**UNIVERSITATEA „POLITEHNICA” din TIMIȘOARA**

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA AUTOMATICĂ**

**ROBOTINHO**

**Robot biped comandabil automat și manual**

Profesor coordonator: Autor:

Ș.l. dr. ing. Sorin NANU Silviu-Dumitru BĂTRÎNUȚ

TIMIȘOARA

2019

Cuprins

[1. INTRODUCERE 4](#_Toc8376822)

[1.1 Contextul aplicației 4](#_Toc8376823)

[1.2 Prezentarea temei 4](#_Toc8376824)

[2. STADIUL ACTUAL AL DEZVOLTĂRILOR ÎN DOMENIU 4](#_Toc8376825)

[2.1 Cassie 5](#_Toc8376826)

[2.2 EZ-Robot JD 5](#_Toc8376827)

[3. ASPECTE TEORETICE 6](#_Toc8376828)

[3.1 Echipamente hardware 6](#_Toc8376829)

[3.1.1 Arduino Uno 6](#_Toc8376830)

[3.1.2 ESP32 Thing 7](#_Toc8376831)

[3.1.3 Pixy CMUcam5 7](#_Toc8376832)

[3.1.4 Setul Biped BRAT 8](#_Toc8376833)

[3.1.5 Extech 382280 8](#_Toc8376834)

[3.2 Tehnologii software 9](#_Toc8376835)

[3.2.1 Limbajul C++ 9](#_Toc8376836)

[3.2.2 Limbajul Java 9](#_Toc8376837)

[3.2.3 Arduino IDE 9](#_Toc8376838)

[3.2.4 Android Studio 10](#_Toc8376839)

[3.2.5 PixyMon 10](#_Toc8376840)

[3.2.6 GitHub 11](#_Toc8376841)

[3.2.7 LynxTerm 11](#_Toc8376842)

[4. Arhitectura sistemului 12](#_Toc8376843)

[4.1 Arhitectura hardware şi mecanică 12](#_Toc8376844)

[4.2 Arhitectura software 14](#_Toc8376845)

[4.2.1 Arduino 14](#_Toc8376846)

[4.2.2 ESP-32 15](#_Toc8376847)

[4.2.3 Telefon cu Android 15](#_Toc8376848)

[4.3 Funcțiile sistemului 15](#_Toc8376849)

[5. Implementare 16](#_Toc8376850)

[5.1 Mișcările robotului 16](#_Toc8376851)

[5.2 Arduino 17](#_Toc8376852)

[5.2.1 Modul manual 17](#_Toc8376853)

[5.2.2 Modul automat 17](#_Toc8376854)

[5.3 ESP32 18](#_Toc8376855)

[5.4 Telefon Android 19](#_Toc8376856)

[6. Rezultate și probleme întâmpinate 21](#_Toc8376857)

[7. Concluzii și direcții de dezvoltare 22](#_Toc8376858)

[7.1 Direcții de dezvoltare 22](#_Toc8376859)

[7.2 Concluzii 22](#_Toc8376860)

[8. Bibliografie 22](#_Toc8376861)

1. INTRODUCERE

1.1 Contextul aplicației

În momentul actual tehnologia se schimbă într-un ritm accelerat. În fiecare an apar paradigme noi, iar inginerii și dezvoltatorii sunt implicați într-un proces de învățare continuă. Marile companii promovează o gamă tot mai variată de unelte și tehnologii.

Dacă până acuma roboții staționari au dominat industria, în ultimii ani s-a remarcat utilitatea roboților mobili. Aceștia, dotați cu o inteligență superioară, vor fi capabili să îndeplinească o multitudine de sarcini, spre deosebire de cei statici, care aveau doar o singură atribuție.

Utilitatea lor este vastă. Ei pot fi găsiți în marile depozite, pe câmpurile de luptă dar și la competiții, în scop recreativ. Noi am hotărât să dezvoltăm un robot din ultima categorie.

1.2 Prezentarea temei

Această lucrare își propune să livreze un robot inteligent de formă umanoidă. Rolul său este de a căuta un obiect de forma sferică într-un spațiu special amenajat, să se deplaseze înspre el și de a șuta cu unul din membrele sale. Acest comportament este specific modului de lucru automat, dar robotul poate fi controlat și manual de către utilizator cu ajutorul unei aplicații Android, care comunică prin Wi-Fi.

Ne-am dorit sa realizăm un robot ușor de controlat, care să fie autonom din punct de vedere al deciziilor și distractiv de urmărit.

2. STADIUL ACTUAL AL DEZVOLTĂRILOR ÎN DOMENIU

Robotul, ca definiție, este un dispozitiv reprogramabil care poate înlocui omul pentru anumite sarcini. Roboții umanoizi sunt acei roboți care au formă umană, în special membre, folosite pentru deplasare sau apucare de obiecte.

Termenul de robot a fost formulat pentru prima oară de către scriitorul ceh de science-fiction Karel Čapek și folosit ulterior de Isaac Asimov. De la stadiul de ficțiune roboții au evoluat la realitate pe la mijlocul secolului 20. Primul robot umanoid a fost completat în 1972 de către specialiștii de la Universitatea Waseda din Japonia, acesta având capacitatea de a se deplasa.[1]

În ultimii ani programarea roboților mobili bipezi s-a dezvoltat foarte mult. Marile institute tehnice au investit mulți bani pentru realizarea unor roboți cât mai performanți și utili. Roboții actuali au utilitate variată, ca și:

* Asistarea la recuperarea după dezastre
* Îngrijirea unei locuințe
* Livrarea de bunuri sau produse
* Utilitate militară

Capacitățile acestor roboți includ:

* Mersul și alergatul pe diferite suprafețe
* Menținerea echilibrului
* Salturi
* Comunicarea cu alte dispozitive

­

2.1 Cassie

Cassie este un robot biped, agil și robust, dezvoltat de Univesitatea din Oregon în 2017. Cu ajutorul unei baterii ce durează 24 de ore și a capacităților de a merge, fugi și menține echilibrul în situații de destabilizare, acest robot reprezintă apogeul dezvoltării tehnice în domeniu.

Proiectanții acesteia au îmbinat caracteristici umane dar și de struț pentru o eficiență cât mai mare. Cu cele 6 grade de libertate, Cassie este capabilă să și se așeze, iar apoi să se ridice. Utilitatea ei variază de la căutarea de victime după dezastre până la livrarea de bunuri.[2]

A picture containing indoor, wall, object

Description generated with very high confidence

Figura **2.1**: Cassie

2.2 EZ-Robot JD

Ez-Robot este un robot umanoid cu 16 grade de libertate, fiecare reprezentat de un servo-motor de calitate, cu carcasă metalică. Acesta poate îndeplini multiple sarcini, cum ar fi deplasarea, apucarea sau detecția de obiecte cu ajutorul camerei plasate pe capul robotului. Camera captează imagini, iar un algoritm inteligent detectează tipare, culori, mișcare, coduri QR( Quick response code) și așa mai departe. Totodată, ochii săi sunt compuși din 18 diode foto emisive sau LED-uri( light-emitting diode), fiecare putând fi configurată să emită o culoare din 1 milion de combinații posibile. Apucarea se face prin intermediul mâinilor, unde sunt plasate servo-motoare speciale pentru această sarcină. Cu ajutorul unei aplicații de pe telefon, Ez-Robot poate fi comandat de către utilizator.[3]

A picture containing object, automaton

Description generated with very high confidence

Figura **2.2**: Ez-Robot

3. ASPECTE TEORETICE

3.1 Echipamente hardware

Pentru elaborarea lucrării de licență, pe partea de hardware( aparatură) s-au folosit următoarele:

* Placa Arduino Uno
* Camera Pixy CMUcam5
* Setul Biped BRAT
* Extech 382280

3.1.1 Arduino Uno

Arduino Uno este un microcontroller bazat pe chipul ATmega328P. Îi este ascociat un mediu de dezvoltare, Arduino IDE, unde se poate programa în limbajul C++. Acesta poate fi alimentat printr-o conexiune USB sau cu ajutorul unei surse externe. Voltajul recomandat de alimentare este situate între 7 și 12 V. La rândul lui, Arduino poate alimenta alte circuite prin pinii speciali de 5V, 3,3V și GND( Ground). Placa este echipată cu:

* Un set de 6 pini analogi și 14 digitali pentru a interfața cu alte circuite
* Capacitatea de a comunica serial( UART TTL, SPI sau I2C)
* 2 surse de întreruperi
* 6 surse de PWM pe 8 biți
* Un LED integrat
* Un bootloader intern

Ca și memorie, ATmega328P deține:

* 32 KB de Flash( memorie de program)
* 2 KB de SRAM( Static Random-Access Memory)
* 1 KB de EEPROM( Electrically Erasable Programmable Read-Only Memory)

A circuit board

Description generated with very high confidence

Figura **3.1**: Pinii atașați plăcii Arduino Uno

Ca și alte caractestici, ceasul intern generator de tact(clock) are frecvența de 16 MHz, generând un semnal de perioadă 62,5 nanosecunde.

Această placă este compatibilă cu o varietate mare de senzori, traductoare, dar și cu cele mai importante sisteme de operare, dând dovadă de flexibilitate și versatilitate. Luate în considerare toate aceste aspecte reiese că Arduino este ușor de folosit și poate fi utilizată în proiecte care variază de la uz casnic până la uz industrial.[4]

3.1.2 ESP32 Thing

ESP32 este un microcontroller capabil să comunice prin Wi-Fi și difertie versiuni de Bluetooth( BT4.0, Bluetooth Smart, Bluetooth Low Energy), dar deținând și 28 de pini de uz general. Această placă este special realizată pentru dezvoltarea IoT(Internet of Things), de aici venindu-i și numele. Ca și capacități, pe lângă cele menționate, acesta deține:

* Microprocesor Tensilica LX6 cu 2 nuclee
* O frecvență de clock de până la 240 MHz
* 520 kB de SRAM
* 4 MB de memorie Flash
* Tensiune de operare situată între 2,2V și 3,6V
* Acumulator litiu polimer integrat
* Port USB pentru alimentare și încărcare de cod[5]



Figura **3.2**: ESP-32 Thing

3.1.3 Pixy CMUcam5

Pixy este o cameră cu software/firmware cu sursă deschisă(open-source) folosită, în special, pentru detecția de obiecte. Printre caracteristicile sale se numără:

* O rată a cadrelor de 50 fps
* Compatibilitate cu plăcile din gama Arduino, Raspberry Pi, BeagleBone
* Compatibilitate cu sistemele de operare Windows, Linux, MacOS
* Capacitatea de comunicare prin SPI, I2C, UART, USB
* Programarea în limbajele C/C++ sau Python
* Procesor NXP LPC4330, cu 204 MHz şi 2 nuclee
* Senzor de imagine OmniVision OV9715 cu retoluţie de 1280x800
* Tip lentilă standard M12
* Aria vizuală a lentilei de 75° pe orizontală şi 47° pe verticală
* 264Kb de RAM
* 1Mb de memorie Flash

Datorită dimensiunilor sale mici ( 2,1 cm pe 1,75 cm pe 1,4 cm), CMUcam5 poate fi integrată cu ușurință în majoritatea proiectelor. Setul său de unelte sau API-ul(Application Programming Interface) său facilitează utilizarea de către toate categoriile de programatori.[6]



Figura **3.3**: Pixy CMUcam5

3**.1.4 Setul** Biped BRAT

Acest set(kit), creat de Lynxmotion, conține :

* Un controller SSC-32
* 6 servo-motoare Hitec HS-422
* Piese prefabricate din aluminiu și șuruburi de diferite dimensiuni

SSC-32(serial servo controller) este un microcontroller special folosit pentru a comanda cu precizie servo-motoare. Se bazează pe microchipul Atmega168-20PU. Deține următoarele capacități:

* Poate comanda până la 32 de motoare
* Poate comunica prin RS232 sau UART TTL
* Rezoluția motoarelor este de 0,09 grade
* Raza de acțiune a motoarelor de 180 de grade
* Un loc pentru memorie EEPROM

Motoarele HS-422 sunt niște servo-motoare cu modulare analogică ce pot fi alimentate cu 4,8V sau 6V. Cu cât este mai mare voltajul cu atât este mai rapidă mișcarea motorului. Ele sunt realizate din plastic și au dimensiuni și greutăți mici, pentru a fi ușor de încorporat în majoritatea proiectelor(4 cm pe 1,96 cm pe 3,66 cm și 45,3 grame). Aria lor de rotație este de 180 de grade în ambele sensuri.[7]

3.1.5 Extech 382280

382280 este o sursă programabilă de curent electric. Aceasta poate genera până la 200 W de curent continuu(40V/5A), dar are și 2 ieșiri fixe, de 3,3V/3A și 5V/2A. Poate memora 200 de tipare și oferă stabilitate. Rezoluția buna, de 1 mV si 1 mA și capacitatea de a comunica cu un PC(Personal Computer) în mod bidirecțional, face ca acest produs să fie de încredere și performant.[8]



Figura **3.4**: Sursa Extech 382280

3.2 Tehnologii software

3.2.1 Limbajul C++

C++ este un limbaj de programare orientat pe obiecte, de nivel înalt dezvoltat de Bjarne Stroustrup ca și o extensie a limbajului C. Apărut pentru prima oară în 1985, C++ este unul dintre cele mai răspândite și de încredere limbaje, fiind folosit de o mare parte de dezvoltatori. În martie 2019, acesta s-a situat pe locul 4 ca și popularitate la nivel mondial.[9]

El este un limbaj flexibil și performant, fiind folosit în diferite domenii, de la jocuri video până la telecomunicații. Poate fi folosit pe cele mai utilizate sisteme de operare( Windows, Linux, MacOS).

3.2.2 Limbajul Java

Java este un limbaj de programare de nivel înalt, concurent, orientat pe obiecte și puternic tipizat. A fost proiectat de către cei de la Sun Microsystems, mai târziu cumpărat de Oracle, în anii 1990, făcându-și debutul în 1995. Java a avut o dezvoltare rapidă datorită licenței sale publice, prin care oricine își putea aduce contribuția. Sintaxa este asemănătoare cu cea din C/C++.

O capacitate utilă și care îi dă portabilitate este aceea că un cod Java compilat poate fi rulat pe orice platformă ce suportă Java fără să fie recompilat. O altă caracteristică este cea de Garbage Collector, care gestionează memoria automat, fără intervenția programatorului. În general, această tehnologie este utilă pentru a elibera zonele de memorie nefolosite și pentru a optimiza aplicația. De asemenea, Java este un limbaj multiplatformă.[10]

3.2.3 Arduino IDE

Arduino IDE este un mediu de dezvoltare folosită pentru a scrie și încărca programe pe placa Arduino. Ca și capacități se pot număra următoarele:

* Este multiplatformă
* Deține un compilator de C și de C++
* Pot fi incluse numeroase biblioteci
* Are o structură a codului ușor de înțeles și folosit
* Lucrează cu fișiere cu extensia .ino

Structura unei aplicații scrise în acest mediu este compusă din 2 funcții. Acestea sunt:

* void setup()- care se rulează o singură data, la începutul programului
* void loop()- care se rulează la nesfârșit după terminarea primei funcții



Figura **3.5**: Mediul de dezvoltare Arduino

3.2.4 Android Studio

Android Studio este un mediu de dezvoltare pentru aplicațiile ce rulează pe sistemul de operare Android. Deține compilatoare pentru limbajele Java și Kotlin. Această tehnologie este una performantă, facilitează dezvoltarea de aplicații pe mobil și oferă multe funcții, precum:

* Vizualizarea designului interfeței
* Capacitatea de a instala și rula aplicațiile pe orice variantă de Android
* Analiză statică
* Editare de text inteligentă
* Statistici în timp real despre utilizarea de resurse a aplicației
* Emulator pentru simularea unui aparat ce rulează sistemul de operare Android[11]



Figura **3.6**: Mediul de dezvoltare Android Studio

3.2.5 PixyMon

Ca și funcționare, Pixy se bazează pe semnături de culoare. Poate stoca până la 7 semnături diferite și poate recunoaște sute de obiecte. Programatorul setează semnăturile obiectelor de interes. O semnătură poate fi compusă din mai multe culori, după cum se poate vedea în figura 2. În acest caz se detectează un obiect cu o semnătură compusă din 3 culori. Se poate observa interfața mediului PixyMon, compusă din bări de acțiune, imaginile date de cameră, opțiunile de setare de semnături și editare, dar și un spațiu pentru log-uri.



Figura : Detecția de obiecte cu ajutorul camerei Pixy

Cu ajutorul mediului PixyMon se pot vizualiza imaginile înregistrate de cameră în 3 moduri:

* Default Program(programul implicit de vizualizare de semnături)
* Raw video(program fără procesare;folosit pentru ajustarea parametrilor camerei, ca și focalizarea sau luminozitatea)
* Cooked video(raw video dar cu un strat de imagine procesată deasupra)

Imaginea utilă este împărțită într-o matrice de pixeli de 320 px pe 200 px, iar procesarea se face pe aceasta.[6]

3.2.6 GitHub

GitHub este un serviciu de găzduire(hosting) și versionare pentru programe software. Acesta poate fi accesat atât de pe un navigator de internet(browser web), cât și printr-o aplicație desktop multiplatformă. Fondat în 2008, GitHub este o unealtă utilă pentru gestionarea de proiecte și este cel mai folosit sistem de hosting din lume.

Este bazat pe sistemul Git, adăugând funcționalități noi, cum ar fi vizualizarea de documente, notificări prin poşta electronică(email) sau un sistem de gestionare a erorilor și problemelor întâmpinate.[12]



Figura : Pagina personală de GitHub

3.2.7 LynxTerm

LynxTerm este un software realizat special, de către LynxMotion, pentru testarea funcționalităților plăcii SSC-32. Acesta se prezintă sub forma unui terminal serial și a unor bări de acțiune, de unde se pot da comenzi motoarelor. Această aplicație este utilă pentru a determina buna funcționare a motoarelor.[13]



Figura : Interfață LynxTerm

4. Arhitectura sistemului



Figura **4.1**: Arhitectura sistemului

4.1 Arhitectura hardware şi mecanică

Setul BRAT a venit însoţit de un ghid de asamblare online. Cadrul mecanic al robotului a fost realizat prin îmbinarea pieselor prefabricate conform ghidului. Totodată, cu ajutorul acestui document am integrat motoarele şi placa SSC-32.

Am stablit şi un spaţiu de lucru pentru robot, unde să îşi desfăşoare mişcările. Arena are o formă dreptunghiulară, cu pereți pe toate părțile, având dimensiunile 84,5 cm pe 78 cm pe 4,4 cm, realizată din polistiren și de culoare verde, pentru a contrasta cu sfera de culoare portocalie. Această hotărâre vine în ajutorul camerei, facilitând detecția. Rolul pereților este de a face ca mingea să ricoșeze din ei, astfel aceasta rămânând constant în interiorul arenei.



Figura **4.2**: Spațiul de lucru

Motoarele sunt legate la pinii 0, 1, 2, 16, 17, 18 ai plăcii SSC-32 și corespund articulațiilor robotului conform tabelului 4.1, considerându-se spatele robotului partea unde este fixat controllerul SSC-32.

Tabel **4.1**: Corespondenţa pinilor de pe SSC-32 cu articulaţiile robotului

|  |  |
| --- | --- |
| **Pin** | **Articulaţie** |
| 16 | glezna stângă |
| 17 | genunchiul stâng |
| 18 | șoldul stâng |
| 0 | glezna dreaptă |
| 1 | genunchiul drept |
| 2 | șoldul drept |

Conexiunile dintre Arduino și SSC-32 sunt minimale, doar pentru a realiza comunicarea serială,iar ele sunt următoarele:

* GND la GND(ground sau masă)
* Tx-ul(pinul de transmisie) de la Arduino la Rx-ul(pinul de recepţie) de la SSC-32

Deoarece am folosit multe protocoale de comunicare ale plăcii Arduino cu alte componente, cea mai simplă soluţie rămasă este comunicarea direct prin intermediul pinilor, prin TTL. Conexiunile dintre Arduino şi ESP-32 sunt reprezentate în tabelul 4.2. Pinii celor 2 plăci sunt conectaţi direct prin intermediul unor fire şi transmit voltaj.

Tabel **4.2**: Corespondenţa pinilor plăcilor ESP-32 şi Arduino, alături de funcţionalitea fiecăruia[14]

|  |  |  |
| --- | --- | --- |
| **ESP-32** | **Arduino** | **Funcţie** |
| 2 | 2 | Manual/Automat |
| 15 | 3 | Spate |
| 5 | 4 | Faţă |
| 18 | 5 | Stânga |
| 23 | 6 | Dreapta |
| 19 | 7 | Şut cu dreptul |
| 22 | 8 | Şut cu stângul |

4.2 Arhitectura software

Ca și software, am dezvoltat programe pentru următoarele componente:

* Arduino
* ESP-32
* Telefon cu Android

Toate entitățile implicate au fost interconectate, iar comunicarea între ele este esențială. Pentru a facilita comunicarea am ales diverse protocoale. În figura următoare acestea sunt prezentate .



Figura : Protocoalele de comunicare utilizate între componente

4.2.1 Arduino

Placa Arduino este nucleul sistemului. Rolul codului ce rulează pe ea este de a prelua datele venite de la camera Pixy sau de la placa ESP32, de a le procesa și de a transmite o comandă plăcii SSC-32 în funcție de acestea.

Preluarea de date și prelucrarea lor se face cu ajutorul bibliotecii Pixy. Prin intermediul API-ului se poate determina:

* Numărul de obiecte detectate
* Coordonatele x și y ale centrului fiecărui obiect
* Lungimea și lățimea fiecărui obiect

Iar prin metoda *print()* se pot transmite toate aceste informații pe portul serial.

Pentru a comanda placa SSC-32 este nevoie de a transmite datele într-un anumit format, și anume: #x Py Tz, unde:

* x este pinul atașat motorului ce se dorește a fi comandat
* y este poziția la care trebuie să ajungă membrul comandat de motorul x; 750 ≤ y ≤ 2250 și corespunde unui unghi situat între [0°,180°]
* z este timpul în care să se facă mișcarea, în milisecunde; acest câmp este opțional

Exemplu: #18 P1500 T500

Există și variații ale acestei comenzi.Se poate enumera o listă de motoare și apoi să li se aplice tuturor o poziție și o durată. Exemplu: #18,#17,#16 P1450 T1000

4.2.2 ESP-32

Rolul lui ESP-32 este de a se comporta ca un server. Preia informaţiile de la telefonul cu Android şi transmite comenzi către Arduino.

Comunicaţia cu telefonul se face prin Wi-Fi, aşadar în primă etapă programul ce rulează pe ESP-32 trebuie să iniţializeze o conexiune cu un router sau switch. Pentru o comunicare reuşită, telefonul şi microcontrollerul trebuie să fie în aceeaşi reţea, Local Area Network. După configurarea serverului, urmează acceptarea clientului şi de aşteptarea de request-uri( cereri) HTTP de tip GET din partea acestuia. În final se dă răspunsul request-ului şi se închide conexiunea.

Pentru a transmite date către Arduino se folosesc legături directe între pinii plăcii ESP32 și pinii plăcii Arduino. În funcție de datele primite de la aplicația Android, adică de parametrii cererilor HTTP GET, se va activa un anumit pin pe ESP32, primind 5V. La capătul celălalt, Arduino va citi tensiunea de pe fiecare pin și va ști să ia o decizie în funcție de aceasta.

4.2.3 Telefon cu Android

Scopul telefonului inteligent este de a transmite comenzi manuale către ESP-32. Cu ajutorul unei interfeţe, utilizatorul poate controla în totalitate robotul, putând fi transmise comenzi cu toate mișcările ce le poate realiza robotul și în modul automat.

Pentru o comunicarea de succes este necesară conexiunea la reţeaua în care se alfă şi ESP-32. Interfaţa cu utilizatorul este simplă, conţinând 7 butoane sugestive pentru fiecare funcţionalitate descrisă în tabelul 4.2. La apăsarea oricărui buton se va trimite un HTTP request de tip Get diferit pentru a informa microcontrollerul ce are de făcut în continuare. Se implementează un design atrăgător şi un logo propriu.

4.3 Funcțiile sistemului

Sistemul este capabil să ofere următoarele funcţionalităţi:

* Căutare automată a obiectului dorit
* Deplasarea în orice direcţie
* Lovirea obiectului când este în proximitate
  + Cu piciorul drept
  + Cu piciorul stâng
* Rularea unui scurt dans
* Comandare manuală

5. Implementare

5.1 Mișcările robotului

Robotinho este capabil să realizeze 6 mișcări de bază:

* Mișcarea în față
* Mișcarea în spate
* Mișcarea în stânga
* Mișcarea în dreapta
* Șut cu piciorul drept
* Șut cu piciorul stâng

Prin combinația acestora se pot forma alte mișcări sau funcții, cum ar fi deplasarea în lateral, care vor fi explicate ulterior.

5.1.1 Pași în lateral

Din cauza limitărilor mecanice ale robotului, axa libertății gleznelor este la robot pe direcția stânga- dreapta spre deosebire de cea umană care este atât față- spate, cât și ușor stânga- dreapta nu am reușit să implementez o metodă de a păși în lateral de natură umană, firească. Așadar a fost necesară o altă abordare. Am decis să optez pentru varianta în care reorientarea robotului se va face prin intermediul unei alunecări spre direcția dorită. **!!eventual pus poza cum aluneca!!** Ulterior alunecării se va face un pas înspre noua direcție. **!!De zis cate grade isi schimba orientarea!!(cam 3 mutari de astea aprox 90 grade),tre masurat cu raportor**

Alunecarea se produce prin întinderea în față a piciorului corespunzător direcției dorite, iar în același timp întinzându-se în spate piciorul celălalt. Datorită poziției obținute, a picioarelor solide și a dimensiunilor mărite ale lăbilor picioarelor, robotul nu va avea probleme de echilibru. Astfel de probleme apar doar în momentul în care robotul se bazează doar pe un picior, în special în cazul în care se produce un șut.

Întinderea piciorului din față se produce prin flexarea genunchiului și ridicarea piciorului de la nivelul șoldului.**!!De zis si cat sa se flexeze!!.** Întinderea piciorului din spate se realizează doar prin împingerea piciorului în spate de la nivelul șoldului, fără a flexa vreun genunchi.

Cu ajutorul acestei metode se pot realiza schimbări de direcție, atât în stânga, cât și în dreapta. Iar pentru a implementa căutarea de obiecte în cazul în care nu este detectat niciunul, am optat pentru 3 întoarceri consecutive în partea stângă, aleasă aleator. Această întoarcere complexă reușește să schimbe o mare parte din orizontul camerei robotului, astfel încât există noi posibilități de detecție a unui obiect.

5.1.2 Mișcarea în față

Pasul în față se va folosi pentru a completa alunecarea discutată anterior, astfel obținându-se o mișcare în lateral completă. Aceasta este constituită din 2 pași. Întâi se păsește cu piciorul drept, iar apoi cu cel stâng, după care se va reveni la poziția inițială pentru a putea acționa și alte mișcări.

Cei 2 pași se bazează pe același principiu de mișcare. Inițial, se înclină cele 2 tălpi pentru ca robotul să aibă o poziție ușor înclinată. Această acțiune se realizează la ambele tălpi pentru că una este necesară ca bază de susținere pentru robot, iar cealaltă pentru a-l propulsa spre poziția dorită. Înclinarea este necesară pentru a permite o pășire cât mai optimă, fară a duce la dezechilibrarea robotului. Fără aceasta, robotul se va prăbuși în direcția piciorului cu care inițiează pasul.

După finalizarea acestei faze, se va trece la pasul efectiv, care constă în ridicarea piciorului de inițiere a pasului de la nivelul șoldului, flexarea genunchiului și întinderea piciorului secundar în spate. Urmează lăsarea primului picior jos și repoziționarea robotului în postura de referință pentru a-l pregăti pentru al doilea pas, care este simetric cu primul.

Toate aceste mișcări sunt consecutive și despărțite de unele pauze temporale pentru a se asigura realizarea lor. În final se va aplica iar o întârziere ca să se evite conflictele între mișcări și o golire a tamponului( buffer) portului serial pentru a șterge orice bit rămas rătăcit ca să nu influențeze următoarele acțiuni.

Mișcările de mers în față, spate și șuturile atăt cu dreptul, cât și cu stângul au fost realizate de colegul meu, Neamțiu Daniel.

5.2 Arduino

Pentru început, am inițializat toate componentele de care am avut nevoie, mai exact cu un obiect de tip Pixy, comunicația serială, cea SPI și am setat pinii 2-8 pe modul intrare( input). Comunicația serială s-a realizat cu o rată de transfer( baudrate) de 115200 biți/secundă, pentru o comandare mai rapidă a motoarelor.

După inițializări, am apelat funcția de poziție dreaptă, iar apoi cea de dans. Până în acest moment toate metodele se execută doar o singură dată, fiind situate în funcția

*void setup()*.

În interiorul funcției *void loop()*, unde toate instrucțiunile se repetă la nesfârșit( echivalent cu un *while(true)*), prima verificare este cea a modului de comandare a robotului. Pinul 2 ataşat plăcii Arduino este cel responsabil pentru detectarea modului. Dacă acesta este alimentat cu tensiune de 3,3V atunci modul de comandă este manual, adică prin comenzile date de utilizator prin aplicaţia mobilă. Dacă pe pinul 2 nu se citeşte nici-o tensiune atunci modul este cel automat.

5.2.1 Modul manual

Odată întrat în acest mod, programul verifică existenţa tensiunii pe fiecare pin în parte, dintre cei de interes, și anume 3, 4, 5, 6, 7, 8. Dacă există, atunci se vor executa funcţiile corespunzătoare, iar apoi robotul va reveni în poziţia de drept. Dacă nu există, atunci se va trece la verificarea următorului pin.

5.2.2 Modul automat

Iniţial se determină numărul de obiecte detectate de camera Pixy. În funcţie de această variabilă se vor lua deciziile următoare:

* Dacă nu se găseşte nici un obiect atunci atunci robotul va realiza o căutare
* Dacă se va găsi un singur obiect atunci se va face o analiză mai complexă
* Dacă se vor găsi mai multe obiecte, atunci se va determina acel obiect cu aria cea mai mare şi se va aplica analiza menţionată anterior

Analiza discutată mai sus este următoarea:

Se verifică numărul cadrului, doar din 50 în 50 de cadre se trimite un semnal de existenţă a unui obiect. Dacă numărul cadrului este divizibil cu 50 atunci se va continua rularea algoritmului, atlfel nu se va lua nici o decizie. Dacă se identifică un obiect atunci se va împărţi imaginea, care are dimensiunea de 320x200 pixeli, în 3 zone distincte pentru a facilita luare unei decizii. Acele 3 zone( fiecare cu o abscisă diferită) sunt următoarele:

* [0, 106) - Reprezintă partea stângă a imaginii, iar dacă se detectează un obiect în această regiune se va lua decizia de deplasare către stânga
* [106, 213) - Reprezintă partea de mijloc a imaginii. Este necesară încă o analiză pentru determinarea deciziei de şut sau deplasare. Dacă obiectul detectat se alfă în ultimul sfert al imaginii( y ≥ 150) atunci se va realiza o deplasare de 2 paşi în faţă, urmată de un şut. Piciorul şutului se determină comparând poziţia obiectului faţă de axa x = 155. Dacă originea obiectului este strict mai mică decât 155 atunci se va şuta cu piciorul stâng, altfel cu cel drept. Dacă y < 150 atunci se va face doar un pas în faţă
* [213, 319) - Reprezintă partea dreaptă a imaginii, iar dacă se detectează un obiect în această regiune se va lua decizia de deplasare către dreapta



Figura **5.1**: Împărțirea imaginii în cadrane conform algoritmului

În cazul în care nu se găseşte nici un obiect, atunci robotul va trebui să realizeze o căutare în felul următor:

1. Întâi se va deplasa un pas în spate
2. Apoi va face o rotaţie de 90° spre stânga. Dacă nici în acest moment nu se găseşte nici un obiect se va relua algoritmul începând cu pasul 1. Rotaţia se va face exclusiv după un pas în spate, doar dacă nu se găseşte nici un obiect

5.3 ESP32

Acest cod se ocupă doar de controlul manual al robotului.

Pentru a putea folosi instrumentele de comunicare prin cereri de tip HTTP și alte capabilități avansate ce necesită conexiunea la internet, se va folosi biblioteca *WiFi.h*.

În primul pas, se inițializeză comunicarea serială pentru a afișa mesaje sugestive pe ecran, de exemplu mesajul de conectare la rețeaua Wi-Fi: *Serial.println("Conectat");*. Aceasta se realizează cu ajutorul unei rate de transfer( baud-rate) de 115200. Baud-rate-ul reprezintă numărul de biți pe secundă transmiși sau recepționați pe UART, transferul fiind sincronizat de un circuit generator de tact. Ulterior, se inițializează vectorul ce reprezintă pinii utilizați de pe ESP32, se vor seta acești pini pe modul de OUTPUT pentru a putea fi citite valorile acestora de către Arduino și li se vor aloca valoarea LOW, corespunzătoare cu 0 Volți, pentru a nu transmite valori eronate la început.

Apoi, ESP32 se conectează la o rețea Wi-Fi, fiind necesare un ssid( service set identifier) și o parolă pentru conectarea la routerul vizat. Este necesară și conectarea telefonului mobil, cu care se va lucra, la această rețea. Codul de pe ESP32 se va comporta ca un server, iar aplicația Android ca și un client. Serverul se creează pe portul 80( portul implicit pentru un server web), cu adresa IP( Internet Protocol) 192.168.1.126, care este o adresă IP privată din familia 192.168.0.0. Acest IP va exista în cadrul rețelei la care se conectează atât telefonul ce rulează aplicația Android, cât și ESP32-ul. Conectarea la aceeași rețea facilitează comunicarea între cele 2. În cazul în care acestea ar fi în două rețele diferite, atunci ar fi nevoie de setarea de reguli de redirecționare pe routerele ce controlează aceste rețele. Această metodă ar duce la complicarea trimiterii de request-uri( cereri) și răspunsuri de tip GET.

Odată realizate toate aceste funcții de inițializare, se trece la codul ce rulează în bucla infinită *void loop()*. În această funcție se verifică dacă serverul este disponibil pentru a accepta conexiuni, apoi dacă clientul este disponibil pentru a transmite cereri. După aceste verificări se creează o sesiune de lucru, care rămâne deschisă doar în cadrul unei iterații a buclei menționate anterior, iar apoi se trece la comunicarea datelor. În cazul serverului, acesta va trebui să citească ce va recepționa de la client. Se citește octet cu octet până se ajunge la terminatorul de șir ‘\n’. Se verifică ce parametrii are cererea de tip GET și în funcție de aceasta se ia o decizie. Decizia constă în setarea unui pin din cei 7 care sunt folosiți: 2, 15, 5, 18, 23, 19, 22. Acești pini au fost aleși deoarece sunt plasați în ordine consecutivă pe placa ESP32 și sunt ușor de urmărit. Resetarea pinilor se face automat după un scurt timp, de 5 secunde, pentru a nu transmite la infinit mesajul de activare a unei funcționalități către Arduino. În acest timp oricare comandă se poate realiza cu succes, iar ea nu va mai fi repetată, doar dacă utilizatorul va dori acest lucru.

În final, se închid conexiunile, sesiunea de lucru și se afișează un mesaj corespunzător.

5.4 Telefon Android

Aplicația ce este destinată telefoanelor inteligente( smart-phone) cu sistem de operare Android conține 2 mari componente:

* Logica sau codul sursă
* Interfața

Codul sursă este constituit din 4 funcții:

* *void onCreate(Bundle savedInstanceState)*
* *void apasareButonManual(View v)*
* *void apasareButonUniversal(View v)*
* *void sendGetRequest(String url)*

*onCreate* este o funcție ce se apelează automat la inițializarea activității de care aparține. Ea poate fi suprascrisă, însemnând că i se poate aloca o diferită funcționalitate de către programator. În cazul nostru, se va suprascrie apelând funcția *onCreate* din clasa de bază, AppCompatActivity, iar apoi se va seta structura( layout-ul) *activity\_main*, care definește interfața și legăturile acesteia cu codul sursă**.**

AppCompatActivity este clasa de bază pentru toate activitățile, care permite utilizatorului să folosească bări de acțiune( action-bar) și alte elemente interactive; pe de altă parte, parametrul *Bundle savedInstanceState* reține toată starea aplicației în cazul în care trebuie recreată, de exemplu, în momentul în care se iese din aplicație, din diferite motive și se intră din nou.

Cea de a doua funcție din lista de mai sus are ca scop trimiterea de cerere de tip HTTP către ESP32 cu informații despre modul de control al robotului( manual sau automat). Clasa ce înglobează toate aceste metode are și o variabilă privată( care nu poate fi accesată din exteriorul clasei) care reține modul de control, numită *manualSauAutomat*. Cum există doar 2 moduri de control am ales ca tipul acestei variabile să fie boolean, care are tot 2 valori posibile:

* true, care reprezintă starea manuală
* false, care reprezintă starea automată

Această funcție are ca parametru un obiect de tip View. El reprezintă componenta grafică, care a apelat această funcție. Aceste componente pot fi, în acest caz, doar butoanul de schimbare a modului de control( manual/automat). Cu ajutorul obiectului de tip View se pot extrage unele informații despre acea componentă, cum ar fi identificatorul ei.

Am declarat o variabilă de tip String unde se va stoca url-ul( Uniform Resource Locator- ce reprezintă o cale, o adresă web) ce se va trimite în cererea de tip HTTP. În funcție de valoarea variabilei de tip boolean discutată mai sus se va lua decizia de a popula variabila url-ului. Totodată, am mai declarat și o variabilă de tip RequestQueue, care este un obiect care gestionează operațiunile de networking, cache- ul și răspunsurile analizei tipurilor de date( parsing)

După ce se decide ce parametrii va avea url-ul, se va apela funcția *sendGetRequest*, care va trimite efectiv cererea și va fi acoperită în rândurile următoare.

În final se va comuta valoarea *manualSauAutomat* și se va adăuga cererea în coada de cereri.

*apasareButonUniversal* este cea de a treia funcție, pe care o vom discuta. Aceasta este asemănătoare cu precedenta doar că are ca țintă toate celelalte butoane, în afară de cel de selectare a modului de control.

Se declară, din nou, variabilele de url și coadă de cereri, în același scop, iar apoi se va verifica ce buton a apelat funcția curentă, prin identificatorul lui. Odată găsită componenta, se va popula url-ul corespunzător. În final, se va apela *sendGetRequest* și se va adăuga cererea în coadă.

Ultima funcție discutată este cea de trimitere a cererii HTTP. Îi este pasat un string, ce reprezintă url-ul țintă. Această funcție declară un obiect de tip cerere, căruia îi este pasat argumentul discutat anterior și următorii parametrii:

* Predicatul, în cazul nostru cererea este de tip GET
* O funcție ce ascultă răspunsul cererii
* O funcție ce ascultă după eventuale erori

Aceste ultimte 2 funcții, am ales să fie nule pentru a nu aglomera interfața cu mesaje.**!!aici vazut ca poate fac totusi ceva!!**

Pe lângă aceste funcții, mai sunt adăugate și alte linii de cod adiționale, dar necesare, cum ar fi permisiunea de a utiliza lucrul cu internetul: *android.permission.INTERNET* sau bibliotecile care permit utilizarea tuturor acestor instrumente, cum ar fi biblioteca Volley care conține mai multe subordonate, ca și *com.android.volley.Request*.

Pe partea de interfață există următoarele elemente:

6. Rezultate și probleme întâmpinate

Pe parcursul dezvoltării lucrării ne-am lovit de unele dificultăți, peste care am trecut cu bine, în final. Acestea sunt următoarele:

* Gestionarea cablurilor. Acestea nu erau fixate strâns, iar mișcările robotului contribuiau la deconectarea cablurilor. Totodată, ele afectau echilibrul robotului.
* Camera Pixy este sensibilă la detecția de obiecte când lumina ambientală este diferită. În unele cazuri se detectau mai multe obiecte deși doar unul era valid.
* Motoarele au diferite referințe. Teoretic, robotul trebuie să stea drept când toate motoarele sunt pe poziția 1500, dar în practică acest lucru nu se întâmpla. A trebuit să găsim pentru fiecare motor propria sa referință. Totodată, sensurile de deplasare erau diferite. Dacă pentru motorul de la șoldul stâng 2250 era poziția maximă în față, pentru motorul de la genunchiul stâng 2250 era poziția maximă în spate.
* Inițial, robotul devia drastic de la traseu și se mișca prea brusc, dezechilibrându-se, iar noi a trebuit să refacem tot algoritmul de pășit.
* La un moment dat s-a stricat laptopul pe care lucram și unde aveam toate instrumentele. A trebuit să reinstalam tot și am fost inspirați să găzduim pe Git toate lucrările pe care le-am făcut.
* A trebuit să recalculăm parametrii funcției de pas deoarece elementele de hardware ce le adăugam pe robot îi afectau echilibrul. Totodată, a trebuit să optimizăm această funcție și la schimbarea de suprafață de mers.
* Sursa nu furniza destul curent. Pe final, motoarele ajungeau să consume până la 1,9A, iar sursa noastră avea ca și limită superioară doar 1A.
* Am încercat să utilizăm alte 2 module de Bluetooth pentru comunicarea între aplicația mobile și robot, dar am eșuat.
* Se transmiteau caractere ininteligibile de la Arduino la SSC-32 sau unele nu se transmiteau deloc.
* Placa SSC-32 a venit împreună cu un jumper ce lega pinul Rx de GND, iar acest aspect împiedica buna desfășurare a testelor deoarece robotul nu răspundea la nicio comandă.
* Îmbinarea tuturor componentelor și comunicarea dintre ele

În final, am reușit să depășim toate problemele și să livrăm un robot funcțional. Acesta reacționează conform așteptărilor. Respectând arhitectura, am implementat cu succes toate funcționalitățile propuse, atât cele de ce vizau comanda manuală, cât şi cele ce vizau comanda automată.

7. Concluzii și direcții de dezvoltare

7.1 Direcții de dezvoltare

Ne dorim să aducem robotul la un nivel cât mai performant. Acesta poate fi îmbunătățit și optimizat prin următoarele:

* Montarea unei baterii, care să genereze 6V de tensiune și 2A de curent, dar care să aibe și dimensiuni reduse pentru a putea fi încorporată cu ușurință în angrenajul robotului
* Implementarea de căutări de alt gen de obiecte
* Adăugarea de noi funcționalități
  + Marcarea unui gol într-o poartă
  + Interacțiunea cu alți roboți
* Estetizarea ansamblului

7.2 Concluzii

În concluzie, acest proiect a fost unul foarte antrenant, atât pentru mine, cât și pentru colegul meu. Ne-am folosit din plin capacitățile tehnice și inovante pentru a livra un produs utilizabil și robust, dar cu loc pentru îmbunătățire. Efortul comun depus a dus la rezultatele dorite, dar cu mari obstacole întâmpinate în proces.

Utilizarea metodelor tehnice învățate în cadrul cursurilor facultății, a unor metode de inovare sau rezolvare de probleme și de gestionare a timpului au fost esențiale pentru elaborarea lucrării. Totodată, munca depusă a dus la propria dezvoltarea în domeniul ingineriei.

Robotinho reprezintă o mândrie pentru mine și privesc cu optimism dezvoltarea sa viitoare.

8. Bibliografie

1. https://en.wikipedia.org/wiki/Humanoid\_robot
2. http://www.agilityrobotics.com/robots#cassie
3. https://www.robotshop.com/en/ez-robot-jd-humanoid-robot.html
4. https://store.arduino.cc/arduino-uno-rev3
5. https://www.sparkfun.com/products/13907
6. https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:overview
7. http://www.lynxmotion.com/p-395-ssc-32-servo-controller.aspx
8. http://www.extech.com/display/?id=14197
9. https://www.tiobe.com/tiobe-index/
10. https://docs.oracle.com/javase
11. https://developer.android.com/studio
12. https://en.m.wikipedia.org/wiki/GitHub
13. http://www.lynxmotion.com/p-567-free-download-lynxterm.aspx
14. http://www.lynxmotion.com/images/html/build104.htm
15. https://developer.android.com/training/volley/simple
16. https://randomnerdtutorials.com/esp32-web-server-arduino-ide
17. [https://www.microsoft.com/en-us/p/arduino-ide/9nblggh4rsd8?activetab=pivot:overviewtab#](https://www.microsoft.com/en-us/p/arduino-ide/9nblggh4rsd8?activetab=pivot:overviewtab) nesortat
18. <https://developers.google.com/ar/develop/java/quickstart> nesortat