UNIVERSITATEA POLITEHNICA
BUCUREȘTI

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

Proiect 2 – Electronică Aplicată

SISTEM DE OCOLIRE A OBSTACOLELOR

Coordonator: Mihai MUNTEAN
Titular disciplină: Ana NEACȘU

Studenți:
Mihnea-Andrei CIUNGU
Cătălin-Ionel STAN
Oana-Andreea TECULESCU

An universitar: 2023–2024

Cuprins

1. Introducere

Proiectul *„Sistem de ocolire a obstacolelor”* are ca scop crearea unui dispozitiv capabil să se deplaseze într-un spațiu cu obstacole, evitând coliziunea cu acestea cât mai eficient posibil. Cu ajutorul senzorilor ultrasonici, sistemul va măsura distanța până la obstacole și va lua decizii pentru a modifica direcția de mișcare a robotului. Sistemul are o viteză constantă prestabilită, iar când se detectează un obstacol, microcontroller-ul va ajusta viteza unuia dintre motoare pentru a schimba direcția de deplasare. Dacă toți senzorii detectează un obstacol, sistemul va efectua o întoarcere la 180° pentru a evita coliziunea. Obiectivul principal este realizarea unui prototip funcțional, capabil să navigheze autonom în medii necunoscute.

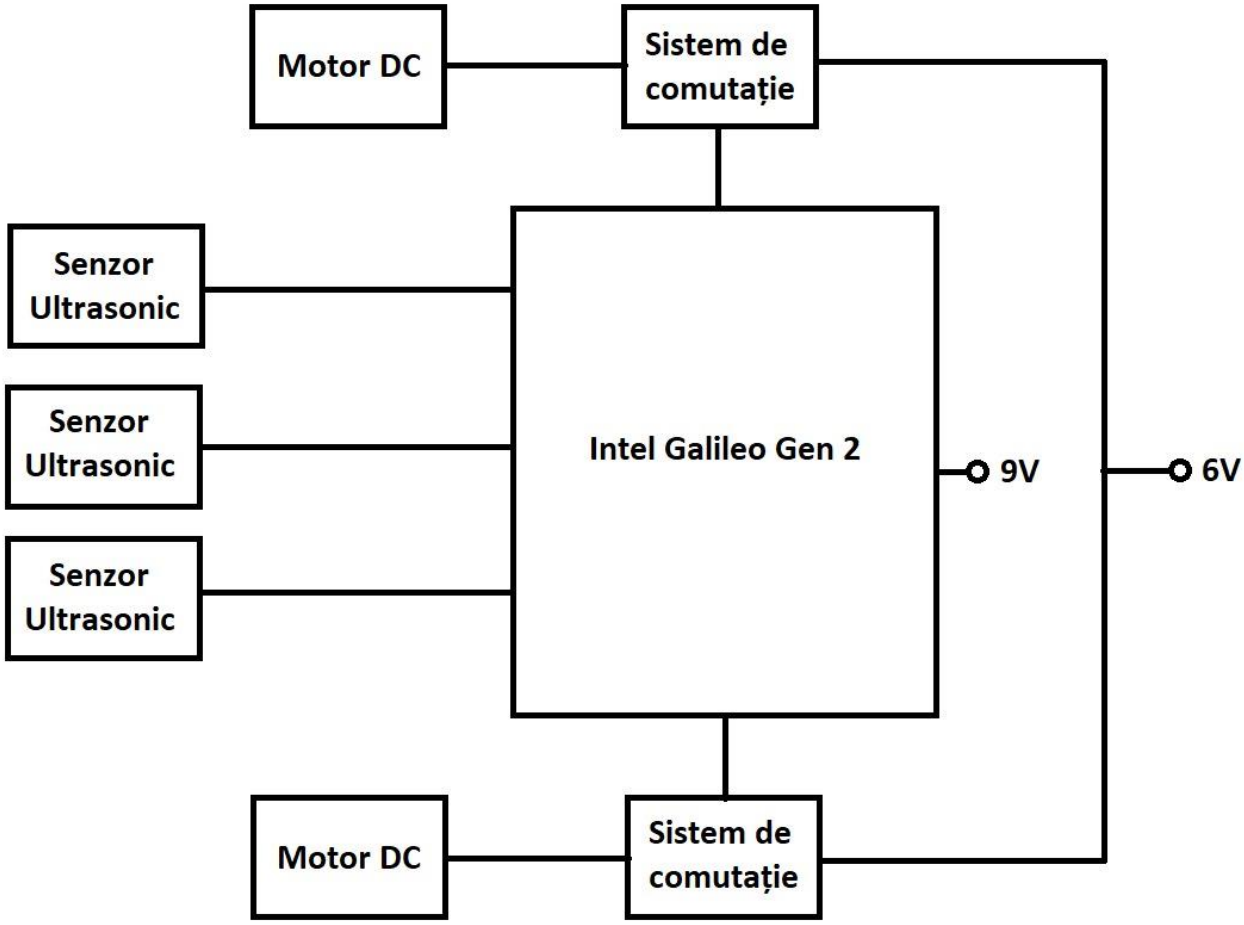


Fig.1 1 Diagrama bloc

2. Resurse hardware

Pentru realizarea sistemului de ocolire a obstacolelor au fost necesare următoarele componente hardware:

2.1. Senzorii ultrasonici HC-SR04

Principalele părți componente ale senzorilor ultrasonici HC-SR04 sunt: transmițătorul ultrasonic (Trigger) care are rolul de a emite unde sonore de înaltă frecvență (40kHz) și receptorul ultrasonic (Echo) care primește ecoul undelor sonore reflectate de obstacole. [1]

2.2. Șasiu cu motoare DC

Șasiul este structura fizică de bază a robotului, care include cadrul și motoarele de curent continuu (DC). Motoarele DC sunt utilizate pentru a asigura mișcarea robotului prin convertirea energiei electrice în lucru mecanic și vor fi controlate de microcontroller pentru a permite schimbarea direcției de deplasare, în funcție de datele furnizate de senzori.

2.3. Tranzistoare bipolare NPN 2n2222

Ne vom folosi de funcția de comutare a tranzistorului bipolar, pentru a putea gestiona alimentarea motoarelor DC, și implicit direcția de deplasare a robotului. Astfel cu ajutorul microcontroller-ului vom genera un semnal de tip PWM cu factor de umplere diferit, pe care îl vom aplica la baza tranzistorului bipolar. Cu cât semnalul aplicat la baza tranzistorului va avea un factor de umplere mai mare, cu atât va trece mai mult curent de la emitor spre colector și deci spre motor.

2.4. Diode 1N4001

Atunci când motoarele DC se opresc brusc sau sunt deconectate de la sursa de alimentare, în cazul de față un pachet de baterii de 6V, datorită inductanței lor acestea pot genera un vârf de tensiune inversă suficient de mare pentru a deteriora tranzistorul, senzorii sau alte componente ale circuitului.

Așadar, dioda este plasată în paralel cu motorul, pentru a preveni distrugerea componentelor, oferind curentului generat de motor un traseu care să permită disiparea energiei electrice fără a afecta negativ tranzistorul sau alte componente.

2.5. Microcontroller Intel Galileo Gen 2

Intel Galileo Gen 2 este o platformă de dezvoltare bazată pe arhitectura Intel Quark SoC X1000, compatibilă cu ecosistemul Arduino. În cadrul proiectului *„Sistem de ocolire a obstacolelor”*, acest microcontroller joacă un rol central în gestionarea senzorilor, prelucrarea datelor și controlul motoarelor pentru a asigura navigarea autonomă a robotului. [2]

Acesta este responsabil pentru citirea semnalelor de la senzorii ultrasonici HC-SR04. Acești senzori măsoară distanța până la obstacole și trimit aceste informații către microcontroller. Pe baza datelor primite de la senzori, Intel Galileo Gen 2 ia decizii în timp real privind direcția de deplasare a robotului, ajustând individual, viteza de rotație a motoarelor.

2.6. Breadboard

Un breadboard este o placă de circuit fără lipire, utilizată pentru prototiparea circuitelor electronice. Permite conexiuni rapide și ușoare între componente fără a necesita lipire permanentă.

2.7. Baterie de 9V

Pentru a alimenta microcontroller-ul am consultat documentația acestuia și am aflat că este necesară o sursă de alimentare care să furnizeze între 7V și 15V, astfel am optat pentru o baterie de 9V pe care o conectăm folosind o terminație cu Jack de 2.1 mm. [2]

2.8. Baterii AA

Pentru alimentarea motoarelor folosim 4 baterii AA de 1.5V, conectate în serie, totalizând 6V.

2.9. Condensatoare de 100n

Folosite pentru a reduce interferentele pe care le pot injecta motoarele in circuit.

3. Resurse software

3.1. Arduino IDE

Arduino Integrated Development Environment (IDE) este un software open-source utilizat pentru scrierea și încărcarea de cod pe plăcile de dezvoltare Arduino. În cadrul proiectului *„Sistem de ocolire a obstacolelor”*, Arduino IDE este un instrument esențial pentru dezvoltarea, testarea și implementarea codului necesar pentru controlul senzorilor, prelucrarea datelor și gestionarea motoarelor.

IDE-ul oferă funcționalitatea de încărcare a codului pe placa de dezvoltare printr-un singur click. Utilizatorii pot compila codul și apoi încărca fișierul binar rezultat direct pe microcontroller.
Procesul de încărcare este simplificat, iar mesajele de eroare sau succes sunt afișate în consola IDE.

Arduino IDE include un monitor serial integrat care permite utilizatorilor să comunice cu placa Arduino în timp real. Acest instrument este util pentru depanarea și monitorizarea datelor transmise și recepționate de la placa de dezvoltare. Utilizatorii pot trimite și primi date prin portul serial, facilitând testarea și ajustarea codului. [3]

Una din funcțiile utile pe care le-am utilizat în implementarea codului este funcția *„pulseIn()”* care este o funcție inclusă în librăria standard de la Arduino, numită *„Arduino.h”*. Biblioteca *„Arduino.h”* este o bibliotecă fundamentală care este inclusă în mod implicit în toate proiectele, și conține funcții de bază pentru controlul intrărilor și ieșirilor, manipularea timpului și alte operațiuni esențiale pentru programarea microcontroller-ului.

Funcția *„pulseIn()”* joacă un rol important în măsurarea duratei unui puls trimis de senzorul ultrasonic. Această funcție este utilizată pentru a citi durata unui puls HIGH sau LOW pe un anumit pin. În contextul proiectului nostru, *„pulseIn()”* este utilizată pentru a măsura timpul în care semnalul ultrasonic, trimis de senzor, călătorește până la obstacol și înapoi.

Funcția primește doi parametri: pinul de pe care să citească și starea (HIGH sau LOW) pe care să o măsoare. În codul nostru, funcția *„pulseIn(echoPin, HIGH)”* returnează durata, în microsecunde, în care pinul *„echoPin”* a fost în stare HIGH. Această durată reprezintă timpul necesar pentru ca semnalul ultrasonic să se reflecte de la un obstacol și să se întoarcă la senzor.

Astfel, *„pulseIn()”* permite calcularea distanței până la obstacol prin măsurarea timpului de propagare a undei sonore și aplicarea formulei prezentate la finalul capitolului 4. [4]

4. Implementare hardware

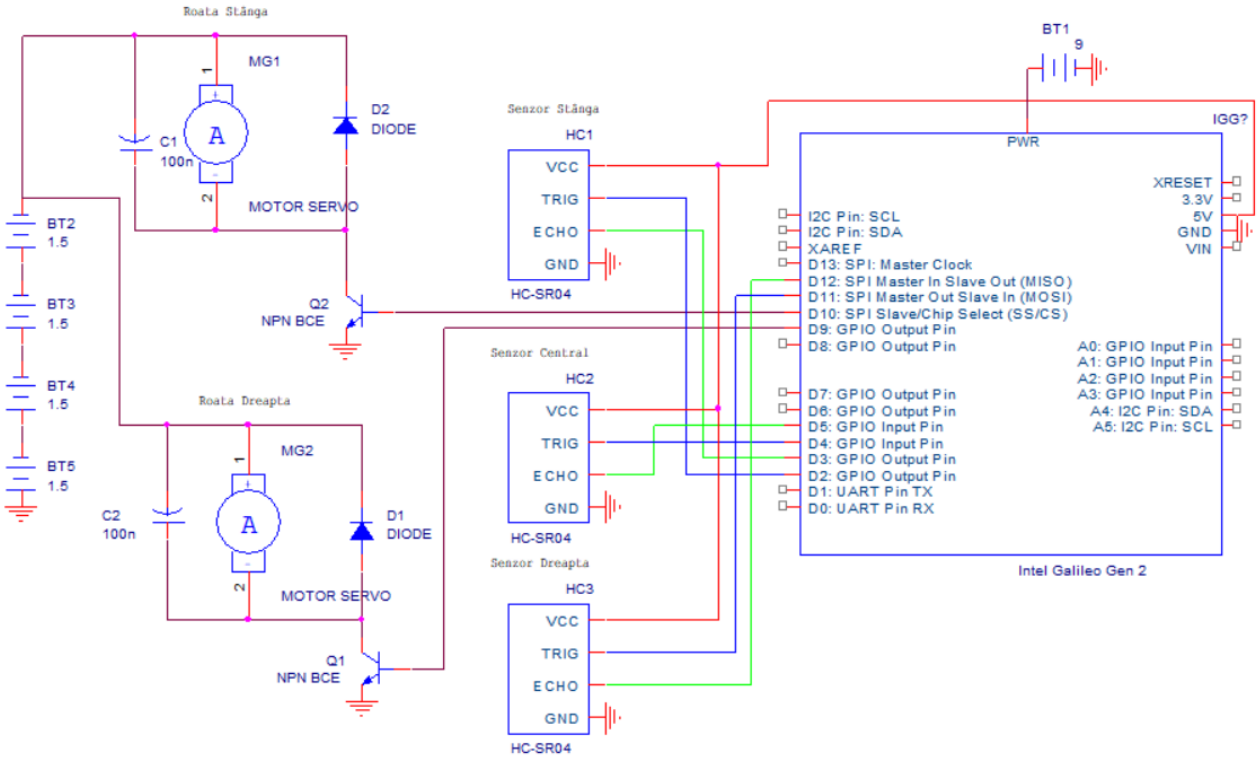


Fig. 4. 1 Schema electrică a montajului în OrCAD

4.1. Principiul de funcționare al motoarelor DC

Pentru implementarea sistemului de ocolire a obstacolelor au fost utilizate două motoare DC, funcționarea fiecăruia fiind asigurată de circuitul cu următoarea schema:

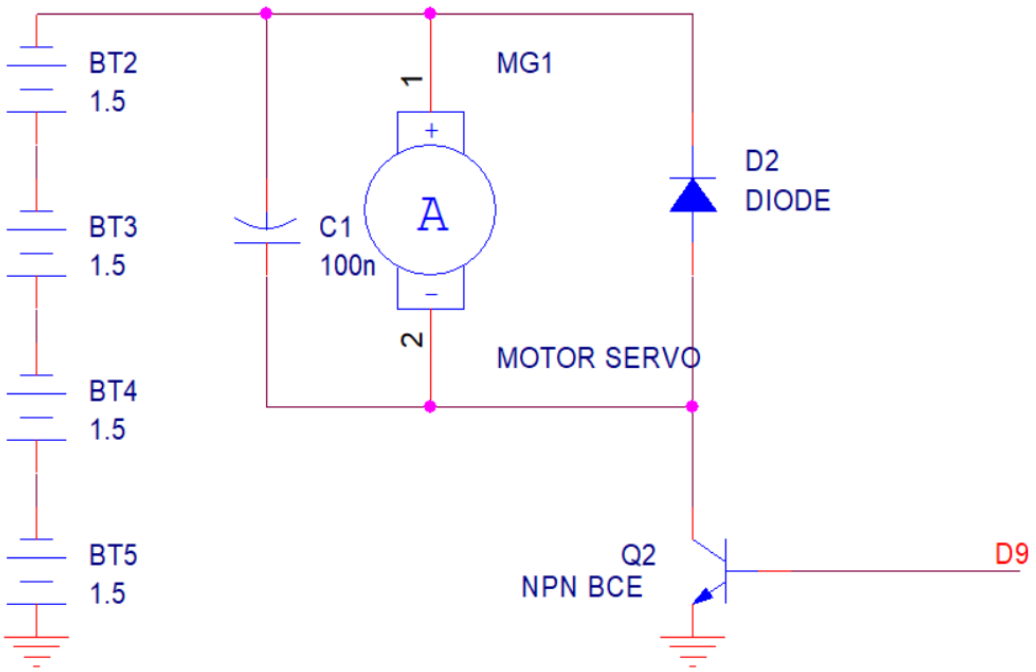


Fig. 4. 2 Schema electrică a motoarelor DC

După cum se poate observa în figura 4.2, circuitul este alcătuit din: patru baterii grupate în serie, de câte 1.5V fiecare (împreună alcătuiesc o sursă de tensiune de 6V), un motor DC, un condensator, o diodă (ambele fiind conectate în paralel cu motorul), un tranzistor bipolar NPN (baza conectată la pinul D9 al plăcuței Intel Galileo Gen 2, emitorul conectat la masă și colectorul conectat la gruparea paralel motor-condensator-diodă). Funcționarea circuitului se bazează pe funcția de comutare a tranzistorului. [5]

Microprocesorul este programat să genereze pe pinul digital D9, setat ca ieșire, un semnal dreptunghiular cu un anumit factor de umplere, având o valoare HIGH de 5V și o valoare LOW de 0V.

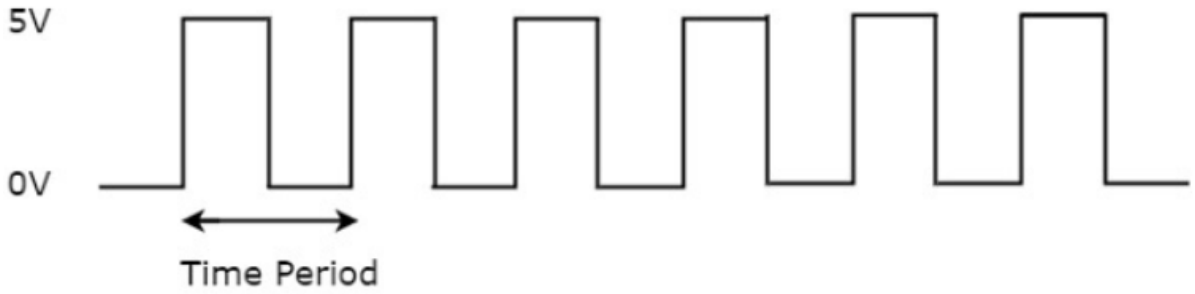


Fig. 4. 3 Semnal dreptunghiular cu amplitudinea între 0V și 5V

În momentul aplicării valorii LOW de 0V, tranzistorul intră în regiunea de blocarea, devenind echivalent cu un întrerupător deschis. Circuitul echivalent astfel obținut va funcționa în gol, deci motoarele nu vor fi alimentate.

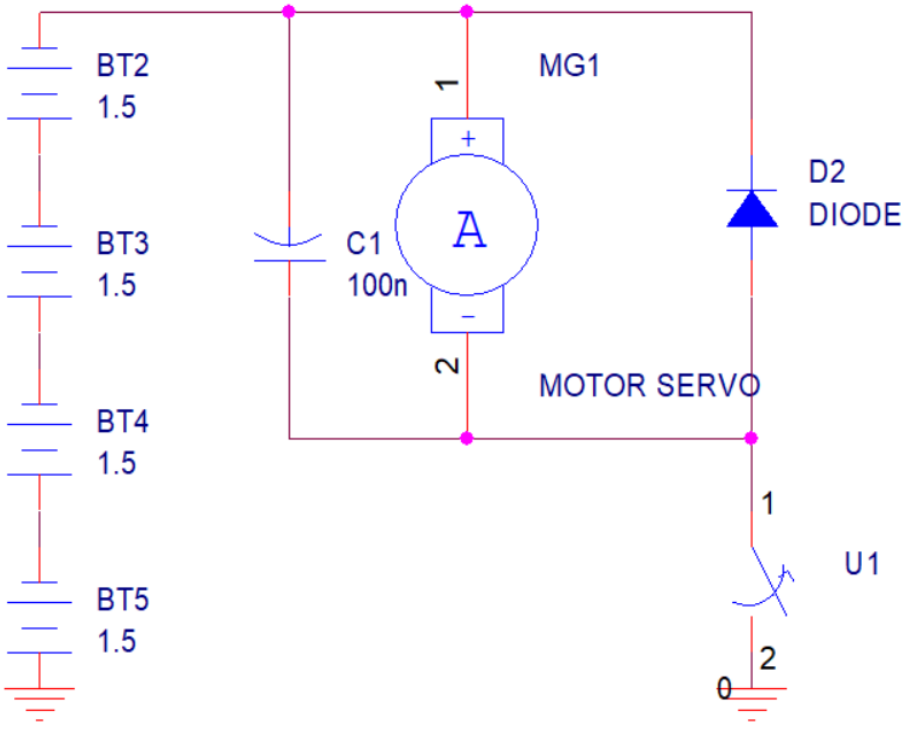


Fig. 4. 4 Schema echivalentă, întrerupător deschis

În momentul aplicării valorii HIGH de 5V, tranzistorul intră în regiunea de saturație, devenind echivalent cu un întrerupător închis. Circuitul echivalent astfel obținut va avea următoarea schemă electrică:

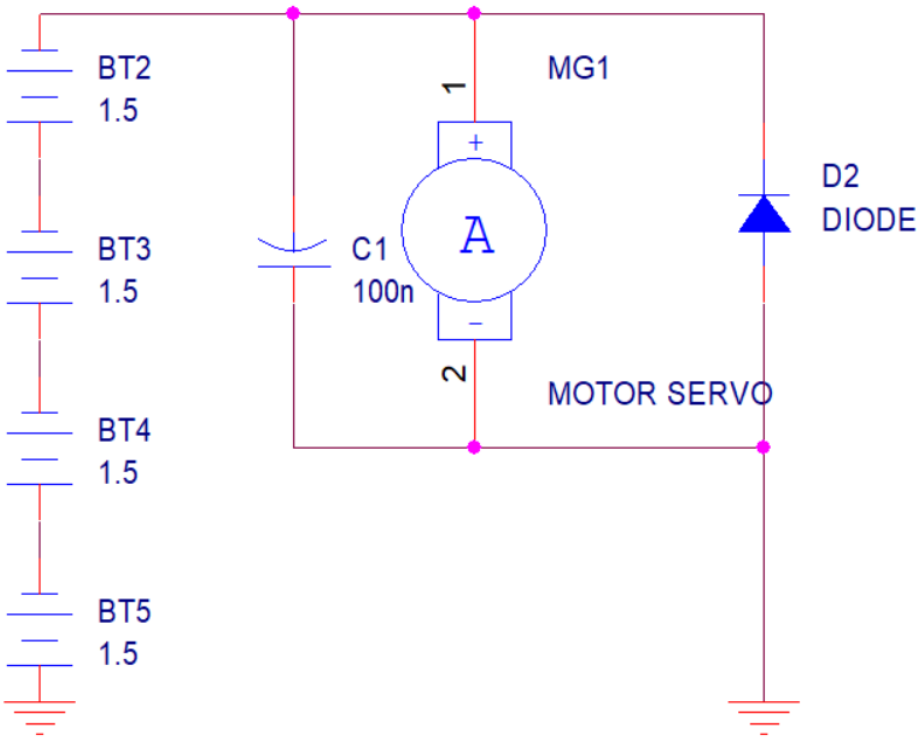


Fig. 4. 5 Schema echivalentă, întrerupător închis

Viteza motoarelor este direct proporțională cu factorul de umplere al semnalului dreptunghiular aplicat în baza tranzistorului. Altfel spus, cu cât valoarea HIGH ocupă cea mai mare parte dintr-o perioadă, cu atât viteza motoarelor va fi mai mare. Pentru o valoare a factorului de umplere de 100%, viteza roților va fi maximă, iar pentru o valoare a factorului de umplere apropiată de 0%, motoarele nu vor funcționa.

Dioda are rolul de a bloca posibilii curenți generați de motor, iar condensatorul de a ajuta la filtrarea zgomotului electric. [5]

4.2. Principiul de funcționare al senzorilor

Pentru implementarea sistemului de detecție a obstacolelor au fost utilizați trei senzori, care prezintă patru terminale: VCC, Trig, Echo, GND.

- VCC este conectat la pinul 5V al placuței Intel Galileo Gen 2, a cărui rol este de a asigura o tensiune continuă, constantă.
- Trig este conectat la unul din pinii digitali, setați ca ieșire (modul output).
- Echo este conectat, în mod similar, la un alt pin digital, setat însă ca intrare (modul input).
- GND este conectat la masa ansamblului electric.

Principiul de funcționare al senzorilor ultrasonici se bazează pe transmiterea unui puls de 10 microsecunde pe pinul TRIG. Acest puls declanșează emiterea unui pachet de 8 cicluri de unde ultrasonice la 40kHz. Transmițătorul ultrasonic emite undele sonore care se propagă în aer până la întâlnirea unui obstacol, moment în care acestea se reflectă înapoi către senzor. Receptorul primește undele reflectate și trimite un semnal de înaltă tensiune (5V) pe pinul ECHO.

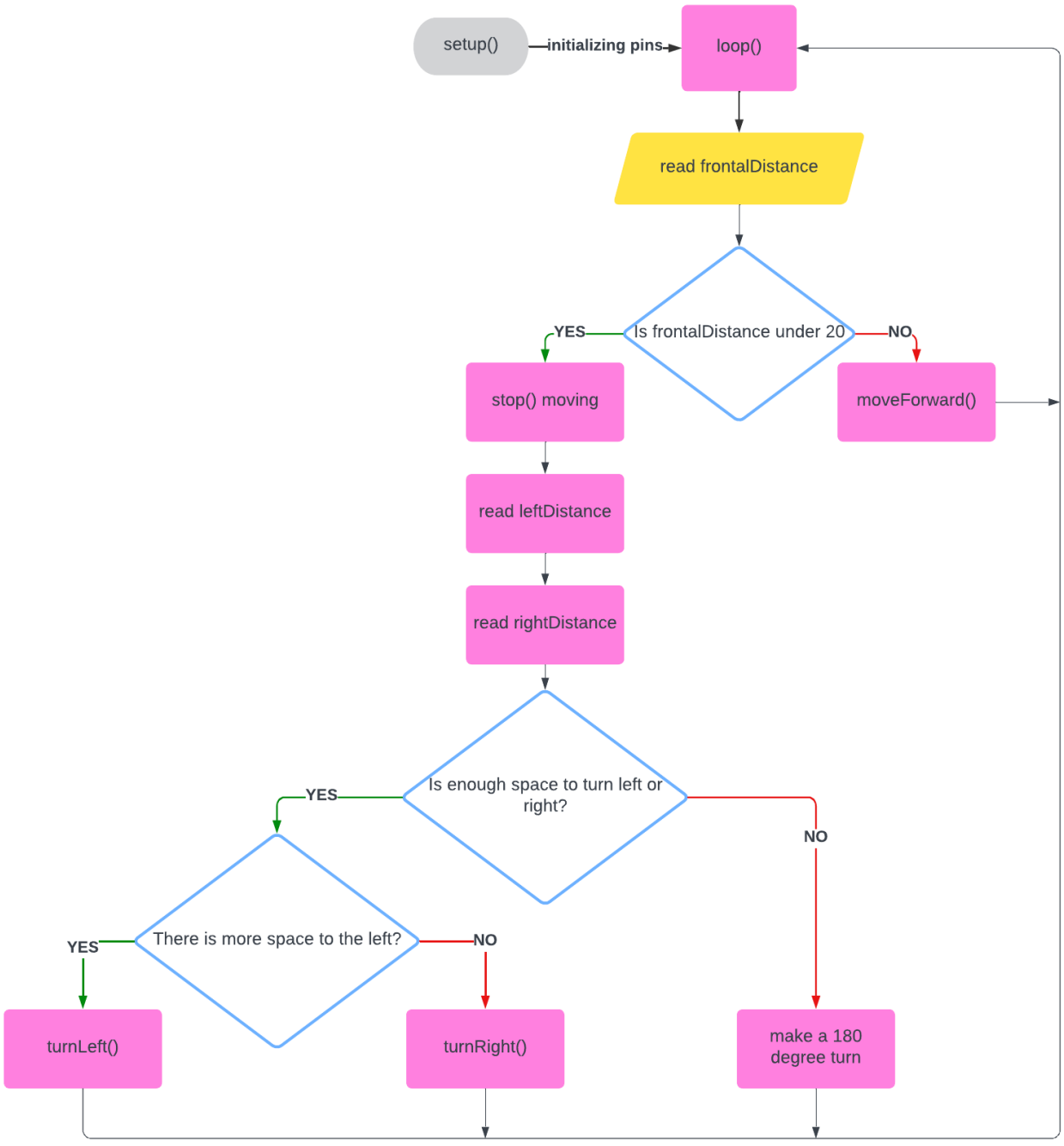
Microcontroller-ul conectat la senzor măsoară durata semnalului de pe pinul ECHO, adică durata de timp dintre emiterea undelor și recepționarea ecoului.

Distanța se calculează pe baza formulei de mai jos [6]:

$$Distanța\ (cm) = \frac{Timpul\ de\ călătorie\ (\mu s) \times Viteza\ sunetului\ (\frac{m}{s})}{2}$$

5. Implementare software

5.1. Organigrama codului



5.2. Codul Arduino C++ încărcat pe microprocesorul Intel Galileo Gen 2

```
// Initializing Pins
// Sensor Pins
#define triggerPinLeft 2
#define echoPinLeft 3
#define triggerPinCentral 4 // This is the pin that will output a high frequency signal for the central sensor
#define echoPinCentral 5 // This is the pin that will read the reflected signal for the central sensor
#define triggerPinRight 11
#define echoPinRight 12

// Motors Pins
// These pins are connected to the base terminals of BJT's to control the flow of current that reaches the motors
#define leftMotor 10 // We are using pin number 10 to control the left motor
#define rightMotor 9 // ... and pin number 9 to control the right motor

// Motor Speed
#define motorSpeed 150

void setup() {
  Serial.begin (9600); // Setting the BaudRate for the serial monitor
  // Specifying how each pin will be used
  pinMode(triggerPinLeft, OUTPUT);
  pinMode(echoPinLeft, INPUT);
  pinMode(triggerPinCentral, OUTPUT);
  pinMode(echoPinCentral, INPUT);
  pinMode(triggerPinRight, OUTPUT);
  pinMode(echoPinRight, INPUT);
  pinMode(leftMotor, OUTPUT);
  pinMode(rightMotor, OUTPUT);
}

void loop() {
  float frontalDistance = getDistance(triggerPinCentral, echoPinCentral); // check the distance in front
  if (frontalDistance < 20) { // If the frontal distance to an obstacle is under 20 cm, than the car will stop and check the surroundings
    stop();
    delay(2000);
    float leftDistance = getDistance(triggerPinLeft, echoPinLeft); // check the distance to the left
    float rightDistance = getDistance(triggerPinRight, echoPinRight); // check the distance to the right
    if (leftDistance >= 20 || rightDistance >= 20) { // If there are enough space in the left or right side, the car will turn in that direction
      if (leftDistance > rightDistance) { // Compare the distances and make a decision
        turnLeft();
      } else {
        turnRight();
      }
    } else {
      turnAround(); // Make a 180 degrees turn, when it's not enough space to the right or left side
    }
  } else {
    moveForward(); // When there are no obstacles, the car will go forward
  }
```

```
 50. }
 51.
 52. float getDistance(int triggerPin, int echoPin) {
 53.   float duration, distance;
 54.   digitalWrite(triggerPin, LOW);
 55.   delayMicroseconds(2);
 56.
 57.   digitalWrite(triggerPin, HIGH);
 58.   delayMicroseconds(10);
 59.   digitalWrite(triggerPin, LOW);
 60.
 61.   duration = pulseIn(echoPin, HIGH);
 62.   distance = (duration / 2) * 0.0344; //distance = (variable time/2) x
speed of sound
 63.   return distance;
 64. }
 65.
 66. void printDistance() {
 67.   // A function used in the debugging stage to check the accuracy of the
sensors, and the correctness of the decisions made
 68.   Serial.print("LEFT: ");
 69.   printDistance(getDistance(triggerPinLeft, echoPinLeft));
 70.   Serial.print("CENTER: ");
 71.   printDistance(getDistance(triggerPinCentral, echoPinCentral));
 72.   Serial.print("RIGHT: ");
 73.   printDistance(getDistance(triggerPinRight, echoPinRight));
 74.   Serial.println("");
 75. }
 76.
 77. void moveForward() {
 78.   analogWrite(leftMotor, motorSpeed);
 79.   analogWrite(rightMotor, motorSpeed);
 80. }
 81.
 82. void turnLeft() {
 83.   analogWrite(leftMotor, 0);
 84.   analogWrite(rightMotor, motorSpeed);
 85.   delay(300);
 86. }
 87.
 88. void turnRight() {
 89.   analogWrite(rightMotor, 0);
 90.   analogWrite(leftMotor, motorSpeed);
 91.   delay(300);
 92. }
 93.
 94. void turnAround() {
 95.   analogWrite(leftMotor, motorSpeed);
 96.   analogWrite(rightMotor, 0);
 97.   delay(1000);
 98. }
 99.
100. void stop() {
101.   analogWrite(leftMotor, 0);
102.   analogWrite(rightMotor, 0);
103. }
104.
```

5.3. Explicațiile codului

Pentru început am definit pinii la care sunt conectați senzorii (pinii Echo și Trigger ai acestora), și bazele tranzistorilor care prin funcția lor de comutare vor controla alimentarea celor două motoare DC. De asemenea, am definit și valoarea factorului de umplere cu denumirea *"motorSpeed"*, care va avea efect asupra nivelului de tensiune care ajunge la motoare și implicit a vitezei de rotație a acestora.

În continuare, în blocul „*setup()"* am inițializat pinii precizând modul acestora de funcționare (INPUT, respectiv OUTPUT) și am stabilit rata de transfer "Baud-Rate" pentru monitorul serial, care ne va ajuta în procesul de debugging, să monitorizăm funcționarea corectă a senzorilor.

Blocul *„loop()"* ce va rula la infinit descrie modul de funcționare al robotului, apelând funcții declarate global, precum *„getDistance()"*, *„moveForward()"*, *„turnLeft()"*, *„turnRight()"*, *„turnAround()"*, sau „*stop()"* . În interiorul blocului se aplează funcția *„getDistance()"* care citește distanța față de obstacolul din față și o salvează în variabila „*frontalDistance"*. Conisiderând 20 cm o distanță suficient de mare pentru a vira cu succes, comparăm valoarea variabilei „*frontalDistance"* cu 20.

În cazul în care expresia este evaluată ca fiind adevărată, înseamnă că nu mai putem înainta pentru că ar avea loc o coliziune, așadar apelăm funcția „*stop()"* pentru a efectua operația de oprire, iar după 2 secunde, un timp suficient de mare pentru a evita interferența cu undele emise anterior de senzorul central, citim distanța față de un potențial obstacol situat în stânga sau dreapta robotului, apelând din nou funcția „*getDistance()"* cu parametrii corespunzători pinilor trigger și echo ai senzorului vizat. Verficăm dacă există suficient spațiu în stânga sau în dreapta, astfel încât să aibă sens să virăm în acea direcție, în caz contrar efectuăm operația de întoarcere apelând funcția „*turnAround()"*. Totuși, dacă există suficient spațiu să virăm, verificăm care din cele două distanțe este mai mare și virăm în acea direcție apelând una din cele 2 funcții: „*turnLeft()"* sau „*turnRight()"*.

Bucla se repetă la infinit astfel încât se va evalua din nou expresia ce are rolul de a verifica distanța frontală. De această dată, cel mai probabil variabila „*frontalDistance"* va avea o valoare mai mare de 20, ceea ce înseamnă că robotul va merge înainte, manevră efectuată prin apelarea funcției „*moveForward()"*.

Funcțiile „*turnLeft()"*, „*turnRight()"*, *„moveForward()"*, *„turnAround()"*, sau „*stop()"* apelate în interiorul blocului *„loop()"* au la baza implementării lor, controlul factorului de umplere al semnalului de tip PWM pentru fiecare motor în parte. Spre exemplu, când vrem să virăm la stânga, în baza tranzistorului corespunzător motorului din stânga vom genera un semnal cu factor de umplere 0, pentru ca la motor să ajungă 0V, iar pentru tranzistorul corespunzător motorului din dreapta vom genera un semnal cu factor de umplere mare pentru a furniza motorului DC o tensiune suficientă funcționării acestuia. Vom controla rotația motoarelor în mod similar și pentru celelalte funcții implementate.

De asemenea, funcția *„getDistance()"*, generează un semnal dreptunghiular, al cărui palier de 1 este de 10 microsecunde și îl transmite pe pinul trigger al senzorului. Cu ajutorul pinului echo și a funcției „*pulseIn()"* vom putea determina durata de timp care a fost necesară pentru ca undele de înaltă frecvență transmise să se întoarcă înapoi la senzor și astfel vom calcula distanța cu ajutorul formulei din capitolul 4.

Funcția getDistance() va avea ca parametrii de intrare valorile pinilor senzorului vizat.

6. Concluzii

Așadar, utilizând doar trei senzori și două roți, antrenate de două motoare DC se poate realiza un sistem de bază pentru ocolirea obstacolelor. Senzorii prelevează informații din mediu despre posibile obstacole, mai precis distanța până la acestea, iar motoarele realizează schimbarea direcției. Prin încetinirea unei anumite roți, sistemul își schimbă direcția de deplasare. Astfel, este capabil să efectueze viraje în mod optim.

Una din importantele probleme apărute în implementarea proiectului a fost reprezentată de controlul motoarelor cu ajutorul microcontroller-ului. Pinii digitali sunt capabili să genereze o tensiune de 5V suficientă pentru a alimenta motoarele, însă nu pot genera un curent de 400 mA necesar pentru alimentarea directă a motoarelor DC. Am rezolvat problema implementând un circuit care să realizeze comutația între motoare și o sursă externă de tensiune conectată la acestea. Comutația este controlată de unul din pinii digitali ai microcontroller-ului.

Proiectul poate fi îmbunătățit adăugându-i și funcția de a se deplasa înapoi. Acest lucru devine posibil înlocuind circuitul format din tranzistorul bipolar și diodă cu o punte H. Aceasta este disponibilă atât sub formă de circuit integrat, dar poate fi realizată în mod facil și din componente discrete: patru tranzistori bipolari.

7. Bibliografie

[1] Ghid pentru senzor ultrasonic HC-SR04, https://sites.google.com/site/arduinoelectronicasiprogramare/arduino-si-senzori/1

[2] Documentație Intel Galileo Gen 2, https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

[3] Arduino Integrated Development Environment (IDE), https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/

[4] Funcția pulseIn0, https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

[5] How To Drive A DC Motor Without A Motor Driver Module, https://techexplorations.com/guides/arduino/motors/dc-motor-with-transistor/

[6] Getting Started with the HC-SR04 Ultrasonic sensor, https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1