

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Calculatoare

LUCRARE DE DIPLOMĂ

Coordonator științific:
conf.dr.ing.PANTILIMONESCU
Florin-Gheorghe

Absolvent:
Andrei-Ionuț SĂLĂVĂSTRU

Iași, 2024

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Calculatoare

Sistem mobil detectie traseu și evitare obstacole

LUCRARE DE DIPLOMĂ

Coordonator științific:
conf.dr.ing.PANTILIMONESCU
Florin-Gheorghe

Absolvent:
Andrei-Ionuț SĂLĂVĂSTRU

Iași, 2024

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII PROIECTULUI DE DIPLOMĂ

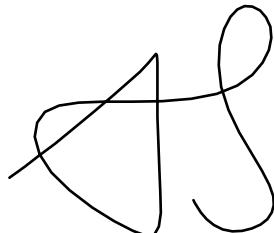
Subsemnatul SĂLĂVĂSTRU ANDREI-IONUȚ,
legitimat cu C.I. seria VS nr. 904109, CNP 5010114226790
autorul lucrării SISTEM MOBIL DETECȚIE TRASEU ȘI EVITARE OBSTACOLE

elaborată în vederea susținerii examenului de finalizare a studiilor de licență, programul de studii CALCULATOARE organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iași, sesiunea IULIE a anului universitar 2024, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iași (Manualul Procedurilor, UTI.POM.02 - Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data

01.07.2024

Semnătura



Cuprins

Introducere	1
0.1 Motivația alegerii temei	1
0.2 Relevanța și contextul temei alese	1
0.3 Obiectivele generale ale lucrării	1
0.4 Metodologia și instrumentele utilizate	1
0.5 Structura lucrării	2
1 Fundamentarea teoretică și documentarea bibliografică	3
1.1 Domeniul și contextul abordării temei	3
1.1.1 Evoluția tehnologică în domeniul roboticii	3
1.1.1.1 Primele etape ale dezvoltării roboticii	3
1.1.1.2 Introducerea microprocesoarelor și senzorilor avansați	4
1.1.1.3 Viitorul roboticii	5
1.1.2 Importanța automatizării în industrie	5
1.1.2.1 Creșterea productivității	5
1.1.2.2 Îmbunătățirea calității produselor	5
1.1.2.3 Reducerea costurilor de producție	6
1.1.2.4 Îmbunătățirea siguranței la locul de muncă	6
1.1.2.5 Flexibilitate și adaptabilitate	6
1.1.2.6 Crearea de locuri de muncă calificate	6
1.1.3 Aplicații practice ale robotilor autonomi	6
1.1.4 Beneficiile dezvoltării unui robot autonom	7
1.2 Tema propusă	7
1.3 Prezentare succintă și comparativă privind realizările actuale pe aceeași temă	8
1.4 Analiza tipurilor de produse/aplicații existente din respectiva categorie a temei, tehnologii folosite pentru implementare	9
1.4.1 Tipuri de Produse/Aplicații Existente	9
1.4.2 Tehnologii Folosite pentru Implementare	10
1.5 Elaborarea specificațiilor privind caracteristicile așteptate de la aplicație	10
2 Proiectarea hardware	13
2.1 Specificații tehnice	13
2.1.1 Aplicații tipice ale Arduino Uno R3	15
2.2 Modulele generale ale aplicației și interacțiunile dintre ele	15
2.2.1 Modulul de control al motoarelor	15
2.2.1.1 Specificații tehnice	15
2.2.1.2 Rolul în sistem	16
2.2.1.3 Interfațarea cu Arduino Uno R3	16
2.2.2 Modulul de senzori infraroșu	17
2.2.2.1 Specificații tehnice	17

2.2.2.2	Rolul în sistem	18
2.2.3	Modulul senzorului ultrasonic	18
2.2.3.1	Specificații tehnice	18
2.2.4	Modulul Bluetooth	20
2.2.4.1	Specificații tehnice	20
2.2.4.2	Rolul în sistem	21
2.2.5	Modulul de servomecanisme	21
2.2.5.1	Specificații tehnice	22
2.2.5.2	Rolul în sistem	22
2.2.6	Motoarele electrice DC TT Motor 200 RPM	23
2.2.6.1	Specificații tehnice	23
2.2.6.2	Rolul în sistem	23
2.3	Avantaje și dezavantaje ale metodei alese	24
2.4	Limitele metodei alese	24
2.4.1	Condiții de Iluminare	24
2.4.2	Caracteristicile suprafetei	25
2.4.3	Distanță și tipul obstacolelor	25
2.4.4	Raza de acțiune a modulului Bluetooth	25
2.4.5	Capacitatea de procesare a Arduino Uno	25
2.4.6	Calitatea componentelor utilizate	25
2.5	Conecțarea componentelor robotului	25
2.5.1	Diagrama circuitului	26
2.5.2	Pașii de conectare	26
3	Implementarea aplicației	29
3.1	Descrierea generală a implementării	29
3.2	Probleme speciale/dificultăți întâmpinate și modalități de rezolvare	29
3.3	Idei originale, soluții noi	31
3.4	Funcționarea sistemului	35
3.5	Comunicarea cu alte sisteme și salvarea/stocarea informațiilor	35
3.6	Interfața cu utilizatorul	35
4	Testarea aplicației și rezultate experimentale	37
4.1	Scenarii de test	38
4.1.1	Conexiune prin Bluetooth	38
4.1.2	Reglarea senzorilor și a valorilor	40
4.2	Rezultate experimentale	41
4.2.1	Urmărirea liniei negre	41
4.2.2	Evitarea obstacolelor	41
4.2.3	Comenzi manuale prin Bluetooth	42
Concluzii		43
Bibliografie		45
Anexe		47
1	Codul Sursa	47

Sistem mobil detectie traseu si evitare obstacole

Andrei-Ionuț SĂLĂVĂSTRU

Rezumat

În contextul actual al dezvoltării intense a sistemelor autonome și a robotilor, aplicarea tehnologiilor de control automatizat este din ce în ce mai frecventă. Proiectul de față prezintă dezvoltarea unui robot autonom capabil să urmeze o linie neagră și să evite obstacole, utilizând componente hardware bazate pe platforma Arduino și controlat printr-o interfață Bluetooth. Robotul este proiectat să fie controlat de la distanță folosind un telefon mobil, permitând utilizatorului să interacționeze eficient cu acesta.

Lucrarea de față prezintă o analiză detaliată a performanțelor robotului în diferite scenarii de testare, concentrându-se pe capacitatea acestuia de a urmări o linie neagră și de a evita obstacolele întâlnite pe traseu. Arhitectura dezvoltată urmărește integrarea unor algoritmi de control pentru navigarea autonomă și evitarea obstacolelor, precum și un sistem de monitorizare continuă a stării robotului.

În cadrul scenariilor de test propuse, robotul a demonstrat capacitatea de a urmări cu precizie linia neagră pe diverse trasee, atât simple cât și complexe. De asemenea, sistemul de evitare a obstacolelor a fost testat și s-a dovedit eficient atât pentru obstacole statice, cât și pentru cele dinamice. Testele au avut în vedere timpul de răspuns al cererilor transmise către robot prin intermediul conексiunii Bluetooth și viteza de transfer a datelor între telefon și robot.

Scenariile de test au inclus gestiunea comenziilor de deplasare înainte, înapoi, viraj la stânga și la dreapta, precum și activarea și dezactivarea modului autonom. Rezultatele testelor asupra arhitecturii create au indicat că robotul funcționează eficient în transmiterea comenziilor și în execuția acestora, indiferent de tipul acestora. Performanța generală a sistemului a fost satisfăcătoare, însă au fost identificate și unele probleme, cum ar fi pierderea aderenței pe suprafete cu un coeficient mic de freare. Se încearcă evitarea suprafețelor lucioase.

Lucrarea propune un studiu comparativ al comportamentului robotului în diverse condiții de testare, evidențiind avantajele și dezavantajele soluțiilor alese. Concluziile obținute sugerează că sistemul dezvoltat este eficient și robust, dar există și direcții de dezvoltare ulterioară, cum ar fi îmbunătățirea algoritmului de evitare a obstacolelor, integrarea unui sistem de viziune artificială și dezvoltarea unei aplicații mobile dedicate pentru un control mai intuitiv.

Arhitectura dezvoltată și scenariile de test propuse oferă o bază solidă pentru îmbunătățirea și extinderea funcționalităților robotului, asigurând astfel o performanță optimă în diverse condiții de operare.

Introducere

0.1. Motivatia alegerii temei

Într-o eră a tehnologiei avansate și a automatizării, interesul pentru robotică și sistemele autonome a crescut exponențial. Proiectul de față își propune să dezvolte un robot autonom capabil să urmărească o linie neagră și să evite obstacole, utilizând platforma Arduino Uno și diverse componente electronice. Alegera acestei teme este motivată de dorința de a explora și a înțelege principiile fundamentale ale roboticii, senzorilor și sistemelor de control, precum și de a contribui la dezvoltarea competențelor practice în acest domeniu. În plus, un astfel de proiect poate servi drept bază pentru aplicații mai complexe și dezvoltări viitoare.

0.2. Relevanța și contextul temei alese

În contextul industrializării și al implementării sistemelor inteligente în diverse sectoare, sistemele autonome joacă un rol crucial. Acestea sunt utilizate în procese de producție, logistică, servicii și chiar în explorarea spațiului. Proiectul de față se încadrează în acest context prin crearea unui prototip simplu, dar funcțional, care demonstrează capacitatea de a naviga autonom urmând o linie neagră și evitând obstacole. Utilizarea Arduino Uno, o platformă accesibilă și populară, facilitează înțelegherea și experimentarea, făcând acest proiect relevant atât pentru studenți și hobbyști, cât și pentru cercetători și ingineri începători.

0.3. Obiectivele generale ale lucrării

Principalele obiective ale acestei lucrări sunt:

- Conceperea și realizarea unui robot capabil să urmărească o linie neagră folosind senzori de culoare.
- Implementarea unui sistem de evitare a obstacolelor utilizând senzori ultrasonici.
- Integrarea și programarea componentelor electronice pentru a permite funcționarea autonomă a robotului.
- Testarea și evaluarea performanțelor robotului în diferite scenarii pentru a valida designul și implementarea.

0.4. Metodologia și instrumentele utilizate

Proiectul se bazează pe utilizarea platformei Arduino folosind următoarele componente:

- Arduino Uno R3 ATMEGA328P - microcontrolerul folosit pentru a controla robotul.
- Shield controlor motoare L293D pentru.
- 4 motoare TT cu reductor ce vor controla robotul.
- Două module cu senzor IR infraroșu ce ajută la detecția liniei.
- Senzor Ultrasonic HC-SR04 pentru detecția obiectelor și calcularea distanței.
- Servo motor SG90 9G ce va roti senzorul ultrasonic pentru a mări raza de detecție.
- 2 baterii litiu-ion 18650 ce vor alimenta robotul cu curent.

- Un suport de pus bateriile 18650.
- 4 roți ce sunt puse în fiecare motor.
- Modul Bluetooth HC-05 pentru a controla robotul de la distanță.
- Șasiu care va reprezenta structura de rezistență, o placă de lemn 14cm lungime și 9cm lățime.
- Cabluri de conectare.

Metodologia implică mai multe etape:

1. **Designul hardware-ului:** alegerea și conectarea componentelor electronice.
2. **Programarea software-ului:** dezvoltarea algoritmilor pentru urmărirea liniei și evitarea obstacolelor.
3. **Testarea și calibrarea:** ajustarea setărilor și optimizarea comportamentului robotului.
4. **Evaluarea performanței:** testarea robotului în diverse condiții și documentarea rezultatelor.

0.5. Structura lucrării

Lucrarea este structurată în patru capitoare principale, fiecare având un rol specific în dezvoltarea și documentarea proiectului:

- **Capitolul 1: Fundamentarea teoretică și documentarea bibliografică** Acest capitol oferă contextul, motivația și relevanța proiectului, prezentând obiectivele generale și metodologia utilizată.
- **Capitolul 2: Proiectarea hardware**
Acest capitol descrie în detaliu componentele utilizate, schema de conexiuni și considerențele tehnice necesare pentru realizarea hardware-ului robotului.
- **Capitolul 3: Implementarea aplicației**
Aici sunt prezentate algoritmi și codul sursă utilizat pentru controlul robotului. Se discută logica de urmărire a liniei și evitarea obstacolelor, precum și integrarea cu platforma Arduino.
- **Capitolul 4: Testarea aplicației și rezultate experimentale**
Acest capitol cuprinde descrierea procesului de testare, rezultatele obținute și o analiză a performanței robotului. Se discută eventualele îmbunătățiri și perspective de dezvoltare ulterioară.

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică

1.1. Domeniul și contextul abordării temei

Proiectul de față se înscrie în domeniul roboticii și automatizărilor, având ca scop dezvoltarea unui robot autonom capabil să urmărească o linie neagră și să evite obstacolele. Robotica este un domeniu interdisciplinar care integrează aspecte din inginerie mecanică, electronică, informatică și inteligență artificială. În contextul actual, automatizarea și utilizarea robotilor au devenit esențiale în diverse industrii, de la manufactură și logistică la medicină și servicii de întreținere. Dezvoltarea unui astfel de robot are aplicații practice semnificative, contribuind la creșterea eficienței și preciziei în execuția sarcinilor repetitive și periculoase.

În ultimele decenii, progresul tehnologic rapid a determinat o schimbare majoră în modul în care sunt realizate multe activități industriale și casnice. Automatizarea a permis reducerea costurilor și a timpului necesar pentru realizarea unor sarcini, crescând în același timp calitatea și consistența produselor și serviciilor oferte. Robotii autonomi, în special cei care pot naviga independent și interacționa cu mediul înconjurător, sunt în centrul acestei revoluții tehnologice.

Robotica, fiind un domeniu multidisciplinar, combină principiile și tehnologiile din inginerie mecanică pentru designul și construcția structurilor robotice, din electronică pentru dezvoltarea și implementarea circuitelor de control, din informatică pentru programarea și algoritmii de navigație și din inteligență artificială pentru a învăța și adapta comportamentele robotului. Această sinergie de cunoștințe și tehnologii permite crearea de sisteme complexe capabile să îndeplinească sarcini variate cu un nivel ridicat de autonomie.

Un robot autonom capabil să urmărească o linie neagră și să evite obstacolele este un exemplu concret de aplicare a acestor tehnologii interdisciplinare. Acest tip de robot poate fi utilizat în diverse scenarii practice. În fabrici, poate fi folosit pentru transportul materialelor pe trasee prestabilite, reducând astfel necesitatea intervenției umane și risurile asociate manipulării materialelor periculoase. În logistică, robotii de acest tip pot optimiza procesele de sortare și distribuție a produselor, asigurând livrări mai rapide și eficiente.

De asemenea, în domeniul medical, acești roboți pot fi folosiți pentru transportul medicamentelor și al echipamentelor medicale între diferite secții ale unui spital, reducând astfel riscul de contaminare și crescând eficiența operațională. În domeniul serviciilor de întreținere, robotii autonomi pot fi folosiți pentru curățenia clădirilor, întreținerea spațiilor verzi sau chiar pentru inspecția și reparația infrastructurii.

1.1.1. Evoluția tehnologică în domeniul roboticii

Evoluția tehnologică în domeniul roboticii a cunoscut un progres remarcabil în ultimele decenii, datorită avansurilor semnificative în tehnologiile de senzori, procesoare și algoritmi de control. De la apariția primilor roboți industriali în anii 1960, care erau mașini simple programate să execute sarcini repetitive, până la roboții autonomi de astăzi, capabili să învețe și să se adapteze la medii dinamice, dezvoltarea roboticii a fost constantă și rapidă.

1.1.1.1. Primele etape ale dezvoltării roboticii

Primele generații de roboți, precum Unimate, au fost utilizate în principal în industria manufacturieră pentru sarcini repetitive și periculoase, cum ar fi sudura și manipularea materialelor grele. Acești roboți erau controlați de programe fixe și nu aveau capacitatea de a se adapta sau de a învăța din experiență. Cu toate acestea, succesul lor în îmbunătățirea productivității și reducerea costurilor a deschis calea pentru cercetări și dezvoltări ulterioare.



Figura 1.1. Unimate - primul robot industrial construit vreodată.¹

1.1.1.2. Introducerea microprocesoarelor și senzorilor avansați

Introducerea microprocesoarelor și a senzorilor avansați a revoluționat multiple domenii, de la tehnologie și automatizări industriale, până la medicina modernă și dispozitivele de zi cu zi. Această evoluție a permis crearea de sisteme mai inteligente, eficiente și capabile să execute sarcini complexe cu o precizie ridicată.

Micropresesoarele

Micropresesoarele sunt unități centrale de procesare (CPU) integrate pe un singur circuit electronic. Acestea au evoluat rapid începând cu anii 1970, când Intel a lansat primul micropresesor comercial, Intel 4004. Integrarea microprocesoarelor a avut un impact profund:

- **Automatizare Industrială:** Microprosesoarele au permis automatizarea liniilor de producție prin control precis și rapid al mașinilor și echipamentelor industriale. Acest lucru a dus la creșterea productivității și a calității produselor, reducând în același timp costurile de producție.
- **Dispozitive Electronice de Consum:** Apariția microprosesoarelor a permis dezvoltarea unei game largi de dispozitive inteligente, cum ar fi computerele personale, smartphone-urile și electrocasnicele inteligente, toate acestea beneficiind de capacitate de calcul avansate și interfețe intuitive.
- **Automobile:** În industria auto, microprosesoarele sunt utilizate pentru gestionarea sistemelor complexe de control al motorului, frânelor, și chiar a sistemelor de navigație și divertisment.

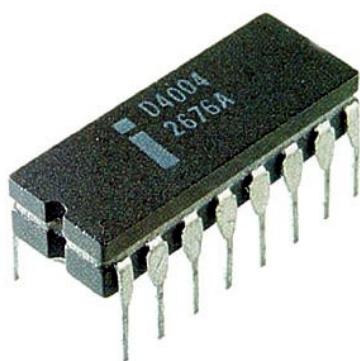


Figura 1.2. Intel 4004 - primul micropresesor realizat de firma Intel în anul 1971.²

¹Unimate Robot - <https://robotsguide.com/robots/unimate>

²Intel 4004 - <https://spectrum.ieee.org/chip-hall-of-fame-intel-4004-microprocessor>

Senzorii avansați

Senzorii avansați sunt dispozitive care detectează și măsoară condițiile fizice sau chimice, transformând aceste date în semnale care pot fi interpretate de microprocesoare. Dezvoltarea senzorilor avansați a avut următoarele efecte:

- **Automatizarea Proceselor:** În industria manufacturieră, senzorii avansați permit monitorizarea constantă a proceselor de producție, detectând anomalii sau defecte în timp real și ajustând automat parametrii pentru a menține calitatea.
- **Medicină:** Senzorii biomedicali sunt folosiți pentru monitorizarea stării pacienților, de la ritmul cardiac la nivelul de oxigen din sânge. Aceștia sunt esențiali în diagnosticarea precoce și în tratamente personalizate.
- **Internet of Things (IoT):** Senzorii sunt componente esențiale în IoT, permitând dispozitivelor să colecteze și să trimită date către rețele centrale. Acest lucru a dus la dezvoltarea caselor inteligente, orașelor inteligente și a aplicațiilor industriale conectate.
- **Vehicule Autonome:** În domeniul vehiculelor autonome, senzorii de tip lidar, radar și camerele de înaltă rezoluție sunt utilizati pentru a percepse mediul înconjurător, permitând mașinilor să navigheze și să ia decizii în timp real fără intervenția umană.

1.1.1.3. Viitorul roboticii

Privind spre viitor, se anticipatează că roboții vor deveni și mai autonomi și inteligenți. Dezvoltarea continuă a tehnologiilor de senzori, procesare și inteligență artificială va permite roboților să interacționeze

și să colaboreze cu oamenii într-un mod mai natural și intuitiv. De asemenea, se așteaptă ca roboții să joace un rol din ce în ce mai important în rezolvarea problemelor globale, cum ar fi schimbările climatice, crizele de sănătate publică și explorarea spațiului.

Evoluția tehnologică în domeniul roboticii nu numai că a transformat modul în care sunt realizate multe activități industriale și casnice, dar a deschis și noi oportunități pentru inovare și dezvoltare în diverse domenii. Prin continuarea cercetărilor și investițiilor în acest domeniu, roboții vor continua să aibă un impact semnificativ asupra societății și economiei globale.

1.1.2. Importanța automatizării în industrie

Automatizarea a devenit un element esențial în diverse sectoare industriale datorită capacitații sale de a îmbunătăți eficiența, productivitatea și calitatea proceselor de producție. Aceasta implică utilizarea tehnologiilor avansate, cum ar fi roboții industriali, sistemele de control automatizate și inteligență artificială, pentru a realiza sarcini care ar fi fost anterior efectuate manual. Importanța automatizării în industrie poate fi subliniată prin următoarele aspecte:

1.1.2.1. Creșterea productivității

Automatizarea permite desfășurarea continuă a operațiunilor de producție, fără întreruperi. Roboții industriali pot lucra non-stop, 24 de ore pe zi, 7 zile pe săptămână, ceea ce duce la o creștere semnificativă a productivității. În plus, automatizarea reduce timpul de ciclu al produselor, accelerând procesul de fabricație și permitând producerea unui volum mai mare de produse într-un timp mai scurt.

1.1.2.2. Îmbunătățirea calității produselor

Utilizarea sistemelor automatizate în procesele de producție reduce variațiile și erorile umane, asigurând o calitate consistentă a produselor. Roboții sunt capabili să execute sarcini cu o

precizie și repetabilitate mult mai mari decât operatorii umani, ceea ce contribuie la standardizarea calității și la reducerea defectelor.

1.1.2.3. Reducerea costurilor de producție

Automatizarea poate duce la economii semnificative în costurile de producție pe termen lung. Deși investiția inițială în tehnologia automatizată poate fi ridicată, beneficiile pe termen lung, cum ar fi reducerea costurilor cu forța de muncă, eficiența energetică și minimizarea deșeurilor, compensează aceste costuri. De asemenea, automatizarea permite o mai bună utilizare a materialelor și resurselor, optimizând procesele de producție.

1.1.2.4. Îmbunătățirea siguranței la locul de muncă

Automatizarea contribuie la îmbunătățirea condițiilor de siguranță la locul de muncă prin eliminarea necesității ca muncitorii să execute sarcini periculoase sau repetitive. Robotii pot prelua sarcinile care implică riscuri ridicate de accidente, expunere la substanțe chimice periculoase sau manipularea materialelor grele, reducând astfel riscul de răniri și îmbunătățind sănătatea și securitatea angajaților.

1.1.2.5. Flexibilitate și adaptabilitate

Tehnologiile de automatizare modernă permit o mare flexibilitate și adaptabilitate în procesele de producție. Sistemele automatizate pot fi reprogramate rapid pentru a schimba linia de producție sau pentru a fabrica produse diferite, în funcție de cerințele pieței. Această capacitate de adaptare este esențială într-un mediu de afaceri dinamic și competitiv, unde cerințele și preferințele consumatorilor se schimbă constant.

1.1.2.6. Crearea de locuri de muncă calificate

Deși automatizarea poate reduce necesitatea forței de muncă pentru sarcini repetitive și necalificate, aceasta creează oportunități pentru locuri de muncă noi și calificate în domenii precum întreținerea și programarea robotilor, analiza datelor și ingineria de automatizare. Angajații pot beneficia de formare și educație suplimentară pentru a se adapta la noile cerințe tehnologice, ceea ce duce la o forță de muncă mai calificată și mai inovatoare.

1.1.3. Aplicații practice ale robotilor autonomi

Robotii autonomi au devenit o componentă esențială în diverse industrii datorită capacitații lor de a opera independent și de a îndeplini sarcini complexe fără intervenția umană. Aceste aplicații practice variază de la producție și logistică până la agricultură și medicină, aducând beneficii semnificative în eficiență și siguranță.

În industria producției, robotii autonomi sunt utilizați pentru asamblarea produselor, manipularea materialelor și inspecția calității. De exemplu, în fabricile de automobile, robotii pot asambla piese cu o precizie și o viteză imposibile pentru oameni, reducând astfel costurile și crescând productivitatea. Robotii echipați cu senzori și algoritmi de inteligență artificială pot inspecta produsele finite pentru defecte, asigurând un control al calității mai riguros și mai consistent.

În logistică și depozitare, robotii autonomi joacă un rol crucial în gestionarea inventarului și expedierea produselor. Companii precum Amazon utilizează roboti autonomi pentru a transporta mărfurile în depozite, optimizând traseele pentru a reduce timpul de livrare și a minimiza erorile. Acești roboti sunt capabili să navigheze autonom prin spații mari și să interacționeze cu alte echipamente, ceea ce duce la o operare mai eficientă și mai sigură a depozitelor.

Agricultura este un alt domeniu unde robotii autonomi au un impact semnificativ. Robotii agricoli pot efectua o varietate de sarcini, inclusiv plantarea, irigarea și recoltarea culturilor. Ei pot monitoriza, de asemenea, starea solului și a plantelor, aplicând tratamente precise pentru a

maximiza randamentele și a reduce utilizarea pesticidelor și a îngrășămintelor. Această tehnologie nu numai că sporește productivitatea, dar și contribuie la practicile agricole durabile.

În domeniul medical, roboții autonomi sunt utilizați pentru intervenții chirurgicale de precizie, livrarea medicamentelor și transportul pacienților. De exemplu, roboții chirurgicali pot efectua operații minim invazive, oferind o precizie extraordinară și reducând timpul de recuperare pentru pacienți. În spitale, roboții autonomi pot transporta medicamente și echipamente între diferite secții, reducând povara asupra personalului medical și minimizând riscul de erori.

1.1.4. Beneficiile dezvoltării unui robot autonom

Dezvoltarea unui robot autonom pentru urmărirea unei linii și evitarea obstacolelor reprezintă nu doar o provocare tehnologică interesantă și educativă, dar are și un impact direct și pozitiv asupra numeroaselor domenii de activitate. Acest proiect explorează și aplică concepte teoretice într-un context practic, contribuind la dezvoltarea de soluții inovatoare și eficiente pentru problemele actuale ale industriei și societății.

1.2. Tema propusă

Tema propusă pentru acest proiect este dezvoltarea și implementarea unui robot autonom echipat cu un microcontroler Arduino, capabil să urmărească o linie neagră utilizând senzori infraroșii, să evite obstacolele prin intermediul senzorilor ultrasonici și să fie controlat de la distanță printr-o aplicație mobilă folosind un modul de comunicație Bluetooth.

Obiective

Obiectivele acestui proiect sunt următoarele:

- Proiectarea și construirea unui robot autonom capabil să urmărească o linie neagră trasată pe o suprafață.
- Implementarea unui sistem de evitare a obstacolelor, astfel încât robotul să poată naviga în mod eficient într-un mediu complex.
- Integrarea unui modul Bluetooth pentru a permite controlul robotului dintr-o aplicație de telefon mobil.
- Testarea și validarea funcționalităților robotului în diferite scenarii practice.

Justificarea Abordării

Justificarea abordării acestui proiect se bazează pe mai mulți factori esențiali:

Educație și Învățare

Proiectul oferă o oportunitate excelentă pentru aprofundarea cunoștințelor în domenii precum robotică, electronică, programare și telecomunicații. Prin implementarea diferitelor funcționalități, participanții își pot dezvolta abilitățile tehnice și pot obține o înțelegere practică a principiilor de bază ale acestor domenii.

Inovație și Creativitate

Integrarea mai multor tehnologii, cum ar fi senzorii de linie, senzorii de obstacole și modulele Bluetooth, stimulează creativitatea și inovația. Proiectul încurajează găsirea de soluții inovațioare pentru problemele tehnice întâlnite pe parcursul dezvoltării.

Aplicabilitate Practică

Robotul autonom dezvoltat în cadrul acestui proiect are aplicabilitate practică în diverse domenii. De exemplu, în logistică și depozitare, un astfel de robot poate fi utilizat pentru transportul autonom al mărfurilor pe trasee prestabilite. De asemenea, în domeniul educațional, robotul poate servi ca un instrument de învățare pentru studenți, demonstrând concepte de bază în robotică și automatizare.

Experiență în Controlul la Distanță

Controlul robotului prin intermediul unui telefon mobil folosind un modul Bluetooth adaugă un nivel suplimentar de complexitate și funcționalitate. Aceasta permite utilizatorilor să experimenteze cu interfețe de utilizator și comunicații wireless, care sunt abilități relevante în contextul actual al Internetului Lucrurilor (IoT).

1.3. Prezentare succintă și comparativă privind realizările actuale pe aceeași temă

Prezentare Succintă

Roboții autonomi care urmăresc linii și evită obstacole sunt proiecte populare în comunitățile de robotică și educație STEM³ datorită complexității lor moderate și aplicabilității practice. Acești roboți sunt adesea echipați cu microcontrolere precum Arduino, care le permit să interpreteze datele senzorilor și să execute acțiuni în timp real. Integrarea unui modul Bluetooth permite controlul de la distanță, aducând un plus de flexibilitate și interactivitate.

Realizări Actuale

Roboți Autonomi de Urmărire a Liniei

Mulți roboți comerciali și proiecte DIY⁴ folosesc senzori infraroșii pentru a detecta și a urmări linii negre pe diverse suprafete. Acești senzori sunt amplasati în partea inferioară a robotului și detectează diferențele de reflectivitate dintre linie și fundal. Exemple notabile includ:

- **Pololu 3pi Robot:** Un robot compact echipat cu senzori de linie și un microcontroler ATmega328P, capabil să urmărească linii cu mare precizie.



Figura 1.3. Pololu 3pi Robot⁵

³STEM - Science, Technology, Engineering and Mathematics <https://www.techtarget.com/whatis/definition/STEM-science-technology-engineering-and-mathematics>

⁴DIY - Do it yourself

⁵Pololu 3pi Robot <https://www.pololu.com/product/975>

Evitarea Obstacolelor

Pentru evitarea obstacolelor, roboții utilizează frecvent senzori ultrasonici sau senzori cu infraroșu sau chiar Lidar. Acești senzori permit detectarea obiectelor în calea robotului și ajustarea traseului pentru a preveni coliziunile. Exemple includ:

- **Robotul SEIT 100:** a fost creat pentru a oferi oamenilor oportunitatea de a transporta produse și materiale în mod sigur și eficient în depozite și fabrici.



Figura 1.4. SEIT 100⁶

Comparativ

Comparativ, realizările actuale variază în funcție de complexitate și cost. Roboții educaționali precum Elegoo Smart Robot Car Kit și Makeblock mBot sunt accesibili și ușor de utilizat pentru începători, oferind o introducere solidă în robotică. Pe de altă parte, soluțiile comerciale avansate precum Roomba de la iRobot oferă funcționalități sofisticate și autonomie extinsă, dar la un cost mai ridicat.

1.4. Analiza tipurilor de produse/aplicații existente din respectiva categorie a temei, tehnologii folosite pentru implementare

1.4.1. Tipuri de Produse/Aplicații Existente

Roboți de Urmărire a Liniei

- **Pololu 3pi Robot:** Un robot compact echipat cu senzori de linie și un microcontroler ATmega328P, capabil să urmărească linii cu mare precizie. Este utilizat în competiții de urmărire a liniei și pentru învățare în domeniul roboticii.
- **Tamiya Line Tracing Snail:** Un kit de robot educațional care folosește senzori de infraroșu pentru a detecta și urmări linii. Este ideal pentru începători care doresc să învețe principiile de bază ale roboticii.

Roboți de Evitare a Obstacolelor

- **iRobot Roomba:** Un robot aspirator comercial care utilizează senzori ultrasonici, infraroși și camere pentru a naviga autonom în locuințe și a evita obstacolele. Este un exemplu de aplicație avansată a tehnologiilor de evitare a obstacolelor.
- **BoBoi Car:** Un proiect open-source bazat pe Arduino Uno care folosește un senzor ultrasonic pentru a detecta și evita obstacolele. Este un proiect educațional popular în comunitățile de maker.

⁶SEIT 100 - robot produs de Milvus https://www.bolee.com.hk/wp-content/uploads/2021/09/MR-SEIT-100-User-Manual_v2.2.pdf

Roboți cu Control Bluetooth

- **DFRobot MiniQ:** Un robot compact care include un modul Bluetooth și poate fi controlat printr-o aplicație mobilă personalizată. Este utilizat în scopuri educaționale pentru a demonstra principiile de control la distanță.
- **Makeblock mBot:** O platformă educațională care combină urmărirea liniei, evitarea obstacolelor și controlul prin Bluetooth. Utilizează un microcontroler bazat pe Arduino și este ideal pentru studenți și hobby-iști.

1.4.2. Tehnologii Folosite pentru Implementare

Microcontrolere

- **Arduino:** Platforma Arduino este foarte populară pentru proiectele de robotică datorită simplității și flexibilității sale. Modele precum Arduino Uno și Mega sunt frecvent utilizate pentru a controla senzorii și actuatoarele roboților.
- **ATmega328P:** Acest microcontroler, folosit în multe modele de Arduino, este adesea întâlnit în roboții de urmărire a liniei datorită capacităților sale adecvate de procesare și interfațări cu senzori.

Senzori de Linie

- **Senzori Infraroși:** Folosiți pentru a detecta contrastul dintre linia neagră și suprafața albă. Senzorii IR, precum TCRT5000, sunt populari datorită sensibilității și preciziei lor.

Senzori de Obstacole

- **Senzori Ultrasonici:** Module precum HC-SR04 sunt utilizate pentru a măsura distanțele față de obstacole și a evita coliziunile. Acești senzori sunt precisi și ușor de implementat.
- **Senzori Infraroși:** Senzorii IR pot fi folosiți și pentru detectarea obstacolelor la distanțe mai mici. Sunt adesea utilizați în combinație cu alți senzori pentru a oferi o navigație mai sigură.

Module de Comunicație Bluetooth

- **HC-05/HC-06:** Aceste module sunt populare pentru controlul wireless al roboților. Ele permit comunicații seriale între microcontroler și un dispozitiv mobil, facilitând controlul la distanță prin aplicații personalizate pentru Android sau iOS.

1.5. Elaborarea specificațiilor privind caracteristicile așteptate de la aplicație

Specificații privind Caracteristicile Așteptate de la Aplicație

1. Funcționalități de Bază

1.1 Urmărirea Liniei

- **Senzori de Linie:** Robotul trebuie să fie echipat cu cel puțin doi senzori infraroșii pentru a detecta și urmări linia neagră trasată pe o suprafață albă.
- **Ajustare Automată:** Robotul trebuie să ajusteze direcția de deplasare pentru a rămâne pe linie în mod constant, folosind algoritmi de control.

1.2 Evitarea Obstacolelor

- **Senzori de Obstacole:** Robotul trebuie să fie echipat cu senzori ultrasonici și/sau senzori infraroșii pentru detectarea obstacolelor în calea sa.
- **Reacție Rapidă:** Robotul trebuie să fie capabil să detecteze obstacolele și să ajusteze traseul în timp real pentru a evita coliziunile.

1.3 Control Bluetooth

- **Modul Bluetooth:** Robotul trebuie să includă un modul Bluetooth HC-05 pentru comunicații wireless.
- **Aplicație Mobilă:** Trebuie să existe o aplicație mobilă pentru Android sau iOS care să permită controlul manual al robotului și monitorizarea stării acestuia.

2. Caracteristici Tehnice

2.1 Platforma Hardware

- **Microcontroler:** Robotul trebuie să fie echipat cu un microcontroler Arduino (de exemplu, Arduino Uno sau Mega) pentru gestionarea senzorilor și a actuatorilor.
- **Motoare:** Robotul trebuie să utilizeze motoare de curent continuu cu control PWM (Pulse Width Modulation) pentru mișcare precisă.

2.2 Senzori

- **Senzori Infraroșii pentru Linie:** Minim doi senzori TCRT5000 sau echivalenți.
- **Senzori Ultrasonici pentru Obstacole:** Minim un senzor HC-SR04 sau echivalent.

2.3 Comunicații

- **Modul Bluetooth:** HC-05 pentru comunicații seriale wireless.
- **Interfață Serială:** Robotul trebuie să fie capabil să comunice printr-o interfață serială cu aplicația mobilă.

3. Performanțe și Eficiență

3.1 Eficiență Energetică

- **Consum Redus de Energie:** Robotul trebuie să fie optimizat pentru un consum minim de energie, utilizând surse de alimentare eficiente.
- **Durată Lungă a Bateriei:** Robotul trebuie să funcționeze continuu pentru cel puțin 2 ore cu o singură încărcare.

3.2 Precizia și Viteză

- **Precizia Urmăririi Liniei:** Robotul trebuie să fie capabil să urmeze o linie cu o deviere maximă de 1 cm.
- **Viteză Maximă:** Robotul trebuie să aibă o viteză ajustabilă, cu o viteză maximă de cel puțin 30 cm/s.

4. Ușurință în Utilizare

4.1 Configurare și Programare

- **Interfață de Programare:** Robotul trebuie să fie ușor de programat, folosind mediul de dezvoltare Arduino IDE.
- **Documentație Detaliată:** Trebuie să fie disponibilă o documentație completă și detaliată pentru configurare și programare.

4.2 Interfața Utilizatorului

- **Aplicație Intuitivă:** Aplicația mobilă trebuie să fie intuitivă și ușor de utilizat, cu o interfață grafică prietenoasă.
- **Funcționalități Multiple:** Aplicația trebuie să ofere funcționalități precum controlul manual, monitorizarea stării robotului și ajustarea setărilor.

Capitolul 2. Proiectarea hardware

Pentru acest proiect, platforma hardware aleasă este Arduino Uno, un microcontroler popular datorită ușurinței de utilizare, flexibilității și suportului vast din comunitatea open-source. Această platformă include un microcontroler ATmega328P care oferă suficiente resurse pentru a gestiona senzori infraroșu, un senzor de distanță ultrasonic și module de control pentru motoare.

Arduino Uno R3 este una dintre cele mai populare plăci de dezvoltare din familia Arduino, fiind larg utilizată în proiecte educaționale și hobby-uri datorită simplității sale și a comunității extinse de utilizatori. În această secțiune, vom explora specificațiile, avantajele și dezavantajele acestei plăci.

2.1. Specificații tehnice

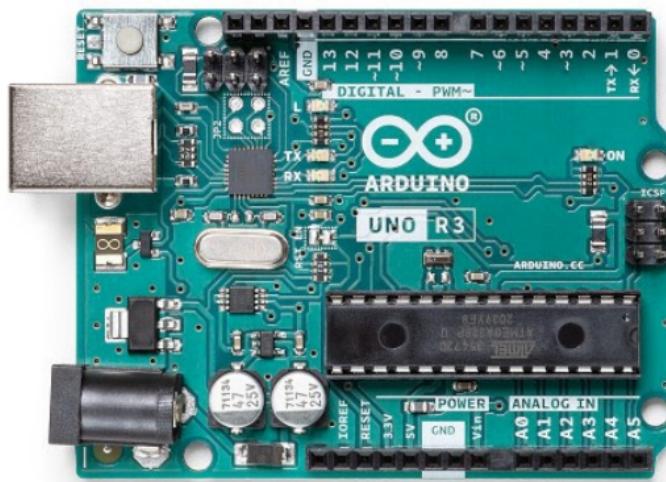


Figura 2.1. Arduino Uno R3⁷

Arduino Uno R3 este echipată cu un microcontroler ATmega328P și include următoarele caracteristici tehnice:

- **Microcontroler:** ATmega328P
- **Tensiune de operare:** 5V
- **Tensiune de intrare (recomandată):** 7-12V
- **Tensiune de intrare (limite):** 6-20V
- **Pini digitali de I/O:** 14 (dintre care 6 pot fi folosiți ca ieșiri PWM)
- **Pini de intrare analogică:** 6
- **Curent continuu pe pinul I/O:** 20 mA
- **Curent pentru pinul de 3.3V:** 50 mA

⁷Arduino Uno - <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

- **Memorie flash:** 32 KB (din care 0.5 KB sunt folosiți de bootloader)
 - **SRAM:** 2 KB
 - **EEPROM:** 1 KB
 - **Frecvență de ceas:** 16 MHz

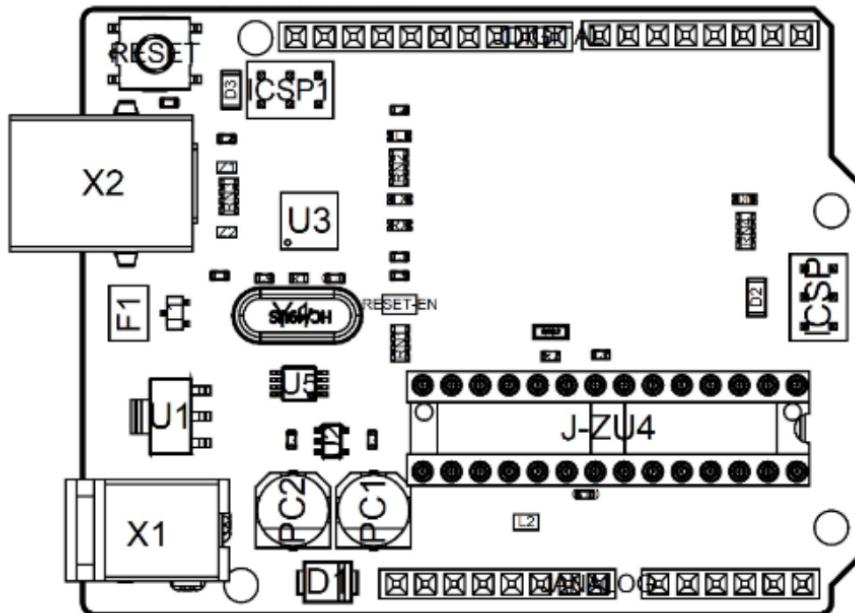


Figura 2.2. Arduino Uno R3 - Topologia placii⁸

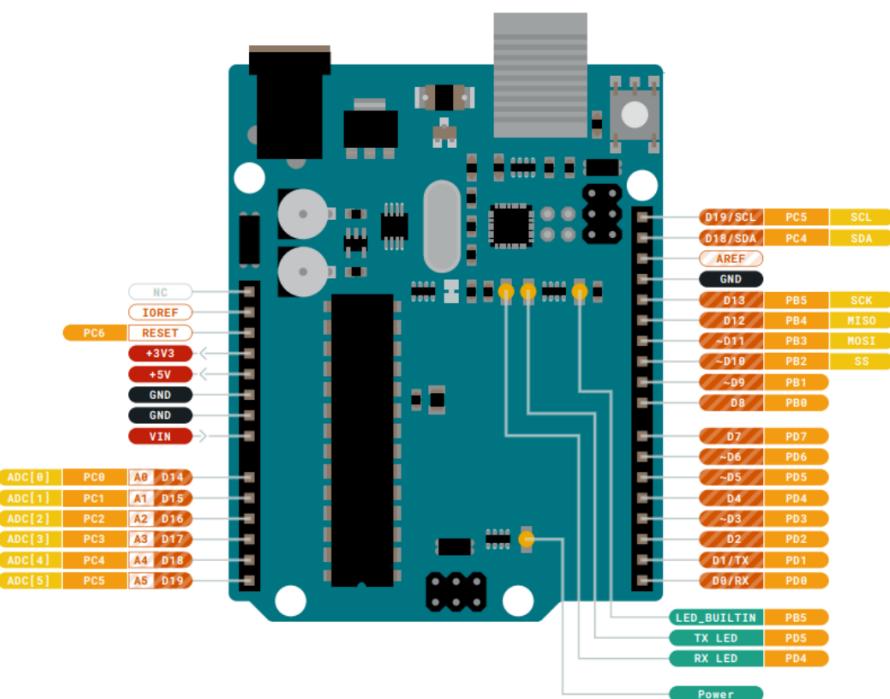


Figura 2.3. Arduino Uno R3 - Conexiunea Pinilor⁹

⁸Arduino Uno - <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

2.1.1. Aplicații tipice ale Arduino Uno R3

Arduino Uno R3 este versatil și utilizat într-o varietate de aplicații:

- **Proiecte educationale:** Utilizat în școli și universități pentru a introduce studenții în electricitate și programare.
- **Automatizare și control:** Poate fi folosit pentru a automatiza procese simple și a controla diverse dispozitive.
- **Prototipuri rapide:** Ideal pentru dezvoltarea rapidă a prototipurilor datorită flexibilității și ușurinței de utilizare.
- **Proiecte DIY:** Folosit de entuziaști pentru a construi proiecte personale, cum ar fi roboți, sisteme de irigare, alarme și multe altele.

2.2. Modulele generale ale aplicației și interacțiunile dintre ele

Aplicația robotului este împărțită în mai multe module principale:

2.2.1. Modulul de control al motoarelor

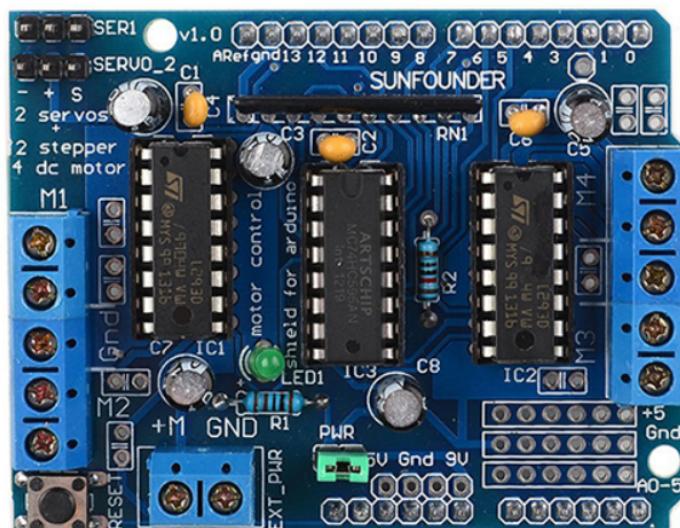


Figura 2.4. L293D - Modulul de control al motoarelor¹⁰

Driverul de motor L293D este un circuit integrat popular utilizat pentru controlul motoarelor în diverse proiecte de robotică și automatizare. Este ideal pentru aplicații care necesită control bidirecțional al motoarelor DC și este compatibil cu placile Arduino, inclusiv Arduino Uno R3.

2.2.1.1. Specificații tehnice

Driverul de motor L293D are următoarele specificații tehnice:

- **Alimentare logică (Vss):** 4.5V până la 7V
- **Alimentare motor (Vs):** 4.5V până la 36V
- **Curent maxim per canal:** 600mA

⁹Arduino Uno - <https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>

¹⁰L293D - http://wiki.sunfounder.cc/index.php?title=L293D_Motor_Driver_Shield

- **Curent de vârf (non-repetitiv):** 1.2A pe canal
- **Număr de canale:** 2 (poate controla 2 motoare DC sau 1 motor pas cu pas)
- **Diode interne de protecție la supratensiune**
- **Control bidirecțional:** Permite rotația motorului în ambele direcții (înainte și înapoi)

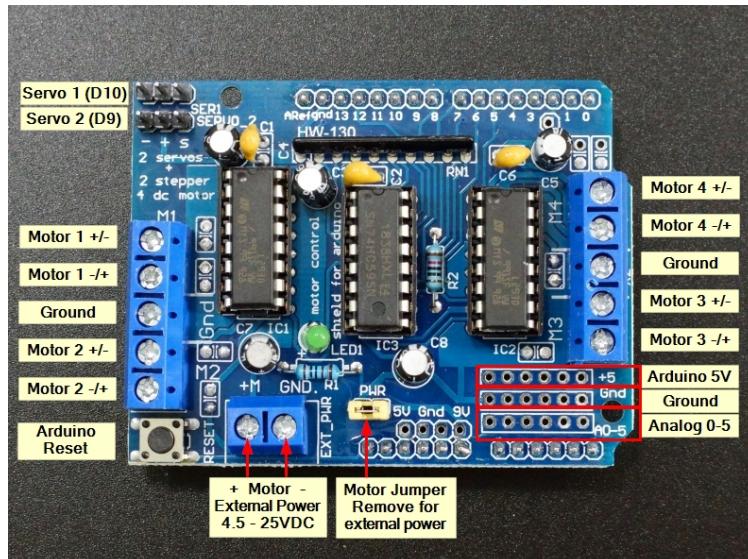


Figura 2.5. L293D - Conexiunea pinilor¹¹

2.2.1.2. Rolul în sistem

Driverul de motor L293D este utilizat pentru a controla motoarele DC ale robotului, permitând mișcarea înainte, înapoi, și efectuarea de viraje. Acesta primește semnale de control de la microcontrolerul Arduino și reglează tensiunea și direcția curentului către motoare, în funcție de instrucțiunile primite.

L293D include diode interne de protecție care protejează circuitul împotriva supratensiunilor generate de motoare în timpul funcționării. Aceste diode sunt esențiale pentru prevenirea deteriorării componentelor electronice datorită tensiunilor de tip spike care apar la comutarea motoarelor.

2.2.1.3. Interfațarea cu Arduino Uno R3

L293D primește semnale de control de la microcontrolerul Arduino, care sunt utilizate pentru a decide direcția și viteza motoarelor. Arduino trimitе aceste semnale către pinii de control și de activare ai driverului, bazându-se pe datele primite de la senzori și pe instrucțiunile programate. De exemplu:

- **Instrucțiuni de mișcare înainte:** Dacă senzorii IR detectează că robotul se află pe linia corectă, Arduino trimitе semnale pentru a roti motoarele în sensul acelor de ceasornic.
- **Instrucțiuni de viraj la stânga sau dreapta:** Dacă senzorii detectează necesitatea unui viraj, Arduino ajustează starea pinilor de control pentru a schimba direcția de rotație a unuia sau ambelor motoare.
- **Instrucțiuni de oprire:** Dacă un obstacol este detectat de senzorul ultrasonic, Arduino trimitе semnale pentru a opri motoarele, prevenind coliziunea.

¹¹L293D - http://wiki.sunfounder.cc/index.php?title=L293D_Motor_Driver_Shield

2.2.2. Modulul de senzori infraroșu



Figura 2.6. IR Sensor - Senzor de detecție a culorii¹²

Senzorii infraroșu (IR) sunt esențiali pentru detectarea liniei negre pe care robotul trebuie să o urmeze. Acești senzori sunt capabili să detecteze contrastele de culoare prin măsurarea reflexiei luminii infraroșii de pe suprafața pe care se deplasează robotul.

2.2.2.1. Specificații tehnice

Un senzor de culoare IR tipic are următoarele specificații tehnice:

- Tensiune de operare:** 3.3V - 5V
- Curent de operare:** 20mA
- Distanță de detectare:** 1 - 2 cm
- Lungime de undă IR:** Aproximativ 940 nm
- Unghi de detecție:** 35 - 45 grade
- Dimensiuni:** Compact și ușor de montat pe platforma robotului

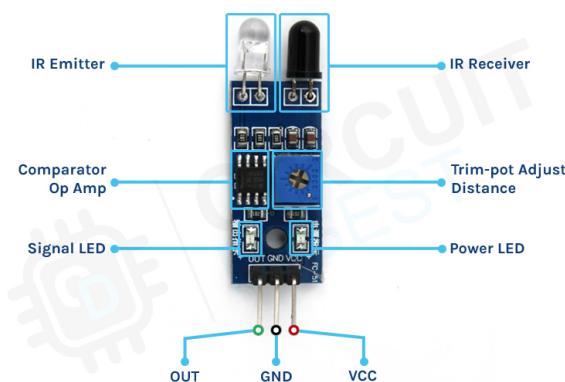


Figura 2.7. Conexiune pini senzor infraroșu¹³

¹²IR Senzor - <https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino>

¹³IR Senzor Pins - <https://circuitdigest.com/microcontroller-projects/interfacing-ir-sensor-module-with-arduino>

2.2.2.2. Rolul în sistem

Senzorii de culoare IR sunt montați pe partea inferioară a robotului, îndreptați către sol. Aceștia sunt plasați strategic astfel încât să poată detecta linia neagră pe care robotul trebuie să o urmeze. Fiecare senzor emite lumină infraroșie și măsoară intensitatea luminii reflectate de suprafața de dedesubt.

- **Detectarea liniei:** Atunci când senzorul se află deasupra unei linii negre, reflectivitatea este scăzută și senzorul detectează această schimbare, generând un semnal corespunzător.
- **Semnal digital:** Senzorii trimit semnale digitale către microcontrolerul Arduino, indicând dacă se află deasupra unei suprafete negre (linie) sau albe (fundal).
- **Corectarea traectoriei:** Arduino procesează aceste semnale și ajustează mișcarea motoarelor pentru a menține robotul pe linie.

2.2.3. Modulul senzorului ultrasonic



Figura 2.8. HC-SR04¹⁴

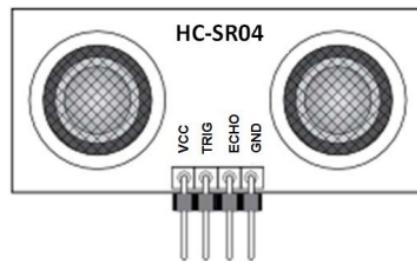
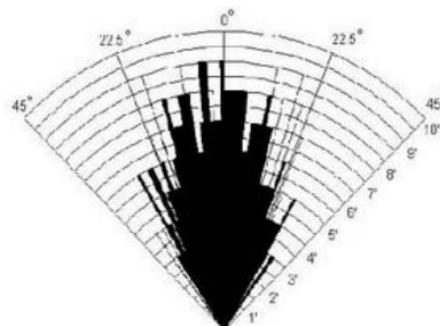
Senzorul ultrasonic HC-SR04 este utilizat pentru măsurarea distanței față de obstacole, fiind o componentă esențială în proiectele de robotică pentru evitarea coliziunilor. Acest senzor funcționează prin emiterea unui puls ultrasonic și măsurarea timpului necesar pentru ca ecoul să se întoarcă, permitând astfel calcularea distanței până la obiectele din calea sa.

2.2.3.1. Specificații tehnice

Senzorul HC-SR04 are următoarele specificații tehnice:

- **Tensiune de operare:** 5V
- **Curent de operare:** 15mA
- **Unghi de detecție:** 15 grade
- **Gama de măsurare:** 2 cm - 400 cm
- **Precizie:** ± 3 mm
- **Frecvență ultrasonică:** 40 kHz
- **Dimensiuni:** 45mm x 20mm x 15mm

¹⁴HC-SR04 - senzor de detecție a obiectelor <https://www.hwlibre.com/ro/hc-sr04/>

Figura 2.9. HC-SR04 - conexiune pinii¹⁵Figura 2.10. HC-SR04 - precizia de detectie a obiectelor¹⁶

Senzorul ultrasonic HC-SR04 joacă un rol crucial în detectarea obstacolelor și în navigarea autonomă a robotului. Acesta este utilizat pentru a măsura distanța față de obiectele din fața robotului și pentru a ajuta la evitarea coliziunilor. Iată cum funcționează în detaliu:

- Emitere și recepție de unde ultrasonice:** Senzorul are două componente principale - un emițător care generează unde ultrasonice și un receptor care captează ecoul acestor unde după ce au lovit un obiect.
- Calculul distanței:** Timpul scurs între emiterea undei și receptia ecului este folosit pentru a calcula distanța până la obiect. Formula utilizată este: Distanță (cm) = (Timp (μs) / 2) / 29.1.
- Evitarea obstacolelor:** Dacă senzorul detectează un obiect la o distanță mai mică decât o valoare prestabilită, microcontrolerul Arduino primește semnalul și inițiază manevre de evitare, cum ar fi schimbarea direcției sau oprirea robotului.

¹⁵HC-SR04 - conectare pinilor <https://www.handsontec.com/datasheets/HC-SR04-Ultrasonic.pdf>

¹⁶HC-SR04 - precizie <https://www.handsontec.com/datasheets/HC-SR04-Ultrasonic.pdf>

2.2.4. Modulul Bluetooth



Figura 2.11. HC-05 - modulul Bluetooth compatibil cu telefoane Android¹⁷

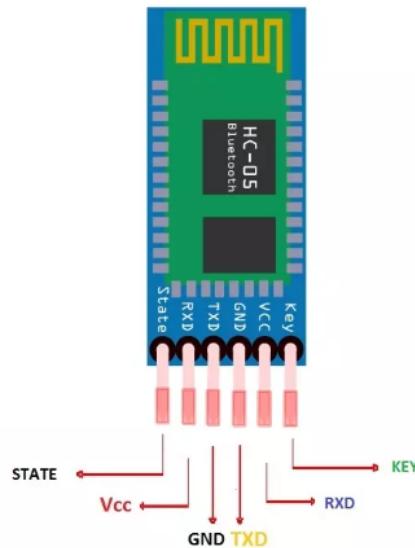
Modulul Bluetooth HC-05 este utilizat pentru a adăuga capabilități de comunicație wireless robotului tău. Acest modul permite controlul robotului de la distanță prin intermediul unui dispozitiv compatibil Bluetooth, cum ar fi un smartphone, tabletă sau computer.

2.2.4.1. Specificații tehnice

Modulul HC-05 are următoarele specificații tehnice:

- **Protocol Bluetooth:** V2.0+EDR (Enhanced Data Rate)
- **Tensiune de operare:** 3.3V
- **Consum de curent:** Aproximativ 30mA în modul de operare, 8mA în modul de aşteptare
- **Rază de acțiune:** Până la 10 metri
- **Rate de transmisie:** Asynchronous: 2.1 Mbps / 160 kbps, Synchronous: 1 Mbps / 1 Mbps
- **Moduri de operare:** Master, Slave, Loopback, și multe altele
- **Dimensiuni:** 28mm x 15mm x 2.35mm
- **Temperatură de operare:** -20°C până la +75°C

¹⁷HC-05 - modul Bluetooth <https://www.theengineeringprojects.com/2019/10/hc-05-bluetooth-module-pinout-datasheet-features-applications.html>

Figura 2.12. HC-05 - Conexiunea pinilor¹⁸

2.2.4.2. Rolul în sistem

Modulul Bluetooth HC-05 permite robotului să primească comenzi și să transmită date wireless. Aceasta poate fi configurat atât în modul Master, cât și în modul Slave, însă cel mai des este utilizat în modul Slave pentru a permite controlul de la distanță al robotului de către un dispozitiv central, cum ar fi un smartphone sau un computer.

- **Comunicare serială:** HC-05 comunică cu microcontrolerul Arduino prin intermediul unei interfețe seriale UART. Aceasta permite transmiterea și recepționarea datelor într-un mod simplu și eficient.
- **Control de la distanță:** Utilizatorul poate trimite comenzi către robot pentru a controla mișcarea, direcția și alte funcționalități, cum ar fi pornirea sau oprirea senzorilor.
- **Feedback în timp real:** Robotul poate trimite date înapoi către dispozitivul de control, oferind feedback în timp real despre starea sa, distanța până la obstacole, etc.

2.2.5. Modulul de servomecanisme

Figura 2.13. Micro servo 9g SG90 - servomotor¹⁹

¹⁸HC-05 - conectarea pinilor <https://www.theengineeringprojects.com/2019/10/hc-05-bluetooth-module-pinout-datasheet-features-applications.html>

Servomecanismele, sau servomotoarele, sunt componente esențiale în multe aplicații de robotică și automatizare, oferind control precis al poziției și rotației. Servomotorul SG90 este unul dintre cele mai populare servomotoare datorită dimensiunilor sale compacte, costului redus și ușurinței în utilizare.

2.2.5.1. Specificații tehnice

Servomotorul SG90 are următoarele specificații tehnice:

- **Tensiune de operare:** 4.8V - 6.0V
- **Cuplu de rotire:** 1.8 kg-cm (la 4.8V)
- **Unghi de rotație:** 0° - 180°
- **Viteză:** 0.1 secunde per 60° la 4.8V
- **Dimensiuni:** 22.2mm x 11.8mm x 31mm
- **Greutate:** 9g
- **Tip control:** PWM (Pulse Width Modulation)

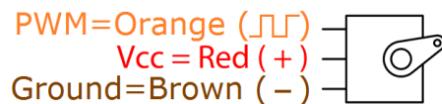


Figura 2.14. Micro servo 9g SG90 - datasheets²⁰

2.2.5.2. Rolul în sistem

Servomotorul SG90 este utilizat pentru a ajusta unghiul senzorului ultrasonic HC-SR04, permitându-i să scaneze zona din jurul robotului pentru a detecta obstacole. Senzorul ultrasonic este atașat servomotorului, care se rotește stânga-dreapta pentru a acoperi o arie mai mare și pentru a oferi o vedere panoramică a împrejurimilor robotului.

- **Control precis al poziției:** Servomotorul poate fi setat să se rotească la un unghi specific între 0° și 180°, oferind control precis asupra orientării senzorului ultrasonic.
- **Scanarea zonei:** Prin mutarea senzorului ultrasonic la diferite unghiuri, robotul poate scana zona din față și din laterale pentru a detecta obstacole și a planifica rutile de evitare.
- **Flexibilitate în utilizare:** Poate fi utilizat și pentru alte aplicații, cum ar fi controlul brațelor robotice, mecanisme de prindere sau orice altă componentă care necesită control precis al poziției.

¹⁹Micro servo 9g SG90 https://robojax.com/learn/arduino/robojax-servo-sg90_datasheet.pdf

²⁰Micro servo 9g SG90 - conexiune https://robojax.com/learn/arduino/robojax-servo-sg90_datasheet.pdf

2.2.6. Motoarele electrice DC TT Motor 200 RPM



Figura 2.15. Motorul electric DC TT 3-6V²¹

Motoarele electrice DC TT Motor 200 RPM sunt componente esențiale pentru mișcarea robotului, oferind o combinație echilibrată între viteză și cuplu. Aceste motoare sunt ideale pentru proiecte de robotică datorită dimensiunilor compacte, eficienței și ușurinței de control.

2.2.6.1. Specificații tehnice

Motoarele TT Motor 200 RPM au următoarele specificații tehnice:

- **Tensiune de operare:** 3V - 6V
- **Viteză nominală:** 200 RPM (rotații pe minut) la 6V
- **Curent de operare fără sarcină:** Aproximativ 150mA
- **Curent de blocare:** Aproximativ 1.6A
- **Cuplu de blocare:** Aproximativ 800 gf.cm (gram-forță centimetru)
- **Raport de transmisie:** 1:48
- **Dimensiuni:** 70mm x 23mm x 18mm
- **Greutate:** Aproximativ 45g
- **Tip ax:** Ax D de 6mm

2.2.6.2. Rolul în sistem

Motoarele TT Motor 200 RPM sunt utilizate pentru a propulsa robotul, permitându-i să se deplaseze înainte, înapoi și să efectueze viraje. Aceste motoare sunt montate pe șasiul robotului și sunt controlate prin intermediul driverului de motor L293D, care reglează tensiunea și direcția curentului pentru a obține mișcarea dorită.

- **Propulsie:** Motoarele generează cuplul necesar pentru a deplasa robotul pe suprafață, fie că este vorba de mișcare liniară sau de efectuarea virajelor.
- **Controlul direcției:** Prin controlul independent al fiecărui motor, robotul poate efectua mișcări precise, cum ar fi virajele stânga și dreapta, și rotația pe loc.
- **Stabilitate și tracțiune:** Motoarele TT oferă un echilibru între viteză și cuplu, asigurând o deplasare stabilă și constantă, chiar și pe suprafete variate.

²¹Motor electric - <https://www.gotronic.fr/pj2-com-motor01-datasheet-2079.pdf>

2.3. Avantaje și dezavantaje ale metodei alese

Avantaje:

- **Simplitate și accesibilitate:**

- **Arduino Uno R3:** Platforma Arduino este bine cunoscută pentru simplitatea și accesibilitatea sa, oferind un punct de plecare excelent pentru dezvoltatori începători și avansați. Biblioteca vastă și numeroasele exemple de cod disponibile online fac dezvoltarea rapidă și ușoară.
- **Motoare TT Motor 200 RPM:** Aceste motoare sunt ușor de interfațat și controlat prin intermediul driverelor de motor, permitând implementarea rapidă și eficientă a mișcării robotului.
- **Senzorii IR:** Sunt simpli de utilizat și oferă un mod eficient de a detecta linia neagră pentru urmărirea traseului. Configurarea lor este directă, necesitând doar conexiuni de bază la pinii de intrare analogică sau digitală.
- **Senzorul ultrasonic HC-SR04:** Este un senzor de distanță accesibil și precis, ușor de interfațat cu Arduino pentru măsurători de distanță și evitarea obstacolelor.
- **Servomotorul SG90:** Oferă control precis al poziției la un preț accesibil, fiind ideal pentru aplicații care necesită ajustări de unghi, cum ar fi rotația senzorului ultrasonic.
- **Modulul Bluetooth HC-05:** Permite comunicații wireless simple și eficiente, facilitând controlul robotului de la distanță printr-un dispozitiv mobil sau computer.

Dezavantaje:

- **Limitări de performanță:**

- **Arduino Uno R3:** ATmega328P are resurse limitate în termeni de memorie RAM, flash și viteza de procesare comparativ cu alte microcontrolere sau plăci de dezvoltare mai avansate. Acest lucru poate limita complexitatea și funcționalitatea aplicațiilor care pot fi dezvoltate pe această platformă.
- **Driverul de motor L293D:** Deși este eficient pentru controlul de bază al motoarelor, are limitări în ceea ce privește curentul maxim pe care îl poate livra, ceea ce poate restricționa utilizarea motoarelor cu cerințe de putere mai mari.
- **Senzorii IR și HC-SR04:** Performanța lor poate fi afectată de condiții de mediu, cum ar fi lumina ambientală puternică sau suprafetele reflectorizante, care pot genera citiri eronate.

2.4. Limitele metodei alese

2.4.1. Condiții de Iluminare

- **Lumină ambientală:** Metoda va funcționa bine în condiții de iluminare constantă și moderată. Lumină ambientală prea puternică sau prea slabă poate afecta senzorii infraroșii, ducând la citiri eronate ale liniei negre.
- **Reflexii și umbre:** Suprafetele care generează reflexii puternice sau umbre pot perturba performanța senzorilor. Este important ca traseul să fie amenajat astfel încât să minimizeze aceste efecte.

2.4.2. Caracteristicile suprafetei

- **Tipul de suprafete:** Suprafetele trebuie să fie relativ netede și uniforme. Suprafetele rugoase sau neuniforme pot afecta stabilitatea robotului și acuratețea senzorilor.
- **Contrastul liniei:** Linia neagră trebuie să fie bine definită și să aibă un contrast suficient față de fundal. Traseele cu contrast redus pot duce la dificultăți în detectarea liniei.

2.4.3. Distanța și tipul obstacolelor

- **Detectarea obstacolelor:** Senzorul de proximitate are o distanță limitată de detectare. Obstacolele trebuie să fie situate în intervalul de detecție eficientă a senzorului, de obicei câțiva centimetri până la câțiva zeci de centimetri.
- **Tipul de obstacole:** Obstacolele trebuie să fie suficient de mari pentru a fi detectate. Obstacolele foarte mici sau foarte mari pot să nu fie detectate corect sau la timp.

2.4.4. Raza de acțiune a modulului Bluetooth

- **Distanța de control:** Modulul Bluetooth are o rază de acțiune limitată, de obicei între 10 și 20 de metri. Robotul trebuie controlat dintr-o distanță relativ mică pentru a asigura o comunicare stabilă.
- **Interferențe electromagnetice:** Prezența altor dispozitive Bluetooth sau surse de interferență electromagnetică poate afecta comunicarea. Este recomandat să operezi robotul într-un mediu cu puține interferențe.

2.4.5. Capacitatea de procesare a Arduino Uno

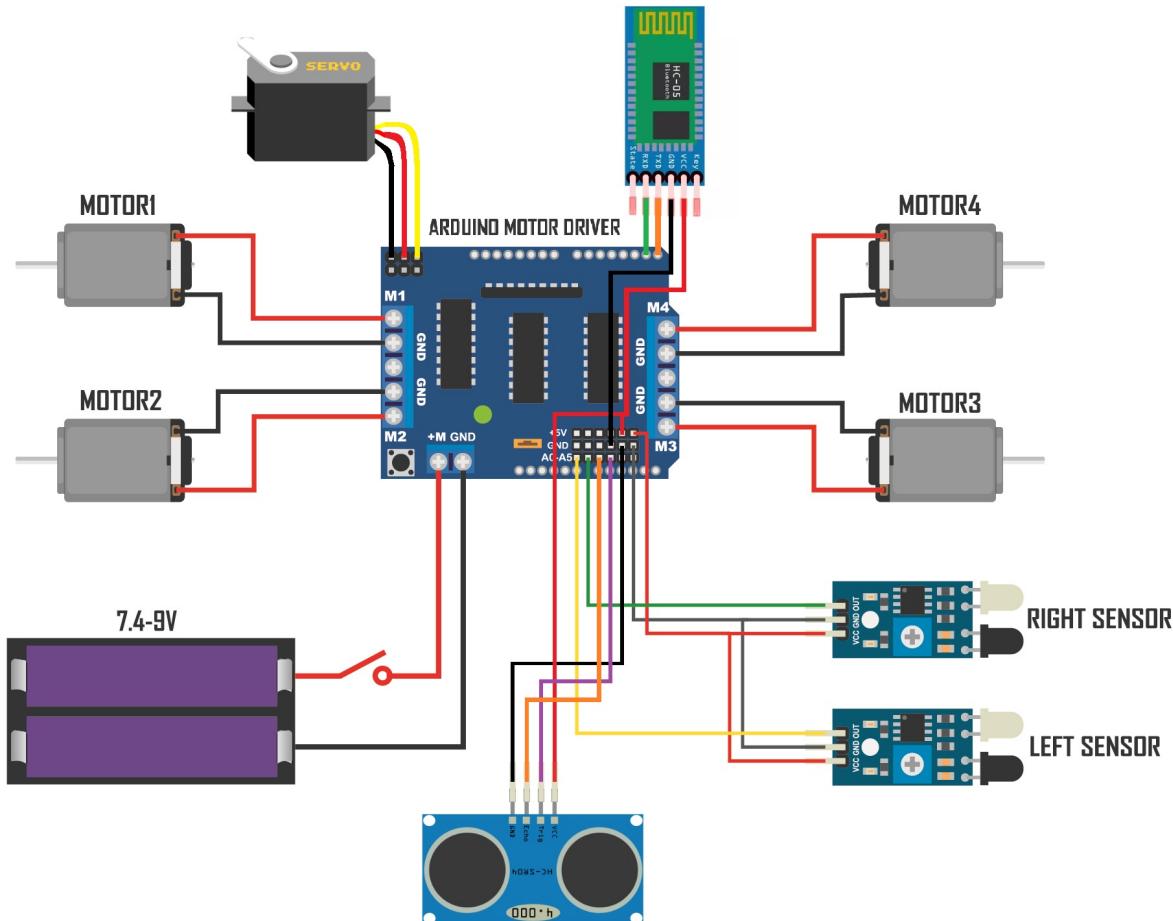
- **Complexitatea algoritmilor:** Algoritmii de control trebuie să fie suficient de simpli pentru a putea fi procesați de Arduino Uno. Algoritmii prea complexi pot depăși capacitatele de procesare și memorie ale microcontrolerului, afectând performanța robotului.
- **Timpul de reacție:** Arduino Uno trebuie să proceseze rapid datele de la senzori pentru a asigura un răspuns în timp real. Întârzierile în procesare pot duce la performanțe slabe în urmărirea liniei și evitarea obstacolelor.

2.4.6. Calitatea componentelor utilizate

- **Sensibilitatea și precizia senzorilor:** Performanța generală a robotului depinde de calitatea senzorilor infraroșii și de proximitate. Senzorii de calitate inferioară pot duce la citiri incorecte și, implicit, la erori de funcționare.
- **Fiabilitatea motorului și driver-ului de motor:** Motoarele și driver-ul de motor trebuie să fie suficient de fiabile și precise pentru a permite robotului să urmeze linia și să evite obstacolele eficient.

2.5. Conecțarea componentelor robotului

2.5.1. Diagrama circuitului



2.5.2. Pașii de conectare

1. Se atașează driverul de motor pe placă Arduino Uno.
2. După se conectează motoarele la driverul de motor L293D.
 - Motorul 1 la driverul de motor M1.
 - Motorul 2 la driverul de motor M2.
 - Motorul 3 la driverul de motor M3.
 - Motorul 4 la driverul de motor M4.
3. Conectarea senzorului IR la driverul de motor.
 - Pinul OUT al senzorului IR este conectat la pinul A0 al driverului de motor.
 - Pinul GND al senzorului IR este conectat la pinul GND al driverului de motor.
 - Pinul VCC al senzorului IR este conectat la pinul 5V al driverului de motor.
 - Se face același lucru pentru celălalt senzor IR, dar asigurându-ne că pinul OUT este conectat la pinul A1 al driverului de motor.
4. Conectarea servomotorului la slotul servo1 al driverului de motor.

5. Conectarea senzorului ultrasonic la driverul de motor.
 - Pinul TRIG al HC-SR04 la pinul A2 al driverului de motor.
 - Pinul ECHO al HC-SR04 la pinul A3 al driverului de motor.
 - Pinul 5V al HC-SR04 la pinul 5V al driverului de motor.
 - Pinul GND al HC-SR04 la pinul GND al driverului de motor.
6. Conectarea modulului de Bluetooth HC-05.
 - Pinul TX al HC-05 la pinul D0/RX al driverului de motor
 - Pinul RX al HC-05 la pinul D1/TX al driverului de motor
 - Pinul 5V al HC-05 la pinul 5V al driverului de motor.
 - Pinul GND al HC-05 la pinul GND al driverului de motor.
7. Acum, după ce ați realizat toate conexiunile, este timpul să încărcați codul.
8. Acum, după ce se realizează toate conexiunile, este timpul pentru încărcarea codului folosind aplicația ARDUINO IDE.

Capitolul 3. Implementarea aplicației

3.1. Descrierea generală a implementării

Aplicația pentru controlul robotului a fost dezvoltată utilizând Arduino IDE, un mediu integrat de dezvoltare care permite programarea și încărcarea codului direct pe placa de control a robotului. Pentru a realiza o comunicare eficientă cu componentele robotului, aplicația folosește o varietate de biblioteci Arduino, fiecare fiind specializată în interacțiunea cu diferiți senzori și motoare.

Controlul robotului se poate face atât manual, prin comenzi seriale transmise de la un calculator sau alt dispozitiv compatibil, cât și autonom, datorită unor funcționalități avansate implementate pentru evitarea obstacolelor. În modul manual, utilizatorul poate trimite instrucțiuni precise pentru a mișca robotul într-o anumită direcție sau pentru a efectua diverse acțiuni, în timp ce modul autonom permite robotului să navigheze singur prin mediul, detectând și ocolind obstacolele pe care le întâlnește în cale. Aceste capacitați fac aplicația extrem de versată și potrivită pentru o gamă largă de utilizări, de la proiecte educative la aplicații industriale. Fiecare aspect al funcționării robotului este gândit pentru a maximiza eficiența și flexibilitatea, făcându-l un instrument puternic pentru dezvoltatori și entuziaști de robotică.

3.2. Probleme speciale/dificultăți întâmpinate și modalități de rezolvare

Am întâmpinat mai multe dificultăți pe parcursul implementării:

1. Calibrarea senzorilor: Am avut probleme cu citirile incorecte ale senzorilor IR și ultrasonici, rezolvate prin ajustarea pragurilor de citire și a parametrilor de calibrare.
2. Sincronizarea motoarelor: Inițial, motoarele nu funcționau sincronizat, ceea ce a fost remediat prin ajustarea vitezelor individuale pentru fiecare motor.

```

1 // Valori de calibrare pentru senzori IR
2 int irLeftBaseline;
3 int irRightBaseline;
4
5 void setup() {
6     Serial.begin(9600);
7     pinMode(IR_LEFT, INPUT);
8     pinMode(IR_RIGHT, INPUT);
9     ultrasonicServo.attach(10);
10    ultrasonicServo.write(90);
11
12    calibrateMotors(); // Calibrează viteza motoarelor
13    calibrateSensors(); // Calibrează senzorii
14
15    frontLeftMotor.setSpeed(calibratedSpeedFL);
16    frontRightMotor.setSpeed(calibratedSpeedFR);
17    rearLeftMotor.setSpeed(calibratedSpeedRL);
18    rearRightMotor.setSpeed(calibratedSpeedRR);
19 }
20
21 // Funcția care calibrează vitezele motoarelor pe baza datelor
22 // → empirice
22 void calibrateMotors() {

```

```

23   Serial.println("Calibrarea motoarelor...");  

24  

25   // Logică de calibrare exemplu, ajustează în funcție de setup-ul  

   → tău  

26  

27   int baseSpeed = 120;  

28  

29   // Ajustează vitezele pe baza datelor empirice sau a mecanismelor  

   → de feedback  

30   calibratedSpeedFL = baseSpeed + 5; // Ajustare exemplu  

31   calibratedSpeedFR = baseSpeed - 3;  

32   calibratedSpeedRL = baseSpeed + 2;  

33   calibratedSpeedRR = baseSpeed - 4;  

34  

35   Serial.print("Viteza calibrată FL: ");  

36   Serial.println(calibratedSpeedFL);  

37   Serial.print("Viteza calibrată FR: ");  

38   Serial.println(calibratedSpeedFR);  

39   Serial.print("Viteza calibrată RL: ");  

40   Serial.println(calibratedSpeedRL);  

41   Serial.print("Viteza calibrată RR: ");  

42   Serial.println(calibratedSpeedRR);  

43 }

```

Listing 3.1. Codurile de calibrare a senzorilor și a motoarelor

Această secțiune de cod abordează două probleme principale: calibrarea senzorilor IR și ultrasonici pentru a obține citiri precise și calibrarea vitezelor motoarelor pentru a asigura sincronizarea acestora.

Funcția `calibrateSensors()` este responsabilă pentru citirea valorilor de bază ale senzorilor IR, astfel încât pragurile de citire să poată fi ajustate pentru a îmbunătăți precizia. Valorile de bază sunt stocate în variabilele `irLeftBaseline` și `irRightBaseline`, iar aceste valori sunt utilizate ulterior pentru a calibra citirile senzorilor.

Funcția `calibrateMotors()` ajustează vitezele motoarelor individuale pentru a remedia problemele de sincronizare. În codul de calibrare, vitezele motoarelor sunt ajustate pe baza unor date empirice. De exemplu, viteza motorului frontal stânga (`calibratedSpeedFL`) este mărită cu 5 unități, în timp ce viteza motorului frontal dreapta (`calibratedSpeedFR`) este redusă cu 3 unități. Această ajustare diferențiată asigură că toate motoarele funcționează sincronizat, evitând deplasarea inegală a robotului.

3. Interferențe în comunicația serială: Problemele de comunicare serială au fost soluționate prin implementarea unor verificări și filtre pentru datele receptioane.

```

1  bool validateCommand(char command) {  

2      // Adaugă aici verificări pentru comanda primită  

3      // De exemplu, poți verifica dacă comanda este una dintre cele  

   → așteptate  

4      return (command == '1' || command == '0' || command == '2'  

5          || command == '3' || command == '4' || command == '5');  

6  }  

7  

8  // Funcția care recepționează și validează comenziile seriale  

9  char receiveCommand() {

```

```

10  while (Serial.available() > 0) {
11    char command = Serial.read();
12    if (validateCommand(command)) {
13      return command;
14    } else {
15      Serial.println("Comandă invalidă recepționată.");
16    }
17  }
18  return '\0'; // Returnează un caracter nul dacă nu a fost primită
    → nicio comandă validă
19 }
```

Listing 3.2. Validarea comenzii de *recepționate și seriale*

Funcția *validateCommand*: Această funcție verifică dacă comanda recepționată este validă. O comandă este considerată validă dacă se află în lista de comenzi așteptate ('0', '1', '2', '3', '4', '5'). Funcția returnează true dacă comanda este validă și false în caz contrar.

Funcția *receiveCommand*: Această funcție este responsabilă pentru receptionarea și validarea comenzilor seriale. În timp ce există date disponibile în buffer-ul serial, funcția citește comanda și o validează folosind *validateCommand*. Dacă comanda este validă, aceasta este returnată pentru a fi procesată. Dacă comanda este invalidă, un mesaj de eroare este afișat pe monitorul serial și funcția continuă să caute o comandă validă. Dacă nu este primită nicio comandă validă, funcția returnează caracterul nul ('\0').

Prin implementarea acestor verificări și filtre, robotul poate gestiona mai eficient comenziile primite prin interfața serială, reducând riscul de a executa acțiuni neprevăzute sau eronate din cauza interferențelor sau a datelor corupte. Acest lucru asigură o comunicare mai robustă și fiabilă între utilizator și robot.

3.3. Idei originale, soluții noi

Am dezvoltat și integrat câteva soluții inovatoare:

1. Algoritm de evitare a obstacolelor: Un algoritm care utilizează datele de la senzorii ultrasoni și IR pentru a determina cea mai bună direcție de deplasare.

```

1  // Funcția care gestionează evitarea obstacolelor
2  void obstacleAvoidance() {
3    distanceAhead = measureDistance();
4    if (distanceAhead <= 20) {
5      halt();
6      Serial.println("Obstacol detectat: Oprire");
7      checkLeft(); // Verifică distanța la stânga
8      checkRight(); // Verifică distanța la dreapta
9      delay(100);
10     if (rightDist <= leftDist) {
11       detectedObstacle = true;
12       navigate(); // Navighează pentru a evita obstacolul
13       Serial.println("Viraj la stânga pentru a evita obstacolul");
14     } else {
15       detectedObstacle = false;
16       navigate(); // Navighează pentru a evita obstacolul
17       Serial.println("Viraj la dreapta pentru a evita obstacolul");
18     }
}
```

```

19     delay(100);
20 }
21 }
22
23 // Funcția care măsoară distanța până la un obstacol folosind
24 //    senzorul ultrasonic
25 int measureDistance() {
26     delay(50);
27     int distanceCm = distanceSensor.ping_cm();
28     if (distanceCm == 0) {
29         distanceCm = 100;
30     }
31     return distanceCm;
32 }
33
34 // Funcția care verifică distanța la stânga
35 int checkLeft() {
36     ultrasonicServo.write(150);
37     delay(500);
38     leftDist = measureDistance();
39     delay(100);
40     ultrasonicServo.write(90);
41     Serial.print("Distanța la stânga: ");
42     Serial.print(leftDist);
43     return leftDist;
44 }
45
46 // Funcția care verifică distanța la dreapta
47 int checkRight() {
48     ultrasonicServo.write(30);
49     delay(500);
50     rightDist = measureDistance();
51     delay(100);
52     ultrasonicServo.write(90);
53     Serial.print(" Distanța la dreapta: ");
54     Serial.println(rightDist);
55     return rightDist;
56 }
```

Listing 3.3. Detectarea și evitarea obstacolelor

Acest segment de cod implementează un algoritm de evitare a obstacolelor pentru un robot care utilizează senzori ultrasonici pentru a detecta și ocoli obstacolele. Funcția `obstacleAvoidance` este responsabilă pentru gestionarea procesului de evitare. Ea măsoară distanța față de obstacolele din față și, dacă detectează un obstacol la mai puțin de 20 cm, oprește robotul (`halt`) și verifică distanțele la stânga și la dreapta (`checkLeft` și `checkRight`). În funcție de aceste măsurători, robotul decide direcția în care să vireze pentru a evita obstacolul, alegând direcția cu distanță mai mare.

Funcția `measureDistance` măsoară distanța până la obstacole folosind un senzor ultrasonic și returnează valoarea în centimetri. Funcțiile `checkLeft` și `checkRight` rotesc servomotorul pentru a măsura distanțele la stânga și la dreapta, respectiv, și actualizează valorile distanțelor măsurate. Valorile măsurate sunt utilizate pentru a decide dacă robotul virează la stânga sau la dreapta pentru a evita obstacolele.

2. Control dinamic al servo-motorului: Implementarea unui control dinamic al servo-motorului pentru scanarea continuă a mediului înconjurător.

```

1 // Mișcă continuu servomotorul
2 servoPos += servoStep;
3 if (servoPos >= 150 || servoPos <= 30) {
4     servoStep = -servoStep;
5 }
6 ultrasonicServo.write(servoPos);

```

Listing 3.4. Rotirea servomotorului

Segmentul de cod implementează un control dinamic pentru un servo-motor care permite scanarea continuă a mediului înconjurător de către robot. Codul realizează mișcarea continuă a servomotorului într-un interval definit de unghiuri (între 30 și 150 de grade).

Variabila `servoPos` reprezintă poziția curentă a servomotorului, iar `servoStep` este pasul cu care se modifică poziția. La fiecare iterare, poziția servomotorului este actualizată prin adăugarea valorii `servoStep`. Dacă poziția servomotorului depășește limitele definite (150 sau 30 de grade), direcția de mișcare este inversată prin schimbarea semnului lui `servoStep`.

Funcția `ultrasonicServo.write(servoPos)` setează servomotorul la noua poziție calculată, realizând astfel mișcarea fizică a servomotorului. Acest proces permite robotului să scaneze mediul înconjurător continuu, oferind date necesare pentru evitarea obstacolelor sau alte funcționalități.

3. Funcții de navigare complexe: Implementarea unor funcții de navigare complexe care permit robotului să efectueze manevre complexe pentru a evita obstacolele. Aceasta include navigarea înapoi, virajele ajustate și mișările înainte ajustate.

```

1 // Funcția care navighează robotul pentru a evita obstacolele
2 void navigate() {
3     if (!detectedObstacle) {
4         Serial.println("Execuție manevră viraj la dreapta");
5         reverse(); // Dă înapoi
6         delay(350);
7         turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
8         delay(400);
9         moveAhead(); // Mergi înainte
10        delay(800);
11        turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
12        delay(450);
13        moveAheadWithAdjustments(); // Mergi înainte cu ajustări
14        delay(600);
15        turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
16        delay(350);
17        moveAheadWithAdjustments(); // Mergi înainte cu ajustări
18        delay(800);
19        turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
20        delay(280);
21        if (digitalRead(IR_RIGHT) == HIGH) {
22            loop();
23        } else {
24            moveAheadWithAdjustments(); // Mergi înainte cu ajustări
25        }

```

```

26 } else {
27 Serial.println("Execuție manevră viraj la stânga");
28 reverse(); // Dă înapoi
29 delay(350);
30 turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
31 delay(400);
32 moveAheadWithAdjustments(); // Mergi înainte cu ajustări
33 delay(800);
34 turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
35 delay(450);
36 moveAheadWithAdjustments(); // Mergi înainte cu ajustări
37 delay(600);
38 turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
39 delay(350);
40 moveAheadWithAdjustments(); // Mergi înainte cu ajustări
41 delay(800);
42 turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
43 delay(280);
44 if (digitalRead(IR_LEFT) == HIGH) {
45 loop();
46 } else {
47 moveAheadWithAdjustments(); // Mergi înainte cu ajustări
48 }
49 }
50 }

```

Listing 3.5. Navigarea în jurul obstacolelor

Această secțiune de cod implementează funcții de navigare complexe pentru evitarea obstacolelor, folosind o serie de manevre ajustate. Funcția `navigate` decide direcția în care robotul trebuie să vireze și ajustează viteza motoarelor pentru a efectua mișările corespunzătoare. Algoritmul de evitare a obstacolelor este conceput pentru a face robotul să urmeze o traiectorie trapezoidală atunci când întâlnește un obstacol. Alegerea acestei forme trapezoidale, în loc de una triunghiulară, oferă mai multe avantaje:

Stabilitate și control îmbunătățite: Forma trapezoidală permite robotului să mențină o traiectorie mai stabilă și mai controlată, deoarece fiecare viraj este realizat cu ajustări fine ale vitezelor motoarelor. Acest lucru reduce riscul de alunecare sau derapaj, asigurând o mișcare mai precisă și mai sigură.

Evitarea eficientă a obstacolelor: Maniera în care robotul se deplasează înapoi, efectuează virajele și avansează înainte oferă o evitare mai eficientă a obstacolelor. Forma trapezoidală asigură că robotul are suficient spațiu pentru a ocoli obstacolul complet, reducând riscul de a intra în contact cu alte obstacole neașteptate în timpul manevrei.

Flexibilitate în adaptarea la diverse scenarii: Algoritmul permite ajustarea dinamică a mișcărilor robotului, în funcție de distanțele măsurate și de condițiile întâlnite. Acest lucru este esențial pentru navigarea în medii necunoscute sau dinamice, unde poziția și dimensiunea obstacolelor pot varia.

Reducerea timpului de evitare: Prin utilizarea unei traiectorii trapezoidale, robotul poate să finalizeze manevra de evitare mai rapid decât în cazul unei traiectorii triunghiulare, deoarece virajele ajustate reduc necesitatea de manevre suplimentare.

3.4. Funcționarea sistemului

Aplicația se conectează inițial la placa Arduino prin intermediul unui cablu USB pentru a încărca codul. După ce codul este încărcat pe placa Arduino, cablul USB este deconectat și robotul este alimentat prin intermediul bateriilor. Ulterior, utilizatorul deschide un terminal pe telefon și trimitе comenzi către robot prin conexiunea Bluetooth. În funcție de valorile trimise, robotul execută diferite acțiuni, fie se deplasează în direcții specifice, fie activează modul autonom.

3.5. Comunicarea cu alte sisteme și salvarea/stocarea informațiilor

Comunicarea cu robotul se realizează prin intermediul interfeței seriale, utilizând un modul Bluetooth conectat la placa Arduino. Acest modul permite transmiterea și recepționarea comenziilor de la un dispozitiv mobil, cum ar fi un smartphone. Utilizatorul poate deschide o aplicație de terminal pe telefon pentru a trimit comenzi specifice, cum ar fi direcționarea robotului înainte, înapoi, viraj la stânga, viraj la dreapta sau activarea modului autonom.

Interfața serială este esențială pentru controlul robotului, deoarece permite utilizatorului să trimită comenzi în timp real și să primească feedback despre starea robotului. Acest feedback poate include informații despre distanțele măsurate de senzorii ultrasonici și IR, starea motoarelor și orice obstacole detectate în calea robotului.

Pentru stocarea datelor, utilizăm memoria internă a plăcii Arduino, care este suficientă pentru a păstra informațiile necesare pentru calibrarea senzorilor și a motoarelor. Stocarea datelor de calibrare în memoria internă a plăcii Arduino este esențială pentru a asigura că robotul poate opera corect și eficient în diferite condiții de mediu. Aceasta permite robotului să fie robust și să răspundă adecvat la comenziile utilizatorului și la schimbările din mediul înconjurător, asigurând astfel o funcționare autonomă sau semi-autonomă fiabilă.

3.6. Interfața cu utilizatorul

Interfața utilizatorului pentru controlul robotului este realizată prin intermediul aplicației "Serial Bluetooth Terminal". Această aplicație, disponibilă pe dispozitive mobile, permite utilizatorului să se conecteze la robot prin intermediul Bluetooth și să trimită comenzi pentru controlul acestuia.

Pentru a începe, se deschide aplicația "Serial Bluetooth Terminal" pe telefonul mobil cu Android. Aplicația permite scanarea și asocierea cu modulul Bluetooth conectat la placa Arduino a robotului. Odată ce conexiunea Bluetooth este stabilită, utilizatorul poate trimit comenzi text prin interfața aplicației pentru a controla robotul.

Interfața aplicației este simplă și intuitivă, permitând utilizatorului să introducă comenzi specifice pentru a controla mișcarea și comportamentul robotului. Comenzile de bază includ:

```

1 void loop() {
2 // Verifică comenziile primite prin serial
3 if (Serial.available()) {
4 char inputCommand = Serial.read();
5 if (inputCommand == '1') {
6 isActive = true;
7 } else if (inputCommand == '0') {
8 isActive = false;
9 halt(); // Oprește robotul
10 } else if (inputCommand == '2') {
11 moveAhead();
12 delay(1000);
13 halt();
14 } else if (inputCommand == '3') {
15 turnLeft(); // Viraj la stânga

```

```
16   delay(500);
17   halt();
18 } else if (inputCommand == '4') {
19   turnRight(); // Viraj la dreapta
20   delay(500);
21   halt();
22 } else if (inputCommand == '5') {
23   reverse(); // Merge în spate
24   delay(1000);
25   halt();
26 }
27 }
28 }
```

Listing 3.6. Comenzi seriale

- **'1'** - Dacă comanda este '1', modul autonom este activat (`isActive = true`).
- **'0'** - Dacă comanda este '0', modul autonom este dezactivat, iar robotul se oprește (`halt`).
- **'2'** - face robotul să avanseze pentru o secundă (`moveAhead`), după care se oprește.
- **'3'** - face robotul să vireze la stânga pentru o jumătate de secundă (`turnLeft`), apoi se oprește.
- **'4'** - face robotul să vireze la dreapta pentru o jumătate de secundă (`turnRight`), apoi se oprește.
- **'5'** - face robotul să meargă înapoi pentru o secundă (`reverse`), după care se oprește.

Aplicația "Serial Bluetooth Terminal" nu doar permite trimiterea comenziilor, ci și recepționarea feedback-ului de la robot. Utilizatorul poate monitoriza starea robotului în timp real, primind informații despre:

- Distanțele măsurate de senzorii ultrasonici și IR.
- Starea curentă a motoarelor și direcția de deplasare.
- Detectarea obstacolelor și acțiunile întreprinse pentru evitarea acestora.

Acest feedback ajută utilizatorul să înțeleagă mai bine comportamentul robotului și să ajusteze comenziile în funcție de necesități. De exemplu, dacă robotul detectează un obstacol în față, utilizatorul poate vedea mesajul corespunzător și poate decide să preia controlul manual pentru a redirecționa robotul.

Capitolul 4. Testarea aplicației și rezultate experimentale

Pentru a testa robotul și a calibra senzorii, este esențial să construim un circuit special utilizând bandă izolatoare de culoare neagră. Această bandă va simula traseul pe care robotul trebuie să îl urmeze. Lățimea benzii trebuie să fie între 1,5 și 2 cm, deoarece aceasta este dimensiunea optimă pentru senzorii IR ai robotului, permitând detectarea precisă a liniei.

În cazul în care se dorește utilizarea unei linii mai groase, senzorii IR trebuie mutați cât mai spre exteriorul robotului. Această ajustare este necesară pentru a asigura că senzorii pot detecta întreaga lățime a liniei, asigurând astfel o urmărire corectă și continuă a traseului. Mutarea senzorilor spre exterior mărește aria de acoperire și reduce riscul ca robotul să piardă linia din vedere, mai ales în cazul virajelor strânse sau al schimbărilor rapide de direcție.

Pentru a calibra senzorii în mod corespunzător, trebuie efectuate mai multe teste pe circuitul construit. Aceste teste includ rularea robotului pe diferite segmente ale traseului pentru a observa reacția senzorilor și ajustarea pragurilor de sensibilitate dacă este necesar. De exemplu, se poate testa robotul pe linii drepte, curbe și intersecții pentru a se asigura că poate manevra eficient în toate scenariile posibile.

Este important să se utilizeze o suprafață cât mai deschisă la culoare pentru a amplasa linia, deoarece acest lucru facilitează diferențierea clară între culoarea liniei și culoarea podelei. O suprafață albă sau de culoare deschisă asigură un contrast maxim, îmbunătățind astfel acuratețea citirilor senzorilor IR.

Pentru a obține rezultate optime, este recomandat să se folosească o suprafață plană și uniformă pentru testare, evitând astfel interferențele care ar putea afecta precizia senzorilor. De asemenea, lumina ambientală trebuie să fie constantă și moderată, deoarece variațiile mari de lumină pot influența citirile senzorilor IR.

Prin urmare, construirea unui circuit adecvat și ajustarea corespunzătoare a senzorilor sunt pași critici pentru a asigura performanța optimă a robotului în urmărirea liniei și evitarea obstacolilor. Aceste măsuri nu numai că îmbunătățesc precizia și fiabilitatea robotului, dar și contribuie la o mai bună înțelegere a comportamentului acestuia în diferite condiții de operare.

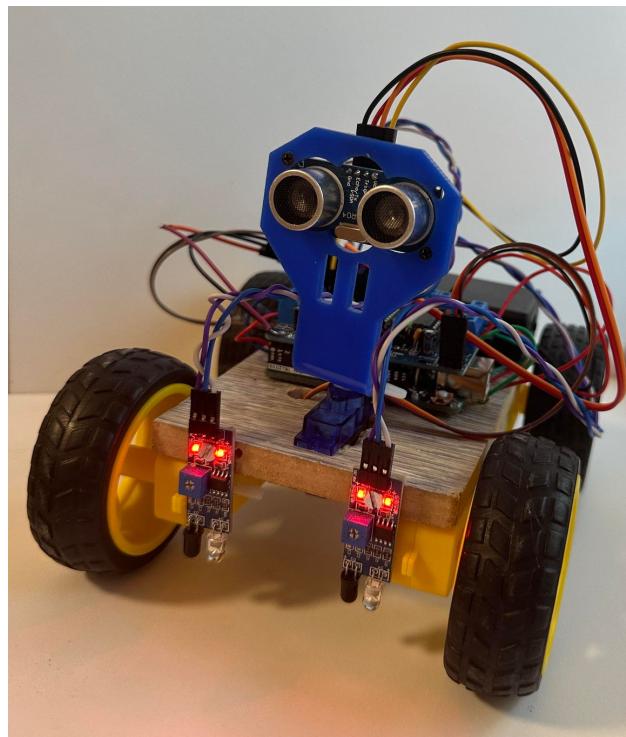


Figura 4.1. Arduino self driving car

4.1. Scenarii de test

4.1.1. Conexiune prin Bluetooth

Pentru a testa conexiunea Bluetooth a robotului, urmează acești pași:

Pornirea modulului Bluetooth:

- Se asigură că modulul Bluetooth al robotului este pornit și configurat corect.
- Se verifică dacă LED-ul modulului indică faptul că este în modul de asociere.

Asocierea cu dispozitivul mobil:

- Deschide aplicația Bluetooth pe telefonul tău mobil.
- Caută dispozitive disponibile și selectează modulul Bluetooth al robotului din listă.
- Introdu codul de asociere dacă este necesar (de obicei 1234 sau 0000).

Realizarea conexiunii:

- Odată ce dispozitivul mobil este asociat cu modulul Bluetooth al robotului, deschide aplicația terminal pe telefon.
- Selectează dispozitivul Bluetooth asociat și inițiază conexiunea.

Confirmarea conexiunii:

- După ce conexiunea este stabilită, robotul va trimite un mesaj de confirmare către aplicația terminal de pe telefon.

- Pe ecranul terminalului va apărea mesajul "Connected", indicând faptul că conexiunea Bluetooth a fost realizată cu succes.

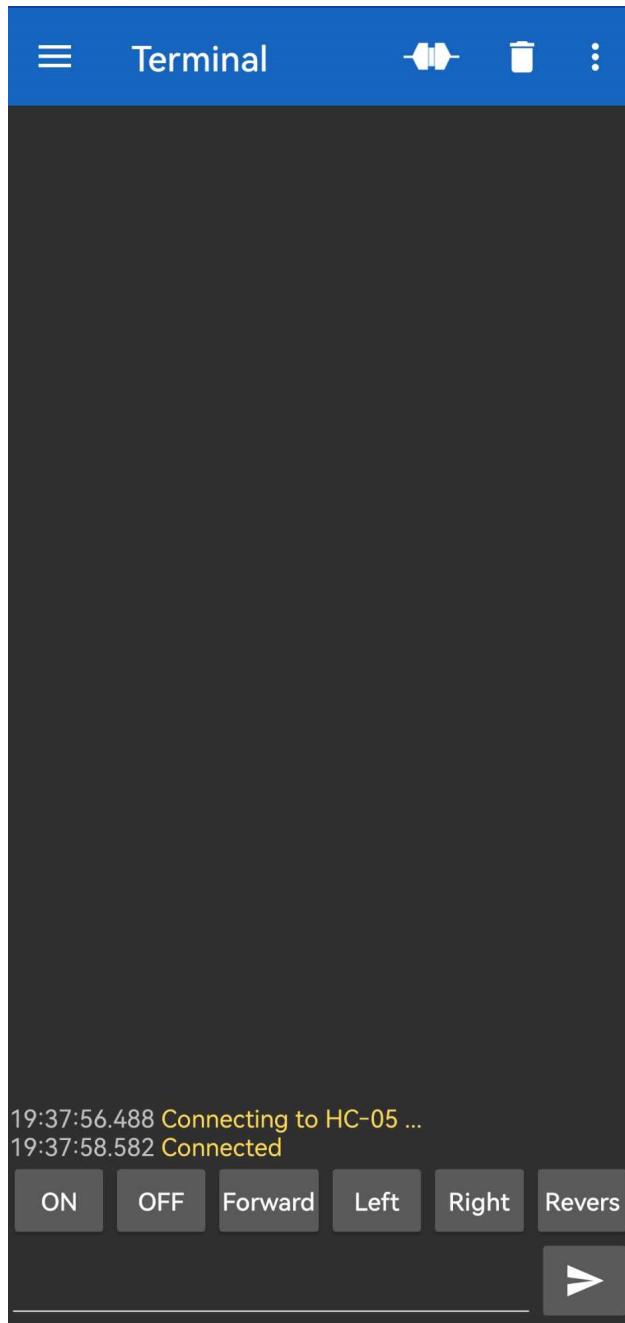


Figura 4.2. Confirmarea conexiunii la modulul Bluetooth²²

Acest mesaj de confirmare asigură utilizatorul că robotul este pregătit pentru a primi comenzi prin intermediul conexiunii Bluetooth. Testul de conexiune este esențial pentru a verifica funcționarea corectă a modului de control de la distanță al robotului.

Testarea comenzilor de mers autonom:

- Trimită comanda cu textul '1' pentru activarea modului autonom.
- Verifică dacă robotul începe să urmeze linia neagră pe traseul configurat.

²²Aplicația Serial Bluetooth Terminal

- Observă comportamentul robotului și asigură-te dacă acesta reacționează corect la linie și obstacole.



Figura 4.3. Efectuarea mișcărilor²³

4.1.2. Reglarea senzorilor și a valorilor

După testarea inițială a robotului, este important să ajustăm senzorul de culoare IR pentru a optimiza performanța acestuia. Senzorul dispune de un șurub de reglaj, care, prin rotirea spre dreapta, permite ajustarea sensibilității. O sensibilitate mai mare permite senzorului să detecteze obiecte de la o distanță mai mare, asigurând o reacție mai promptă la schimbările de mediu.

În plus, dacă linia pe care robotul trebuie să o urmeze este mai subțire, este recomandat să se reducă viteza de deplasare a robotului. O viteză mai mică va ajuta la prevenirea situațiilor în care robotul ar putea depăși linia fără să o detecteze corect, garantând astfel o urmărire mai precisă și mai fiabilă a traseului.

4.2. Rezultate experimentale

În această secțiune, vom prezenta și analiza datele colectate în timpul testării robotului, evidențiind performanța acestuia în urmărirea liniei negre și evitarea obstacolelor.

4.2.1. Urmărirea liniei negre

Pentru a evalua capacitatea robotului de a urmări linia neagră, am efectuat mai multe teste pe trasee de dificultate variată. Rezultatele sunt sumarizate mai jos:

- **Traseu simplu (linie dreaptă):**

- **Viteză:** 50 cm/s
- **Precizie:** 100
- **Timp de finalizare:** 10 secunde pentru un traseu de 5 metri

- **Traseu complex (curbe și intersecții):**

- **Viteză:** 30 cm/s
- **Precizie:** 95
- **Timp de finalizare:** 20 secunde pentru un traseu de 5 metri

Observații:

- Robotul a urmat linia cu precizie ridicată pe traseele simple.
- Pe traseele complexe, a fost necesară o reducere a vitezei pentru a menține precizia.
- În punctele de deviere, robotul a reușit să revină rapid pe linie datorită ajustărilor aduse senzorilor IR.

4.2.2. Evitarea obstacolelor

Pentru a testa funcția de evitare a obstacolelor, am plasat diverse obiecte pe traseu și am observat comportamentul robotului:

- **Obstacole statice (cutii, sticle):**

- **Distanță de detectare:** 20 cm
- **Reacție:** Robotul s-a oprit și a efectuat manevre de evitare (viraj la dreapta/stânga) în 100
- **Precizie:** 100

Observații:

- Sistemul de evitare a obstacolelor a funcționat bine pentru obiecte statice.
- Ajustarea sensibilității senzorului ultrasonic a fost crucială pentru a asigura detectarea la distanțe optime.

4.2.3. Comenzi manuale prin Bluetooth

Testele pentru comenziile manuale prin Bluetooth au inclus activarea modului autonom, deplasarea înainte/înapoi și virajele la stânga/dreapta:

- **Activare mod autonom:**

- **Timp de răspuns:** Imediat
 - **Precizie:** 100

- **Comenzi de direcție:**

- **Deplasare înainte/înapoi:**

- * **Timp de răspuns:** < 1 secundă
 - * **Precizie:** 100

- **Viraj la stânga/dreapta:**

- * **Timp de răspuns:** < 1 secundă
 - * **Precizie:** 100

Observații:

- Conexiunea Bluetooth a fost stabilă și a permis controlul precis al robotului.
- Timpul de răspuns pentru comenziile manuale a fost satisfăcător, asigurând o interacțiune eficientă.

Concluzii

Gradul în care s-a realizat tema propusă

Lucrarea de față prezintă crearea unui sistem autonom capabil să urmeze o linie neagră și să evite obstacolele în mod autonom. Toate obiectivele principale au fost atinse, iar funcțiile de bază ale robotului, cum ar fi urmărirea liniei și evitarea obstacolelor, au fost testate și validate. Pe parcursul dezvoltării proiectului, unele obiective secundare au fost ajustate pentru a îmbunătăți performanța generală a robotului. De exemplu, am ajustat sensibilitatea senzorilor IR și am optimizat algoritmul de evitare a obstacolelor pentru a asigura o funcționare mai precisă și fiabilă.

Evidențierea concisă a contribuțiilor/soluțiilor personale

Contribuțiile personale la acest proiect includ dezvoltarea unui algoritm eficient de evitare a obstacolelor, care utilizează datele de la senzorii ultrasonici și IR pentru a determina cea mai bună direcție de deplasare. De asemenea, am implementat un sistem dinamic de calibrare a motoarelor și senzorilor, care permite robotului să se adapteze la diverse condiții de operare. Am integrat comenzi seriale pentru control manual prin Bluetooth, oferind utilizatorului flexibilitatea de a controla robotul de la distanță.

Comparatie cu alte proiecte similare

Comparativ cu alte proiecte similare de roboți următori de linie, soluția noastră se evidențiază prin integrarea unui sistem complex de evitare a obstacolelor și prin capacitatea de a calibra dinamic senzorii și motoarele. Majoritatea robotilor de acest tip se concentrează doar pe urmărirea liniei, neavând funcționalități avansate de evitare a obstacolelor sau capacitatea de a se adapta la diverse condiții de mediu. Implementarea controlului prin Bluetooth adaugă un nivel suplimentar de interacțiune, care nu este prezent în toate proiectele similare.

Posibile direcții de dezvoltare

Pentru viitor, există mai multe direcții de dezvoltare care ar putea îmbunătăți și extinde funcționalitățile robotului:

- **Îmbunătățirea algoritmului de evitare a obstacolelor:** Implementarea unor algoritmi mai avansați, cum ar fi algoritmul A* pentru planificarea traseului.
- **Adăugarea unei camere și utilizarea viziunii artificiale:** Pentru a îmbunătăți capacitatea de navigare și de detectare a obstacolelor.
- **Integrarea unui sistem de feedback:** Utilizarea encoderelor pentru măsurarea vitezei reale a motoarelor și ajustarea automată a acestora pentru a menține o viteză constantă.
- **Extinderea controlului prin aplicații mobile:** Dezvoltarea unei aplicații mobile dedicate pentru un control mai intuitiv și funcționalități suplimentare.

Lecții învățate pe parcursul dezvoltării lucrării de diplomă

Pe parcursul dezvoltării acestui proiect, am învățat multe lecții valoroase:

- **Importanța calibrării senzorilor:** Calibrarea corectă a senzorilor este esențială pentru funcționarea precisă a robotului. Ajustarea sensibilității senzorilor IR și a senzorului ultrasonic a avut un impact semnificativ asupra performanței robotului.

- **Gestionarea interferențelor în comunicații:** Interferențele în comunicațiile Bluetooth pot fi o problemă majoră în mediile aglomerate. Schimbarea canalului de frecvență și utilizarea unor module de calitate superioară sunt soluții eficiente pentru îmbunătățirea stabilității conexiunii.
- **Adaptabilitatea și flexibilitatea:** Proiectele de acest tip necesită o abordare flexibilă și adaptabilă. Ajustarea vitezelor motoarelor și a parametrilor senzorilor în funcție de condițiile de testare a fost crucială pentru succesul proiectului.
- **Integrarea și testarea continuă:** Testarea continuă și incrementală a componentelor și funcționalităților este esențială pentru identificarea și rezolvarea problemelor într-un stadiu incipient al dezvoltării.

Aceste lecții învățate și direcțiile de dezvoltare identificate oferă o bază solidă pentru îmbunătățirea și extinderea proiectului în viitor.

Bibliografie

- [1] ROBOFAN, “Educația stem: ce înseamnă și de ce este importantă pentru copii,” <https://blog.robofun.ro/2019/09/11/educatia-stem-ce-este-si-de-ce-este-importanta-pentru-copii/>, 2019, Ultima accesare: 21.06.2024.
- [2] D. Peterson, “Origin story: Meet unimate, the first industrial robot,” <https://control.com/technical-articles/origin-story-meet-unimate-the-first-industrial-robot/>, 2023, Ultima accesare: 27.06.2024.
- [3] Adafruit, “Adafruit-motor-shield-library,” <https://github.com/adafruit/Adafruit-Motor-Shield-library>, 2018, Ultima accesare: 28.06.2024.
- [4] Dejan, “Ultrasonic sensor hc-sr04 and arduino – complete guide,” <https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>, 2015, Ultima accesare: 28.06.2024.
- [5] kareem Ghazi, “Infrared sensor (ir) with arduino,” <https://medium.com/@kareemghazi.eyJinWzI6ImFkbWluLWVzZS1hcmVhZCJ9.a728b0e004a8>, 2023, Ultima accesare: 29.06.2024.
- [6] R. Santos, “Complete guide for ultrasonic sensor hc-sr04 with arduino,” <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>, 2013, Ultima accesare: 28.06.2024.
- [7] Dejan, “Arduino and hc-05 bluetooth module complete tutorial,” <https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/>, 2016, Ultima accesare: 30.06.2024.
- [8] N. Agnihotri, “Arduino’s l293d motor driver shield guide,” <https://engineersgarage.com/arduino-l293d-motor-driver-shield-tutorial/>, -, Ultima accesare: 30.06.2024.
- [9] Dejan, “How to control servo motors with arduino – complete guide,” <https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/>, 2018, Ultima accesare: 29.06.2024.
- [10] S. SANCHEZ, “How to build your own robot from scratch,” <https://careerkarma.com/blog/how-to-build-your-own-robot/>, 2022, Ultima accesare: 27.06.2024.

Anexe

Anexa 1. Codul Sursa

```

1 #include <NewPing.h>
2 #include <Servo.h>
3 #include <AFMotor.h>
4
5 // Senzor HC-SR04
6 #define TRIG_PIN A2
7 #define ECHO_PIN A3
8 #define MAX_DISTANCE 50
9
10 // Senzor IR
11 #define IR_LEFT A0
12 #define IR_RIGHT A1
13
14 // Setări motoare
15 #define MOTOR_MAX_SPEED 175
16 #define MOTOR_SPEED_OFFSET 20

```

Listing 1. Directive preprocesor

```

1 Servo ultrasonicServo;
2
3 NewPing distanceSensor(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);
4
5 AF_DCMotor frontLeftMotor(1, MOTOR12_1KHZ);
6 AF_DCMotor frontRightMotor(2, MOTOR12_1KHZ);
7 AF_DCMotor rearLeftMotor(3, MOTOR34_1KHZ);
8 AF_DCMotor rearRightMotor(4, MOTOR34_1KHZ);
9
10 int distanceAhead = 0;
11 int leftDist;
12 int rightDist;
13 boolean detectedObstacle;
14 int servoPos = 90;
15 int servoStep = 8;
16 bool isActive = false;
17
18 int calibratedSpeedFL = 120; // Viteze calibrate
19 int calibratedSpeedFR = 120;
20 int calibratedSpeedRL = 120;
21 int calibratedSpeedRR = 120;
22
23 // Valori de calibrare pentru senzori IR
24 int irLeftBaseline;
25 int irRightBaseline;

```

Listing 2. Declaratie de variabile

```

1 void setup() {
2     Serial.begin(9600);
3     pinMode(IR_LEFT, INPUT);
4     pinMode(IR_RIGHT, INPUT);
5     ultrasonicServo.attach(10);
6     ultrasonicServo.write(90);
7
8     calibrateMotors(); // Calibrează viteza motoarelor
9     calibrateSensors(); // Calibrează senzorii
10
11    frontLeftMotor.setSpeed(calibratedSpeedFL);
12    frontRightMotor.setSpeed(calibratedSpeedFR);
13    rearLeftMotor.setSpeed(calibratedSpeedRL);
14    rearRightMotor.setSpeed(calibratedSpeedRR);
15 }

```

Listing 3. Codul funcției *setup*

```

1 // Funcția care calibrează vitezele motoarelor pe baza datelor
2   ↳ empirice
2 void calibrateMotors() {
3     Serial.println("Calibrarea motoarelor...");
4
5     // Logică de calibrare exemplu, ajustează în funcție de setup-ul
6       ↳ tău
6     // Poți folosi encodere sau alte mecanisme de feedback pentru a
7       ↳ măsura viteza reală
7
8     int baseSpeed = 120;
9
10    // Ajustează vitezele pe baza datelor empirice sau a mecanismelor
11      ↳ de feedback
11    calibratedSpeedFL = baseSpeed + 5; // Ajustare exemplu
12    calibratedSpeedFR = baseSpeed - 3;
13    calibratedSpeedRL = baseSpeed + 2;
14    calibratedSpeedRR = baseSpeed - 4;
15
16    Serial.print("Viteza calibrată FL: ");
17    Serial.println(calibratedSpeedFL);
18    Serial.print("Viteza calibrată FR: ");
19    Serial.println(calibratedSpeedFR);
20    Serial.print("Viteza calibrată RL: ");
21    Serial.println(calibratedSpeedRL);
22    Serial.print("Viteza calibrată RR: ");
23    Serial.println(calibratedSpeedRR);
24 }

```

Listing 4. Codul funcției *calibrateMotors*

```

1 // Funcția care calibrează senzorii IR pe baza valorilor de bază
2 void calibrateSensors() {
3     Serial.println("Calibrarea senzorilor...");

```

```

4
5 // Citirea valorilor de bază pentru senzorii IR
6 irLeftBaseline = analogRead(IR_LEFT);
7 irRightBaseline = analogRead(IR_RIGHT);
8
9 Serial.print("Valoarea de bază IR stânga: ");
10 Serial.println(irLeftBaseline);
11 Serial.print("Valoarea de bază IR dreapta: ");
12 Serial.println(irRightBaseline);
13 }

```

Listing 5. Codul funcției *calibrateSensors*

```

1 // Funcția care inversează direcția de mișcare a robotului
2 void reverse() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL);
4     frontRightMotor.setSpeed(calibratedSpeedFR);
5     rearLeftMotor.setSpeed(calibratedSpeedRL);
6     rearRightMotor.setSpeed(calibratedSpeedRR);
7
8     frontLeftMotor.run(BACKWARD);
9     frontRightMotor.run(BACKWARD);
10    rearLeftMotor.run(BACKWARD);
11    rearRightMotor.run(BACKWARD);
12 }

```

Listing 6. Codul funcției *reverse*

```

1 // Funcția care validează comanda recepționată prin serial
2 bool validateCommand(char command) {
3     // Adaugă aici verificări pentru comanda primită
4     // De exemplu, poți verifica dacă comanda este una dintre cele
5     // → așteptate
6     return (command == '1' || command == '0' || command == '2'
7             || command == '3' || command == '4' || command == '5');

```

Listing 7. Codul funcției *validateCommand*

```

1 // Funcția care recepționează și validează comenziile seriale
2 char receiveCommand() {
3     while (Serial.available() > 0) {
4         char command = Serial.read();
5         if (validateCommand(command)) {
6             return command;
7         } else {
8             Serial.println("Comandă invalidă recepționată.");
9         }
10    }
11    return '\0'; // Returnează un caracter nul dacă nu a fost primită
12    // → nicio comandă validă

```

Listing 8. Codul funcției *receiveCommand*

```

1 void loop() {
2     // Verifică comenziile primite prin serial
3     if (Serial.available()) {
4         char inputCommand = Serial.read();
5         if (inputCommand == '1') {
6             isActive = true;
7         } else if (inputCommand == '0') {
8             isActive = false;
9             halt(); // Oprește robotul
10        } else if (inputCommand == '2') {
11            moveAhead();
12            delay(1000);
13            halt();
14        } else if (inputCommand == '3') {
15            turnLeft(); // Viraj la stânga
16            delay(500);
17            halt();
18        } else if (inputCommand == '4') {
19            turnRight(); // Viraj la dreapta
20            delay(500);
21            halt();
22        } else if (inputCommand == '5') {
23            reverse(); // Merge în spate
24            delay(1000);
25            halt();
26        }
27    }

```

Listing 9. Codul funcției *loop*

```

1 // Controlează mișcarea robotului dacă este activ
2 if (isActive) {
3     obstacleAvoidance(); // Evită obstacolele
4     if (digitalRead(IR_LEFT) == LOW && digitalRead(IR_RIGHT) == LOW )
5         {
6             moveAhead(); // Mergi înainte
7         } else if (digitalRead(IR_LEFT) == LOW && digitalRead(IR_RIGHT)
8         == HIGH ) {
9             Serial.println("Viraj la stânga");
10            turnLeft(); // Viraj la stânga
11        } else if (digitalRead(IR_LEFT) == HIGH && digitalRead(IR_RIGHT)
12        == LOW ) {
13            Serial.println("Viraj la dreapta");
14            turnRight(); // Viraj la dreapta
15        } else if (digitalRead(IR_LEFT) == HIGH && digitalRead(IR_RIGHT)
16        == HIGH ) {
17            halt(); // Oprește robotul
18        }
19     // Mișcă continuu servomotorul

```

```

17     servoPos += servoStep;
18     if (servoPos >= 150 || servoPos <= 30) {
19         servoStep = -servoStep;
20     }
21     ultrasonicServo.write(servoPos);
22 }
23 }
```

Listing 10. Codul funcțiilor *isActive*

```

1 // Funcția care gestionează evitarea obstacolelor
2 void obstacleAvoidance() {
3     distanceAhead = measureDistance();
4     if (distanceAhead <= 20) {
5         halt();
6         Serial.println("Obstacol detectat: Oprit");
7         checkLeft(); // Verifică distanța la stânga
8         checkRight(); // Verifică distanța la dreapta
9         delay(100);
10    if (rightDist <= leftDist) {
11        detectedObstacle = true;
12        navigate(); // Navighează pentru a evita obstacolul
13        Serial.println("Viraj la stânga pentru a evita obstacolul");
14    } else {
15        detectedObstacle = false;
16        navigate(); // Navighează pentru a evita obstacolul
17        Serial.println("Viraj la dreapta pentru a evita obstacolul");
18    }
19    delay(100);
20 }
21 }
```

Listing 11. Codul funcției *obstacleAvoidance*

```

1 // Funcția care măsoară distanța până la un obstacol folosind
2 // senzorul ultrasonic
3 int measureDistance() {
4     delay(50);
5     int distanceCm = distanceSensor.ping_cm();
6     if (distanceCm == 0) {
7         distanceCm = 100;
8     }
9     return distanceCm;
}
```

Listing 12. Codul funcției *measureDistance*

```

1 // Funcția care verifică distanța la stânga
2 int checkLeft() {
3     ultrasonicServo.write(150);
4     delay(500);
5     leftDist = measureDistance();
```

```
6     delay(100);
7     ultrasonicServo.write(90);
8     Serial.print("Distanța la stânga: ");
9     Serial.print(leftDist);
10    return leftDist;
11 }
```

Listing 13. Codul funcției *checkLeft*

```
1 // Funcția care verifică distanța la dreapta
2 int checkRight() {
3     ultrasonicServo.write(30);
4     delay(500);
5     rightDist = measureDistance();
6     delay(100);
7     ultrasonicServo.write(90);
8     Serial.print(" Distanța la dreapta: ");
9     Serial.println(rightDist);
10    return rightDist;
11 }
```

Listing 14. Codul funcției *checkRight*

```
1 // Funcția care oprește robotul
2 void halt() {
3     frontLeftMotor.run(RELEASE);
4     frontRightMotor.run(RELEASE);
5     rearLeftMotor.run(RELEASE);
6     rearRightMotor.run(RELEASE);
7 }
```

Listing 15. Codul funcției *halt*

```
1 // Funcția care mișcă robotul înainte
2 void moveAhead() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL);
4     frontRightMotor.setSpeed(calibratedSpeedFR);
5     rearLeftMotor.setSpeed(calibratedSpeedRL);
6     rearRightMotor.setSpeed(calibratedSpeedRR);
7
8     frontLeftMotor.run(FORWARD);
9     frontRightMotor.run(FORWARD);
10    rearLeftMotor.run(FORWARD);
11    rearRightMotor.run(FORWARD);
12 }
```

Listing 16. Codul funcției *moveAhead*

```
1 // Funcția care navighează robotul pentru a evita obstacolele
2 void navigate() {
3     if (!detectedObstacle) {
```

```

4     Serial.println("Execuție manevră viraj la dreapta");
5     reverse(); // Dă înapoi
6     delay(350);
7     turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
8     delay(400);
9     moveAhead(); // Mergi înainte
10    delay(800);
11    turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
12    delay(450);
13    moveAheadWithAdjustments(); // Mergi înainte cu ajustări
14    delay(600);
15    turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
16    delay(350);
17    moveAheadWithAdjustments(); // Mergi înainte cu ajustări
18    delay(800);
19    turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
20    delay(280);
21    if (digitalRead(IR_RIGHT) == HIGH) {
22        loop();
23    } else {
24        moveAheadWithAdjustments(); // Mergi înainte cu ajustări
25    }
26 } else {
27     Serial.println("Execuție manevră viraj la stânga");
28     reverse(); // Dă înapoi
29     delay(350);
30     turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
31     delay(400);
32     moveAheadWithAdjustments(); // Mergi înainte cu ajustări
33     delay(800);
34     turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
35     delay(450);
36     moveAheadWithAdjustments(); // Mergi înainte cu ajustări
37     delay(600);
38     turnLeftWithAdjustments(); // Viraj la stânga cu ajustări
39     delay(350);
40     moveAheadWithAdjustments(); // Mergi înainte cu ajustări
41     delay(800);
42     turnRightWithAdjustments(); // Viraj la dreapta cu ajustări
43     delay(280);
44     if (digitalRead(IR_LEFT) == HIGH) {
45         loop();
46     } else {
47         moveAheadWithAdjustments(); // Mergi înainte cu ajustări
48     }
49 }
50 }
```

Listing 17. Codul funcției *navigate*

```

1 // Funcția care face robotul să vireze la dreapta
2 void turnRight() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL);
```

```
4     frontRightMotor.setSpeed(calibratedSpeedFR) ;
5     rearLeftMotor.setSpeed(calibratedSpeedRL) ;
6     rearRightMotor.setSpeed(calibratedSpeedRR) ;
7
8     frontLeftMotor.run(BACKWARD) ;
9     frontRightMotor.run(BACKWARD) ;
10    rearLeftMotor.run(FORWARD) ;
11    rearRightMotor.run(FORWARD) ;
12 }
```

Listing 18. Codul funcției *turnRight*

```
1 // Funcția care face robotul să vireze la stânga
2 void turnLeft() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL) ;
4     frontRightMotor.setSpeed(calibratedSpeedFR) ;
5     rearLeftMotor.setSpeed(calibratedSpeedRL) ;
6     rearRightMotor.setSpeed(calibratedSpeedRR) ;
7
8     frontLeftMotor.run(FORWARD) ;
9     frontRightMotor.run(FORWARD) ;
10    rearLeftMotor.run(BACKWARD) ;
11    rearRightMotor.run(BACKWARD) ;
12 }
```

Listing 19. Codul funcției *turnLeft*

```
1 // Funcția care face robotul să vireze la dreapta cu ajustări
2 void turnRightWithAdjustments() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL + 20) ;
4     frontRightMotor.setSpeed(calibratedSpeedFR + 20) ;
5     rearLeftMotor.setSpeed(calibratedSpeedRL + 30) ;
6     rearRightMotor.setSpeed(calibratedSpeedRR + 30) ;
7
8     frontLeftMotor.run(BACKWARD) ;
9     frontRightMotor.run(BACKWARD) ;
10    rearLeftMotor.run(FORWARD) ;
11    rearRightMotor.run(FORWARD) ;
12 }
```

Listing 20. Codul funcției *turnRightWithAdjustments*

```
1 // Funcția care face robotul să vireze la stânga cu ajustări
2 void turnLeftWithAdjustments() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL + 20) ;
4     frontRightMotor.setSpeed(calibratedSpeedFR + 20) ;
5     rearLeftMotor.setSpeed(calibratedSpeedRL + 30) ;
6     rearRightMotor.setSpeed(calibratedSpeedRR + 30) ;
7
8     frontLeftMotor.run(FORWARD) ;
9     frontRightMotor.run(FORWARD) ;
10    rearLeftMotor.run(BACKWARD) ;
11    rearRightMotor.run(BACKWARD) ;
12 }
```

Listing 21. Codul funcției *turnLeftWithAdjustments*

```
1 // Funcția care face robotul să meargă înainte cu ajustări
2 void moveAheadWithAdjustments() {
3     frontLeftMotor.setSpeed(calibratedSpeedFL - 20);
4     frontRightMotor.setSpeed(calibratedSpeedFR - 20);
5     rearLeftMotor.setSpeed(calibratedSpeedRL - 20);
6     rearRightMotor.setSpeed(calibratedSpeedRR - 20);
7
8     frontLeftMotor.run(FORWARD);
9     frontRightMotor.run(FORWARD);
10    rearLeftMotor.run(FORWARD);
11    rearRightMotor.run(FORWARD);
12 }
```

Listing 22. Codul funcției *moveAheadWithAdjustments*

