1,"Distance = ");
    Lcd_Out(1,12,tst);
    Lcd_Out(1,15,"cm");
}
else
{
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1,1,"Out of Range");
}
Delay_ms(400);
}
}

```

MPLAB XC8 Code

```

#define _XTAL_FREQ 8000000

#define RS RD2
#define EN RD3
#define D4 RD4
#define D5 RD5
#define D6 RD6
#define D7 RD7

#include <xc.h>
#include "lcd.h";
#include <pic16f877a.h>

// BEGIN CONFIG
#pragma config FOSC = HS
#pragma config WDTE = OFF
#pragma config PWRTE = OFF
#pragma config BOREN = ON
#pragma config LVP = OFF
#pragma config CPD = OFF
#pragma config WRT = OFF
#pragma config CP = OFF
//END CONFIG

```

```
void main()
{
    int a;

    TRISB = 0b00010000;    //RB4 as Input PIN (ECHO)
    TRISD = 0x00;          // LCD Pins as Output

    Lcd_Init();

    Lcd_Set_Cursor(1,1);
    Lcd_Write_String("Developed By");
    Lcd_Set_Cursor(2,1);
    Lcd_Write_String("electroSome");

    __delay_ms(3000);
    Lcd_Clear();

    T1CON = 0x10;          //Initialize Timer Module

    while(1)
    {
        TMR1H = 0;          //Sets the Initial Value of Timer
        TMR1L = 0;          //Sets the Initial Value of Timer

        RB0 = 1;            //TRIGGER HIGH
        __delay_us(10);     //10uS Delay
        RB0 = 0;            //TRIGGER LOW

        while(!RB4);        //Waiting for Echo
        TMR1ON = 1;          //Timer Starts
        while(RB4);         //Waiting for Echo goes LOW
        TMR1ON = 0;          //Timer Stops

        a = (TMR1L | (TMR1H<<8)); //Reads Timer Value
        a = a/58.82;         //Converts Time to Distance
        a = a + 1;          //Distance Calibration
        if(a>=2 && a<=400)   //Check whether the result is valid or not
        {
            Lcd_Clear();
            Lcd_Set_Cursor(1,1);
            Lcd_Write_String("Distance = ");

            Lcd_Set_Cursor(1,14);
            Lcd_Write_Char(a%10 + 48);

            a = a/10;
            Lcd_Set_Cursor(1,13);
            Lcd_Write_Char(a%10 + 48);

            a = a/10;
            Lcd_Set_Cursor(1,12);
            Lcd_Write_Char(a%10 + 48);

            Lcd_Set_Cursor(1,15);
            Lcd_Write_String("cm");
        }
        else
```

```
{
  Lcd_Clear();
  Lcd_Set_Cursor(1,1);
  Lcd_Write_String("Out of Range");
}
__delay_ms(400);
}
}
```

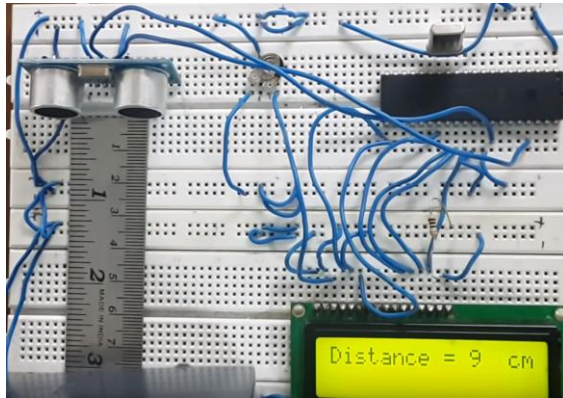


Fig. 6-2 Schema fizică de măsurare a distanței cu HC-SR04.