**UNIVERSITATEA „POLITEHNICA” din TIMIȘOARA**

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA AUTOMATICĂ**

**ROBOTINHO**

**Robot umanoid autonom pentru urmărirea obiectelor**

Profesor coordonator: Autori:

Ș.l. dr. ing. Sorin NANU Daniel-Pavel NEAMȚIU

Silviu-Dumitru BĂTRÎNUȚ

TIMIȘOARA

2019

**Cuprins**

[1. INTRODUCERE 4](#_Toc11197818)

[1.1 Contextul aplicației 4](#_Toc11197819)

[1.2 Prezentarea temei 5](#_Toc11197820)

[2. STADIUL ACTUAL AL DEZVOLTĂRILOR ÎN DOMENIU 5](#_Toc11197821)

[2.1 Cassie 6](#_Toc11197822)

[2.2 EZ-Robot JD 6](#_Toc11197823)

[3. ASPECTE TEORETICE 7](#_Toc11197824)

[3.1 Echipamente hardware 8](#_Toc11197825)

[3.1.1 Arduino Uno 8](#_Toc11197826)

[3.1.2 ESP32 Thing 9](#_Toc11197827)

[3.1.3 Pixy CMUcam5 10](#_Toc11197828)

[3.1.4 Setul Biped BRAT 11](#_Toc11197829)

[3.1.5 Extech 382280 11](#_Toc11197830)

[3.1.6 PC 12](#_Toc11197831)

[3.2 Tehnologii software 12](#_Toc11197832)

[3.2.1 Limbajul C++ 12](#_Toc11197833)

[3.2.2 Limbajul Java 12](#_Toc11197834)

[3.2.3 Arduino IDE 13](#_Toc11197835)

[3.2.4 Android Studio 13](#_Toc11197836)

[3.2.5 PixyMon 14](#_Toc11197837)

[3.2.6 GitHub 15](#_Toc11197838)

[3.2.7 LynxTerm 15](#_Toc11197839)

[4. Arhitectura sistemului 16](#_Toc11197840)

[4.1 Arhitectura hardware şi mecanică 17](#_Toc11197841)

[4.2 Arhitectura software 18](#_Toc11197842)

[4.2.1 Arduino 18](#_Toc11197843)

[4.2.2 ESP-32 19](#_Toc11197844)

[4.2.3 Telefon cu Android 19](#_Toc11197845)

[4.3 Funcțiile sistemului 20](#_Toc11197846)

[5. Implementare 20](#_Toc11197847)

[5.1 Asamblare 21](#_Toc11197848)

[5.2 Mișcările robotului 21](#_Toc11197849)

[5.3 Arduino 21](#_Toc11197850)

[5.3.1 Modul comandat manual 23](#_Toc11197851)

[5.3.2 Modul autonom / Pixy 24](#_Toc11197852)

[5.4 Mișcarea în față și în spate 26](#_Toc11197853)

[5.5 Șutul 27](#_Toc11197854)

[6. Rezultate și probleme întâmpinate 29](#_Toc11197855)

[7. Concluzii și direcții de dezvoltare 31](#_Toc11197856)

[7.1 Direcții de dezvoltare 31](#_Toc11197857)

[7.2 Concluzii 31](#_Toc11197858)

[8. Bibliografie 32](#_Toc11197859)

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. Aceștia pot fi găsiți în marile depozite, fabrici, sunt utilizați în diferite competiții sportive, în scop militar, dar și în scop recreativ. Noi am hotărât să dezvoltăm un robot din ultima categorie.

Robotinho este un robot biped, de formă umanoidă, capabil să funcționeze atât în mod autonom cât și controlat prin intermediul unei aplicații mobile. Rolul acestuia este de detecta și căuta un obiect sferic pe baza culorii sale. În momentul în care obictul este găsit, robotul lovește obiectul cu unul dintre picioarele sale, după care începe din nou căutarea lui.

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 în funcție de caz, să lovească obiectul cu unul dintre cele două membre ale sale. Acest comportament este specific modului de lucru automat, în care robotul este ghidat de către algoritmul de detecție și cel decizional, dar robotul poate fi controlat și manual de către utilizator cu ajutorul unei aplicații Android, prin intermediul unui telefon mobil, care comunică prin Wi-Fi cu acesta.

Eu împreună cu colegul meu, Silviu Bătrînuț, am dorit să realizăm un robot ușor de controlat, care să fie autonom din punct de vedere al deciziilor și distractiv de urmărit. Sarcinile noastre au fost împărțite conform următorului tabel:

Tabel **1.1**: Împărțirea sarcinilor

|  |  |
| --- | --- |
| **Bătrînuț Silviu** | **Neamțiu Daniel** |
| Implementare cod ESP32 | Implementare cod Pixy |
| Implementare cod Android | Implementare cod Arduino |
| Implementare mișcare în lateral | Implementare mișcare față și spate |
| Implementare mișcare de întoarcere | Implementare șut cu stângul și dreptul |

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 eficienți.

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 acestui robot 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



Figura 3.1: Arhitectura sistemului

Pentru o mai bună înțelegere a ceea ce urmează este necesară prezentarea arhitecturii sistemului. Aceasta poate fi observată în imaginea de mai sus, unde sunt prezentate componentele sub formă grafică și modul de comunicare dintre ele, inclusiv sensul transmiterii de mesaje, reprezentat de săgeți. Comunicarea se realizează prin variate metode și protocoale, care vor fi descrise ulterior în lucrare.

3.1 Echipamente hardware

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

* Placa Arduino Uno
* ESP32 Thing
* Camera Pixy CMUcam5
* Setul Biped BRAT
* Extech 382280
* PC

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 limbajele C sau C++. Acesta poate fi alimentat printr-o conexiune USB sau cu ajutorul unei surse externe. Voltajul recomandat de alimentare este situat 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)

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]

A circuit board

Description generated with very high confidence

Figura **3.2**: Pinii atașați plăcii Arduino Uno

3.1.2 ESP32 Thing

ESP32 este un microcontroller, creat și dezvoltat de cei de la Espressif Systems, 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 **3.3**: 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 (cadre pe secunde)
* 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 **3.4**: 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 care alcătuiesc scheletul robotului și șuruburi de diferite dimensiuni pentru îmbinarea acestora.

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 tensiunea de alimentare, 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 bună, 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]

Cu ajutorul acestei surse de tensiune se va alimenta placa SSC-32 care controlează și alimentează cele 6 servo-motoare ale robotului.



Figura **3.5**: Sursa Extech 382280

3.1.6 PC

PC-ul a fost utilizat pentru instalarea și stocarea tuturor mediilor de programare, programarea algoritmilor și pentru aplicațiile auxiliare folosite în cadrul dezvoltării proiectului. De asemenea, acesta a fost utilizat inițial și pentru stocarea documentelor și versionarea codurilor sursă pâna în momentul în care am recurs la o soluție profesionistă de a face acest lucru utilizând platforma GitHub.

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. C++ este într-un continuu progres, în momentul actual folosindu-se versiunea 17, urmând versiunea 20, care este în dezvoltare. Poate fi folosit pe cele mai utilizate sisteme de operare ( Windows, Linux, MacOS).

În contextul acestui proiect, acest limbaj se va folosi pentru a implementa algoritmii ce vor rula pe Arduino și ESP32.

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:

* 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
* Este multiplatformă

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 **3.6**: Mediul de dezvoltare Arduino[11]

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[12]



Figura **3.7**: Mediul de dezvoltare Android Studio[13]

3.2.5 PixyMon

Ca și funcționare, camera 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 3.7. 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.

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]



Figura **3.8**: Detecția de obiecte cu ajutorul camerei Pixy

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.[14]

Figura **3.9**: Pagina de GitHub de găzduire a documentelor legate de această lucrare

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.[15]

Cu ajutorul acestei aplicații se poate selecta o comandă pentru un anumit motor. Se pot simula toate combinațiile posibile de valori ale motoarelor. Acestea pot fi introduse manual, prin tastare sau prin ajustarea cu ajutorul unei bări de tip scroll.



Figura **3.10**: Interfață LynxTerm[16]

4. 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 **4.1**: 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 **4.2**: 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 se aplică un logo.

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 Asamblare

Pentru asamblarea robotului am utilizat piesele din kitul BRAT robot biped, o trusă de ustensile și piese procurate de la alți roboți deoarece kitul era incomplet. Ca acest proces să fie desfășurat într-un mod optim, fiecare am asamblat câte un picior al robotului ca apoi să le prindem de partea superioara, care reprezintă un suport pentru baterii, dar și pentru placa SSC-32, obținând astfel scheletul de metal al robotului.

În urmăptorul pas, am montat cele 6 servo-motoare în interiorul cadrului, am realizat conexiunile dintre acestea și placa de comandă, iar pentru a ne asigura că firele nu vor împiedica mișcările robotului am decis să le prindem de scheletul metalic al acestuia prin intermediul unor coliere de plastic.

În lipsa unor baterii de greutate redusă care să nu ducă la afecțiuni asupra echilibrului robotului și care să asigure condițiile necesare pentru alimentarea plăcii și a motoarelor, am decis să utilizăm suportul pentru acestea pentru a depozita elementele hardware prin care se face ghidarea robotului (Arduino, ESP-32). Camera Pixy a fost fixată pe un suport metalic confecționat de noi și care a fost prins cu suruburi de suportul de baterii.

5.2 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 combinații ale acestora se pot realiza și alte mișcări sau funcții, cum ar fi deplasarea în lateral sau întoarcerea, acestea fiind realizate de către colegul meu. Implementarea acestor mișcări vor fi scrise în limbajul C/C++ și aplicate în codul de Arduino.

5.3 Arduino

Placa Arduino reprezintă creierul robotului. Codul de pe aceasta este scris în limbajele C/C++ și are rolul de a gestiona toate informațiile primite de la camera Pixy legate de mediul înconjurător și comenzile primite de la ESP-32, cât și de a lua decizii pe baza acestora.

Codul conține două funcții principale în cadrul cărora se execută toată logica:

* setup() - se execută doar o singură dată în momentul în care se alimentează placa Arduino UNO și care are rolul de a inițializa sistemul pentru rulare. Se seteză baud rate-ul(115200 baud/s) pentru comunicarea cu placa SSC-32 în vederea transmiterii comenzilor de control pentru motoare. Se inițializează biblioteca pentru Pixy, iar apoi se setează pinii în modul input, prin care se vor primi comnezi de la placa ESP-32, atunci când robotul este comandat manual.

După ce s-au realizat toate inițializările necesare rulării algoritmului, toate motoarele robotului sunt duse în poziție inițială, ceea ce face ca robotul să stea drept, iar apoi este pus să execute dansul de început, la finalul căruia este readus în poziția de drept pentru a fi pregătit să execute viitoare comenzi.

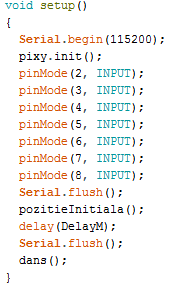


Figura **5.1**: Funcția de inițializare a robotului

* loop() - este apelată după executarea funcției ”setup()” și se execută ciclic cât timp placa este alimentată

Robotul poate să ruleze în două moduri diferite: autonom și comandat manual. Arbitrarea modului în care se află robotul se face pe baza valorii primite de la placa ESP-32 la pinul 2 la începutul fiecărui ciclu al funcție ”loop()”. Dacă valoarea citită este ”HIGH” (nivelul ”1” logic) atunci robotul este comandat manual prin comenzi date din aplicația mobilă, iar în cazul în care valoarea este ”LOW” (nivelul ”0” logic) acesta intră în modul autonom și incepe căutarea obiectului.

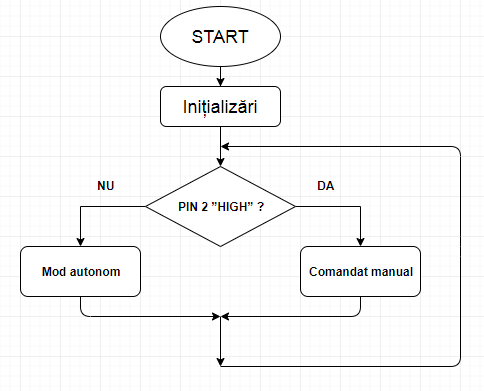


Figura **5.2**: Organigrama de ansamblu a aplicației

5.3.1 Modul comandat manual

În acest mod, robotul execută comenzi primite de la aplicația mobilă care comunică cu placa ESP-32 prin WI-FI, iar aceasta transmite mai departe comenzile către Arduino. În funcție de mișcarea care se dorește a fi executată de robot, este transmisă valoarea ”1” logic la pinul corespunzător acesteia. Corespondența dintre mișcări și pini se poate observa în tabelul din figura 4.2. După executarea mișcării dorite, robotul revine în poziție inițială (poziție de drept), fiind pregătit să execute următoarea comandă.

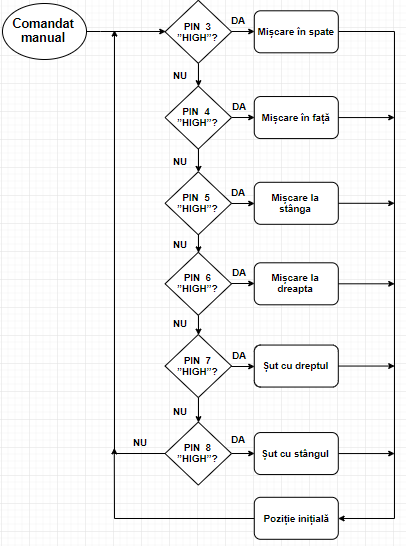


Figura **5.3**: Organigrama modului comandat manual

Pentru părăsirea modului comandat manual, din aplicația mobilă se pune acest mod pe ”OFF”, iar ca rezultat, la pinul 2 al plăcii Arduino UNO va fi transmisă valoarea ”LOW” (”0” logic).

5.3.2 Modul autonom / Pixy

În modul autonom, robotul se folosește de informațiile primite de la camera Pixy pentru a detecta obiectul de o anumită culoare setată în prealabil. În momentul în care obiectul este detectat, robotul se îndreaptă înspre acesta, iar când ajunge în apropierea lui, se poziționează pentru a șuta.

Datele primite de la cameră și comunicația cu aceasta se realizează prin intermediul bibliotecii ”SPI” și a bibliotecii ”Pixy” care stochează datele într-un obiect de tipul Pixy care conține următoarele câmpuri:

* semnătura - poate lua o valoarea de la 0 la 7 (numărul maxim de semnături de culoare pe care Pixy le poate reține este 8) în funcție de semnătura asignată culorii detectate
* x - poate lua o valoare de la 0 la 319 (lățimea imaginii este de 320 px) și reprezintă coordonata pe axa x a mijlocului obiectului detectat
* y - poate lua o valoare de la 0 la 119 (înălțimea imaginii este de 120 px) și reprezintă coordonata pe axa y a mijlocului obiectului detectat
* lățime - lățimea obiectului detectat și poate lua valori între 1 și 320,

320 fiind lățimea imaginii în pixeli

* înăltime - înălțimea obiectului detectat și poate lua valori între 1 și 120, 120 fiind înălțimea imaginii în pixeli
* print() - funcție membră care permite afișarea pe serial a informațiilor despre obiectul detectat
* getBlocks() - funcție membră care returneză numărul de obiecte detectate de către camera într-un cadru

Pixy are capacitatea de a procesa 50 de cadre pe secundă și de a transmite informațiile descrise anterior pentru fiecare cadru în parte și pentru fiecare obiect detectat în frame-ul respectiv. Toate obiectele detectate sunt salvate alături de informațiile descrise anterior într-un vector ”blocks”, acestea fiind indexate în ordinea detectării lor. Pe baza indexului, pot fi accesate toate informațiile referitoare la un anumit obiect printr-un apel de forma:

* pixy.blocks[0].x – utilizat pentru a accesa coordonata x a mijlocului obiectului detectat
* pixy.blocks[0].y – utilizat pentru a accesa coordonata y a mijlocului obiectului detectat
* pixy.blocks[0].print() – folosit pentru a afișa toate date informțiile referitoare la obiectul selectat

Datorită faptului că Arduino UNO nu poate procesa acest flux mare de date, am luat decizia de a procesa datele doar din 50 în 50 de cadre în care camera a detectat un obiect. În momenul în care un obiect a fost detectat, algoritmul va analiza poziția în care acesta se află în imagine și va lua o decizie.

În cazul în care un timp îndelungat nu s-a detectat niciun obiect, robotul va face un pas în spate pentru a lărgi aria vizuală a camerei. Dacă nici în acest caz nu s-a depistat obiectul căutat, robotul va face o întoarcere la stânga de 90 de grade pentru a schimba punctul de vedere al acesteia, iar acești doi pași vor fi repetați până în momentul în care va fi detectat obiectul.



Figura **5.4**: Împărțirea imaginii conform algoritmului

Dacă toate condițiile pentru ca datele să fie procesate au fost îndeplinite, în vederea luării unei decizii, imaginea de 320x120 de pixeli ai camerei va fi împărțită în 3 zone:

* [0, 106) - reprezintă partea stângă a imaginii, iar dacă se detectează un obiect al cărui mijloc se află în această regiune se va lua decizia de deplasare către stânga
* [106, 213) - reprezintă partea de mijloc a imaginii. Dacă obiectul se află în această zonă 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 cu care robotul va şuta 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. În cazul în care originea obiectului are coordonata y < 150, atunci se va face doar un pas în faţă pentru a se apropia de obiect deoarece distanța este prea mare pentru a se poziționa pentru un șut
* [213, 319) - reprezintă partea dreaptă a imaginii, iar dacă se detectează un obiect în această regiune se va lua decizia de deplasare către dreapta

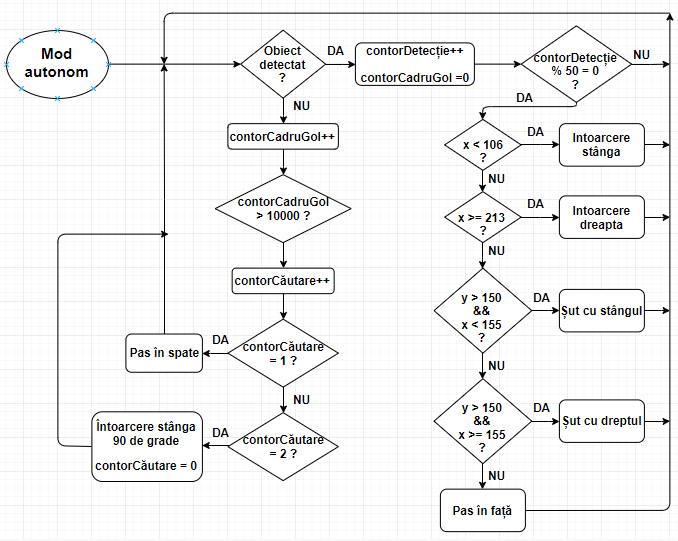


Figura **5.5**: Organigrama modului autonom

5.4 Mișcarea în față și în spate

În faza de început a dezvoltării proiectului am încercat să realizez o mișcare de deplasare în față și în spate cât mai similară cu cea a unui om, dar datorită structurii diferite a membrelor robotului față de cele umane, nu am reușit. Robotul avea probleme cu menținerea echilibrului și cu menținerea direcției de deplasare. Din acest motiv am ales să regândesc algoritmul de pășire pentru a se plia mai bine pe capacitățile sale.

Toate mișcările robotului au fost gândite să pornească din poziție inițială (poziție de drept).

Pentru realizarea mișcării în față, robotul începe prin a-și schimba centrul de greutate pe piciorul drept. Acestul lucru se realizează prin mișcarea simultană a gleznelor sale. Glezna dreaptă este înclinată către exterior pentru a se pregăti pentru schimbarea centrului de greutate pe piciorul drept, iar cea stângă către interior pentru a ajuta la înclinarea robotului, deoarece simpla mișcare a motorului care acționează glezna dreaptă nu produce destulă forță. În acest moment robotul este pregătit să facă primul pas în față, care este realizat printr-o flexare a genunchiului și ridicarea din șold a piciorului stâng. În aceste timp genunchiul drept este ușor flexat în față și șoldul în spate pentru a menține echilibrul. La finalul acestor instrucțiuni, centrul de greutate este mutat pe partea stângă, piciorul stâng devenind cel de sprijin, iar cel drept cel care execută mișcarea. Ca robotul să fie pregătit pentru executarea unor noi mișcări, după excuția celor doi pași, acesta este readus în poziția de drept.

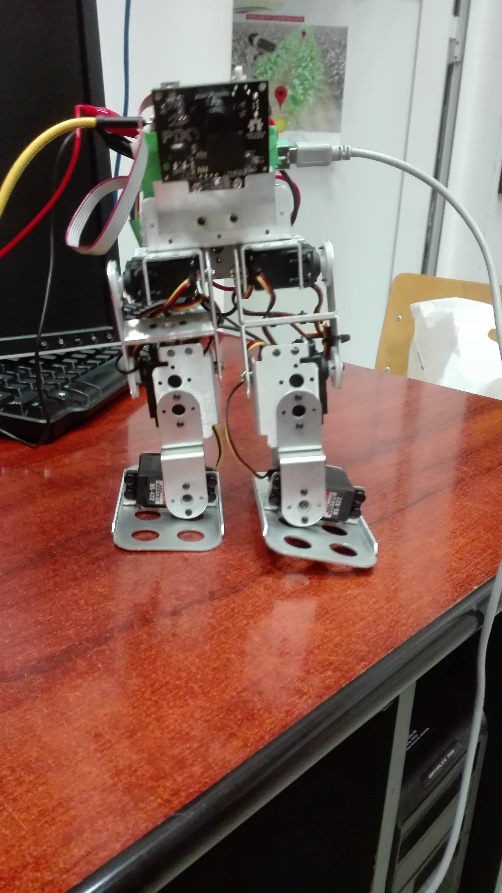


Figura **5.6**: Pasul în față al robotului

Pentru evitarea unui comportament neașteptat al robotului, între fiecare set de comenzi transmis prin intermediul portului serial către placa SSC-32 în vederea controlării motoarelor pentru realizarea mișcărilor dorite, au fost adăugate mici pauze. La final tamponul (buffer-ul) portului serial va fi golit, pentru a evita influențarea unor viitoare mișcări. Am ales această abordare deoarece în cazul unor serii consecutive de comenzi, robotul nu reușea să finalizeze mișcare precedent, ceea ce ducea la un comportament necontrolat și neasteptat, care adesea rezulta prin pierderea completă a echilibrului și răsturnarea robotului.

5.5 Șutul

În funcție de distanța la care se află robotul față de minge și poziția acesteia față de axa centrală a robotului, se decide care picior este potrivit pentru a lovi obiectul. Pentru realizarea șutului au loc următorii pași:

* Piciorul opus celui care va urma să lovească obiectul este pregătit pentru a deveni picior de sprijin. Acest lucru se realizează în același mod precum în cazul deplasării în față. Tălpile robotului sunt acționate simultan, cea a piciorului de sprijin va fi mișcată către exterior, iar glezna piciorului cu care se va șuta, către interior, pentru a ajuta la înclinarea robotului pe piciorul de sprijin. În tot acest timp, motorul care aționează genunchiul și cel care acționează șoldul piciorului de sprijin vor fi setate să mențină poziția inițiala (poziția de drept).

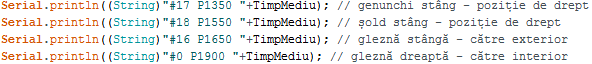


Figura **5.7**: Cod pentru înclinarea pe piciorul stâng al robotului

* După ce primul pas a fost realizat, urmează readucerea gleznei piciorului de șut în poziție inițială pentru a fi dreaptă, iar apoi ducerea în spate a acestuia pentru a pregăti lovirea mingii. Acestu lucru se îndeplinește prin aționarea motorului care controlează șoldul spre ducerea acestuia în spate. Totodata, genunchiul este ușor flexat în spate cu scopul de a evita o posibilă agățare cu talpa, a asuprafeței, în momentul în care se face deplasarea în jos a piciorului, în vederea lovirii mingii.



Figura **5.8**: Cod pentru ducerea în spate a piciorul drept și îndreptarea gleznei

* Ultimul pas este reprezentat de lovirea efectivă a obictului. În acest scop, piciorul de șut este deplasat în față la 66% (două treimi) din capacitate. Ca urmare a acestui pas, mingea este lovită, iar apoi motoarele care acționează șoldul și genunchiul robotului sunt readuse în poziție inițială, în scopul de a pozitiona robotul în starea de drept, pentru a fi pregătit să execute viitoare mișcări dictate de algoritm sau comenzi primite de la aplicația mobilă.

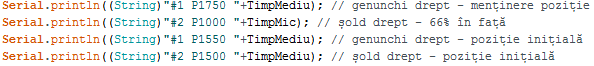


Figura **5.9**: Cod pentru lovirea mingii cu piciorul drept și revenire în poziție inițială

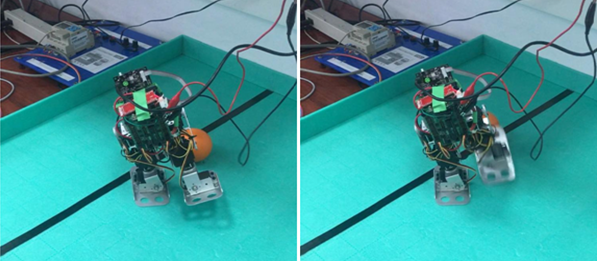


Figura **5.10**: Mișcarea de șut

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 sau slabirea contactelor, lucru care ducea la un comporament neașteptat din partea robotului. Totodată, ele afectau și echilibrul său în timpul anumitor mișcări.
* 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. De asemenea, dacă intensitatea luminoasă din încăpere se schimbă cu mult față de cea din momentul în care semnătura de culoare a obiectului de identificat a fost salvată, există posibilitatea ca obiectul să nu mai fie detectat.
* 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, am încercat să realizăm un stil de a păși al robotului cât mai apropiat de cel uman, însă robotul avea probleme mari de echilibru, deoarece construcția sa e diferită față de cea umană, neavând măini care să îl ajute pentru a evita acest lucru. Ca rezultat devia drastic de la traseu și se mișca prea brusc chiar se răsturna uneori. Pentru a rezolva această problemă, am regândit stilul în care robotul pășește și se deplasează astfel încât să se plieze mai bine pe capacitățile sale ajungând să refacem tot algoritmul.
* La un moment dat s-a stricat laptopul pe care îl utilizam pentru programarea robotului și stocarea uneltelor de lucru necesare vizualizării imaginii de la Pixy și pentru controlul motoarelor robotului. A trebuit să reinstalăm toate instrumetele software necesare. Am fost inspirați să găzduim pe Git toate lucrările pe care le-am realizat.
* A trebuit să recalculăm parametrii funcției pentru realizarea pașilor deoarece elementele hardware pe care le adăugam pe robot duceau la creșterea masei acestuia, lucru care îi afecta echilibrul. Totodată, a trebuit să optimizăm această funcție și la schimbarea de suprafață de mers deoarece frecarea era diferită.
* 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.
* Placa SSC-32 a venit împreună cu un jumper care 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 care vizau comanda manuală, cât şi cele care 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
* Optimizarea modului de lucru manual
* Optimizarea algoritmului de căutare de obiecte
* Adăugarea unui difuzor
* Adăugarea de noi funcționalități
  + Marcarea unui gol într-o poartă
  + Interacțiunea cu alți roboți
  + Metodă de înștiințare în cazul în care nu se află niciun obiect în perimetrul în care se face căutarea
  + Recunoașterea unor coduri de culori, nu doar a culorilor simple
* 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. Istorie roboți umanoizi

https://en.wikipedia.org/wiki/Humanoid\_robot

1. Descriere robot Cassie

http://www.agilityrobotics.com/robots#cassie

1. Descriere robot EZ-Robot-JD

https://www.robotshop.com/en/ez-robot-jd-humanoid-robot.html

1. Fișă tehnică placă Arduino

https://store.arduino.cc/arduino-uno-rev3

1. Fișă tehnică placă ESP32

https://www.sparkfun.com/products/13907

1. Ghid utilizare cameră Pixy și PixyMon

https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:overview

1. Fișă tehnică placă SSC32

http://www.lynxmotion.com/p-395-ssc-32-servo-controller.aspx

1. Fișă tehnică sursă Extech 382280

http://www.extech.com/display/?id=14197

1. Statistici popularitate limbaje de programare

https://www.tiobe.com/tiobe-index/

1. Documentație limbaj Java

https://docs.oracle.com/javase

1. Imagine mediu de dezvoltare Arduino

https://www.microsoft.com/en-us/p/arduino-ide/9nblggh4rsd8?activetab=pivot:overviewtab#

1. Documentație mediu de dezvoltare Android Studio

https://developer.android.com/studio

1. Imagine mediu de dezvoltare Android Studio

https://developers.google.com/ar/develop/java/quickstart

1. Descriere GitHub

https://en.m.wikipedia.org/wiki/GitHub

1. Descriere LynxTerm

http://www.lynxmotion.com/p-567-free-download-lynxterm.aspx

1. Ghid de asamblare robot BRAT Biped

http://www.lynxmotion.com/images/html/build104.htm