



UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI
ELECTRONICĂ
DEPARTAMENTUL DE AUTOMATICĂ ȘI
INFORMATICĂ APLICATĂ



LUCRARE DE DISERTATIE

Coordonator științific:

Șef lucr. Dr. Ing. GANEA ION - EUGEN

Absolvent:

LAURENȚIU - IONUȚ PISTRITU

Iulie, 2021

CRAIOVA



UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI
ELECTRONICĂ
DEPARTAMENTUL DE AUTOMATICĂ ȘI
INFORMATICĂ APLICATĂ



INTERPRETAREA LIMBAJULUI SEMNELOR

Coordonator științific:

Șef lucr. Dr. Ing. GANEA ION - EUGEN

Absolvent:

LAURENȚIU - IONUȚ PISTRIȚU

Iulie, 2021

CRAIOVA

„Învățătura este o comoară care își urmează stăpânul pretutindeni.”

Proverb popular

DECLARAȚIE DE ORIGINALITATE

Subsemnatul, Pistrițu Ionuț - Laurențiu, student la specializarea Ingineria Sistemelor din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea proiectului meu de licență:

- cu titlul Interpretarea Limbajului Semnelor,
- coordonată de Șef lucr. dr. ing. Ganea Ion-Eugen,
- prezentată în sesiunea iulie 2021.

La elaborarea proiectului de licență, se consideră plagiat una dintre următoarele acțiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele și referința precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obținute sau a unor aplicații realizate de alți autori fără menționarea corectă a acestor surse,
- însușirea totală sau parțială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situații neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et cetera,
- precizarea referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data,

01.07.2021

Semnătura candidatului,





UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare și Electronică
Departamentul de Automatică și Informatică Aplicată

Aprobat la data de
.....
Profesor Dr. Ing.

PROIECTUL DE DIPLOMĂ

Numele și prenumele studentului:	Pistitu Ionut - Laurentiu
Enunțul temei:	Interpretarea Limbajului Semnelor
Datele de pornire:	
Conținutul proiectului:	Capitole
Material grafic obligatoriu:	Prezentarea proiectului Prezentare PowerPoint Cod sursă
Consultații:	Periodice
Conducătorul științific (titlul, nume și prenume, semnătura):	Șef lucr. dr. ing. Ganea Ion-Eugen
Data eliberării temei:	
Termenul estimat de predare a proiectului:	01.07.2021
Data predării proiectului de către student și semnătura acestuia:	01.07.2021



UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare și Electronică

Departamentul de Automatică și Informatică Aplicată

REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului: Pistritu Ionut - Laurentiu
Specializarea: Ingineria Sistemelor
Titlul proiectului: Interpretarea limbajului semnelor
Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta):
În facultate ☐
În producție ☐
În cercetare ☐
Altă locație: [se detaliază]

În urma analizei lucrării candidatului au fost constatate următoarele:

Nivelul documentării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Tipul proiectului		Cercetare <input type="checkbox"/>	Proiectare <input type="checkbox"/>	Realizare practică <input type="checkbox"/>	Altul [se detaliază]
Aparatul matematic utilizat		Simplu <input type="checkbox"/>	Mediu <input type="checkbox"/>	Complex <input type="checkbox"/>	Absent <input type="checkbox"/>
Utilitate		Contract de cercetare <input type="checkbox"/>	Cercetare internă <input type="checkbox"/>	Utilare <input type="checkbox"/>	Altul [se detaliază]
Redactarea lucrării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Partea grafică, desene		Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
Realizarea practică	Contribuția autorului	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Mare <input type="checkbox"/>	Foarte mare <input type="checkbox"/>
	Complexitatea Temei	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Analiza cerințelor	Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
	Arhitectura	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Întocmirea specificațiilor funcționale	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Implementarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>

	Testarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Funcționarea	Da <input type="checkbox"/>	Parțială <input type="checkbox"/>	Nu <input type="checkbox"/>	
Rezultate experimentale		Experiment propriu <input type="checkbox"/>		Preluare din bibliografie <input type="checkbox"/>	
Bibliografie		Cărți	Reviste	Articole	Referințe web
Comentarii și observații					

În concluzie, se propune:

ADMITEREA PROIECTULUI <input type="checkbox"/>	RESPINGEREA PROIECTULUI <input type="checkbox"/>
---	---

Data,

Semnătura conducătorului științific,

REZUMATUL PROIECTULUI

Această lucrare urmărește realizarea prototipului unei mănuși electronice dezvoltat în platforma Arduino, pentru a obține date de la senzorii de îndoire și unitatea de măsurare inerțială MPU-6050 (IMU) pentru a fi transmise în aplicația virtuală ce rulează pe Android. Mănușa trebuie să includă capacitatea de a detecta mișcările de rotație ale mâinii utilizatorului, lucru realizat cu ajutorul valorilor oferite de giroscop existent în MPU-6050. Atunci când utilizatorul va îndoi degetele, senzorii de îndoire amplasați pe mănușă își vor schimba rezistența și vor returna o valoare ce va fi interpretată și prelucrată de procesorul ATmega328p existent pe placa Arduino UNO R3. Informația va fi trimisă printr-un modul Bluetooth, informație care va fi utilizată de o aplicație ce rulează pe orice dispozitiv ce are Android ca soft de operare.

Componentele de bază ale mănușii sunt cei cinci senzori de îndoire, o placuță Arduino UNO R3, un dispozitiv cu Android, mănușa, modulul Bluetooth HM-10, MPU-6050 folosit pentru a detecta rotația mâinii utilizatorului, o baterie de 9V și conectori pentru a conecta componentele între ele.

Aplicația ce rulează pe dispozitiv conține un spațiu virtual unde există un model tridimensional (3d) al unei mâini. Aceasta va avea aceeași poziție și va executa aceleași mișcări cu mâna utilizatorului pe care se află mănușa electronică. Modelul tridimensional poate să execute mișcări identice cu cele ale mâinii utilizatorului. Când utilizatorul învârtă mâna sau strânge pumnul același lucru se va întâmpla și cu modelul mâinii tridimensionale.

Atunci când modelul tridimensional și degetele acesteia ajung într-o anumită poziție în spațiul virtual va fi emis un sunet și afișată o imagine corespunzătoare poziției respective a modelului 3d. Acestea au scopul de a ajuta persoanele surdo-mute să comunice mai ușor și mai eficient.

De exemplu, când utilizatorul are mână întinsă orizontal cu degetul mare și arătător în poziție dreaptă, iar celelalte degete sunt în poziția de strâns, modelul 3d din spațiul virtual va avea aceeași poziție, și în același timp pe ecranul dispozitivului va fi afișată imaginea ce ilustrează un preparat culinar și va fi emis sunetul “vreau mâncare”.

MULȚUMIRI

Aș dori să încep prin a mulțumi domului profesor Ganea Ion - Eugen pentru faptul că m-a ales pentru a mă implica în acest proiect, pentru toată munca depusă împreună și mai ales pentru răbdarea și înțelegerea de care a dat dovadă pe parcursul unui întreg an de proiectare, construire și testare a proiectului.

În același timp, aș dori să mulțumesc tuturor profesorilor din cadrul Facultății de Automatică, Calculatoare și Electronică, care, de-a lungul celor doi ani de studiu mi-au împărtășit din experiența și cunoștințele dâșilor și nu în ultimul rând m-au motivat să muncesc și să învăț tot mai mult. Pot spune că cea mai mare parte a cunoștințelor mele a fost dobândită în timpul celor patru ani de studiu prin bunăvoința și răbdarea domnilor profesori.

Nu în ultimul rând doresc să mulțumesc Facultății de Automatică, Calculatoare și Electronică, pentru oportunitatea de a studia într-un mediu complet profesionist cu personal calificat.

Vă mulțumesc !

Cuprins

1	Introducere	1
1.1	Scopul lucrării.....	1
1.2	Motivația.....	1
2	Noțiuni teoretice	3
2.1	Mănușile electronice (mănuși de date)	3
2.2	Interfața grafică.....	5
3	Hardware	7
3.1	Senzori	7
3.1.1	Principiul de funcționare	7
3.1.2	Clasificarea senzorilor & exemple de senzori	7
3.1.3	Senzori de îndoire.....	9
3.1.4	Rezistența	10
3.2	Modulul Bluetooth.....	11
3.3	Unitatea de măsurare inerțială (IMU).....	12
3.3.1	Accelerometrul	13
3.3.2	Giroscopul	14
3.3.3	Magnetometrul	15
3.4	Arduino	15
3.4.1	Microcotroller.....	20
3.4.2	Criterii de alegere a unui microcontrolle.....	23
3.5	Semnal digital	24
3.6	Semnal PWM.....	25
3.7	Convertor Analog – Digital	26
4	Software	27
4.1	Comunicare I2C.....	27
4.1.1	Protocolul de comunicare I2C.....	30

4.2	Unity	31
4.3	MonoDevelop	33
4.4	C#.....	34
4.5	ARDUnity.....	35
4.6	Android.....	37
4.6.1	Caracteristici si specificații ale sistemului Android.....	38
4.6.2	Extensia APK.....	39
5	Dezvoltarea aplicației practice	40
5.1	Diagrama Use Case	40
5.2	Diagrama de secvență.....	40
5.3	Mâna fizică	42
6	Dezvoltarea Hardware.....	43
6.1	Schema bloc.....	47
6.2	Mănușă electronică	48
7	Dezvoltarea Software.....	49
7.1	Codul Arduino	50
7.2	Spațiul virtual	53
7.3	Platforma de dezvoltare	59
8	Testare.....	60
9	Concluzii și modificări pentru viitor	66
9.1	Dezvoltarea viitoare.....	66
10	BIBLIOGRAFIE.....	67
	Codul sursă	69

LISTA FIGURILOR

FIGURA 1 - SAYRE GLOVE	3
FIGURA 2 - EXEMPLE DE MĂNUȘI ELECTRONICE.....	4
FIGURA 3 - SENZOR DE ÎNDOIRE	9
FIGURA 4 - SENZOR DE ÎNDOIRE (PARTICULE CONDUCTIVE DEPĂRTATE- 70K OHM)	9
FIGURA 5 - SENZOR DE ÎNDOIRE (PARTICULE CONDUCTIVE DE APROPIERE- 30K OHM).....	10
FIGURA 6 - SIMBOLUL REZISTENȚEI ELECTRICE [12]	10
FIGURA 7 - MODULUL BLUETOOTH HM-10	12
FIGURA 8 - AXELE DE DETECTARE ALE IMU.....	13
FIGURA 9 - STRUCTURA ACCELEROMETRULUI.....	14
FIGURA 10 - STRUCTURA GIROSCOPULUI [10].....	14
FIGURA 11 - STRUCTURA MAGNETOMETRULUI [11]	15
FIGURA 12 - ARDUINO UNO R3.....	17
FIGURA 13 - CONFIGURAREA HARDWARE ARDUINO (A).....	18
FIGURA 14 - CONFIGURAREA HARDWARE ARDUINO (B).....	19
FIGURA 15 - DIAGRAMA BLOC A MICROCONTROLLERULUI ATMEGA328	21
FIGURA 16 - MICROCONTROLLER VS. MICROPROCESOR.....	22
FIGURA 17 - SCHMITH TRIGGER.....	25
FIGURA 18 - SEMNAL PWM.....	25
FIGURA 19 - CONVERTOR ANALOG – DIGITAL	26
FIGURA 20 - PORTUL SERIAL	28
FIGURA 21 - COMUNICAREA SPI	28
FIGURA 22 - COMUNICAREA I2C (A)	29
FIGURA 23 - COMUNICAREA I2C (B).....	29
FIGURA 24 - PROTOCOLUL DE COMUNICAREA	30
FIGURA 25 - PLATFORME DISPONIBILE ÎN UNITY[22]	32
FIGURA 26 – GAMEOBJECT [22].....	32
FIGURA 27 - PROPIETĂȚILE UNUI GAMEOBJECT [23].....	32
FIGURA 28 - FUNCȚIONARE ARDUNITY	35
FIGURA 29 - ARHITECTURA ARDUNITY [15]	36
FIGURA 30 - ANDROID STRUCTURAT PE 4 NIVELURI	38
FIGURA 31 - A.DIAGRAMA USE CASE	40
FIGURA 32 - B. DIAGRAMA DE SECVENȚĂ.....	41
FIGURA 33 - ARDUINO UNO R3.....	43
FIGURA 34 - ROTAȚIA [17].....	44
FIGURA 35 - SCHEMA DE CONECTARE ANDROID SI BLUETOOTH HM-10	45
FIGURA 36 - SCHEMA DE CONECTARE A SENZORULUI DE ÎNDOIRE LA ARDUINO	46
FIGURA 37 - SCHEMA BLOC.....	47

FIGURA 38 - SCHEMA ELECTRICĂ	47
FIGURA 39 - MĂNUȘĂ ELECTRONICĂ - PRIVIRE DE SUS	48
FIGURA 40 - MĂNUȘĂ ELECTRONICĂ - PRIVIRE LATERALA	48
FIGURA 41 - DIAGRAMA DE PROCESARE	49

LISTA TABELELOR

TABELUL 1 - COMPARAȚIE ÎNTRE MODELELE DE MĂNUȘI ELECTRONICE	4
TABELUL 2 - SPECIFICAȚII ALE IMU MPU-6050	13
TABELUL 3 - PUNCTUL DE MAXIM ȘI MINIM.....	57
TABELUL 4 - SPECIFICAȚIILE PLATFORMEI DE DEZVOLTARE.....	59
TABELUL 5 - POZIȚIA REALĂ VS POZIȚIA VIRTUALĂ.....	65

1 Introducere

1.1 Scopul lucrării

Acest proiect a fost realizat cu scopul de a îmbunătăți viața persoanelor cu dizabilități prin dezvoltarea unei aplicații care ajută la comunicarea acestora cu alte persoane folosindu-se de gesturile mâinii. Această aplicație le va permite utilizatorilor să comunice prin intermediul unei interfețe foarte prietenoasă cu aceștia, lucru fiind posibil prin emiterea unor sunete, dar și prin vizualizarea unor imagini aferente, care sunt în stransă legătură cu pozițiile degetelor.

Codul sursă va fi încărcat pe procesorul existent pe plăcută Arduino, iar acesta va avea capacitatea de a detecta și interpreta mișcările mâinii care mai apoi vor fi trimise printr-un modul Bluetooth la aplicația Unity care v-a rula pe un dispozitiv Android. Mișcările mâinii cât și cele ale degetelor vor fi apoi interpretate de software-ul prezent pe modelul 3d al unei mâini umane care mai apoi va emite sunete și va afișa o imaginea aferentă poziției.

Pe viitor doresc să îmbunătățesc acest proiect prin adăugarea mai multor funcționalități care vor veni în ajutorul persoanelor cu handicap. Utilizatorii vor putea să își introducă semne în aplicație prin intermediul butoanelor special create pentru acest scop, dar și pentru a-și alege sexul vocii care se va auzi la emitere deoarece persoanele care prezintă acest handicap sunt atât femei cât și bărbați.

1.2 Motivația

Potrivit unor cercetări, în România sunt peste 23 000 de persoane cu deficit de vorbire cât și de auz, oameni care trăiesc într-o lume lipsită de sunete și glasuri... o lume a tăcerii, prea puțin cunoscută de noi, cei care auzim. Pentru acești oameni, să te faci înțeles, să poți reda multitudinea de expresii și sensuri, stări și sentimente chiar și niste banale cuvinte este infinit mai greu. Dar cu toate acestea, acești oameni nu sunt muți, ei vorbesc limba română prin semne.

Limbaajul semnelor, special conceput pentru persoanele surdo-mute, face uz de mișcare a degetelor, a buzelor, de limbaajul corpului și de gestică pentru a transmite un mesaj. Datorită limbaajului semnelor, oamenii cu dizabilități pot comunica la fel de ușor și eficient la fel ca restul lumii fără nici un handicap.

Limbaajul semnelor a jucat mult timp un rol vital în comunicarea între persoanele surde și mute, cu toate acestea, nu îi ajută să comunice cu cei care nu o înțeleg, lovindu-se adesea de discriminare.

Motivul pentru care am demarat acest proiect s-a nascut din dorința de a veni în ajutorul acestor persoane cu dizabilități prin intermediul tehnologie. Folosind "mănușa inteligentă", semnele pot fi interpretate și transformate în limbaj verbal, oferind utilizatorilor o voce care poate fi înțeleasă universal.

Mănușa are o serie de senzori care detectează mișcarea degetelor și transmite datele prin bluetooth unui sistem care procesează informațiile primite.

Sistemul este conceput pentru a identifica toate tipurile de gesturi și semene, iar mai apoi aplicația va converti gesturile într-o ieșire vizuală și audio pe telefonul mobil.

Mănușa inteligentă dă o voce nouă persoanelor cu dizabilități!

2 Noțiuni teoretice

2.1 Mănușile electronice (mănuși de date)

Mănușile electronice (mănuși de date) sunt dispozitive care utilizează senzori de mișcare, cum ar fi accelerometre, giroscopice sau senzori de flexibilitate, pentru recunoașterea sau reproducerea gesturilor umane [1]. Acestea sunt folosite în filme, jocuri, controlul dispozitivelor cu acționare robotizată, interacțiunea om-calculator sau interacțiunea cu mediile RV.

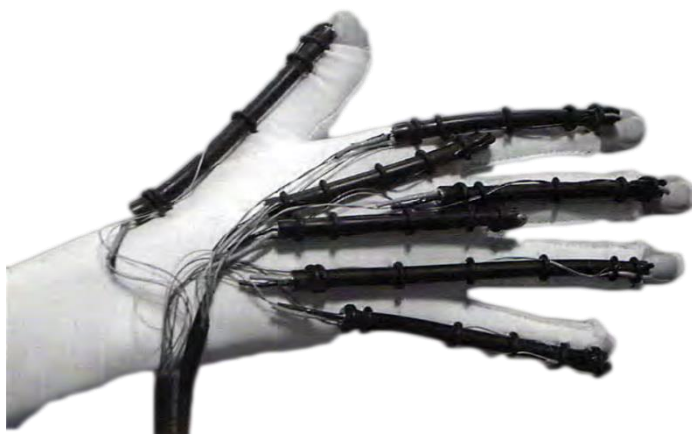


Figura 1 - Sayre Glove

Potrivit lui Sturman și Zeltzer (1994), prima mănușă electronică, Sayre Glove (**Error! Reference source not found.**) a fost elaborată de Thomas DeFanti și Daniel Sandin de la Universitatea din Illinois din Chicago, în 1976 și a funcționat cu tuburi flexibile care aveau un emițător și o lumină poziționată pe degete, iar atunci când s-a îndoit, lumina de pe receptor a variat, generând o diferență de tensiune care a fost măsurată și apoi utilizată ca intrare la un sistem. Aceasta a fost începutul cercetării recunoașterii gesturilor în domeniul informaticii.

În industria de divertisment, în opinia autorilor, producătorul de jucării Mattel a produs în 1989 o mănușă electronică numită Power Glove, care a fost folosită la controlul pentru unele jocuri cu platformă Nintendo. Produsul său a folosit cerneală rezistivă care a înregistrat flexia și un senzor acustic care a capturat poziția mâinii în spațiu cu ajutorul unui transmițător instalat pe televizor. Tehnologia utilizată în mănușile electronice a evoluat considerabil, bazându-se pe senzori mai preciși, prin tehnici noi de captare a mișcării prin intermediul senzorilor de câmp magnetic [2] sau accelerometre [1] în spațiu utilizând senzori cu infraroșu sau camere de luat vedere.



Figura 2 - Exemple de mănuși electronice

CyberGlove III (**Error! Reference source not found.c**) produs de CyberGlove Systems este utilizată pe scară largă în cercetarea academică, ca de exemplu în lucrarea lui Pérez-Duarte (2014), care urmărea să analizeze ergonomia chirurgilor atunci când efectuează o laparoscopică, capturarea mișcărilor încheieturii mâinii și a degetelor chirurgilor. 5DT Data Glove Ultra (**Error! Reference source not found.a**) este o altă mănușă electronică utilizată în capturarea mișcărilor pentru animații realiste în timp real [3]. Diferența este că utilizează senzori de fibră optică pentru a detecta flexia degetelor.

În prezent, se elaborează noi mănuși electronice cu scopul de a fi utilizate în aplicațiile VR sau Realitatea Augmentată (RA). Două exemple notabile sunt Manus VR (**Error! Reference source not found.d**) și Avatar VR (**Error! Reference source not found.b**) care diferă de mănușile menționate mai sus, deoarece, pe lângă senzorii de flexiune, folosesc și așa-numitele Unități de măsură inerțiale (IMU).

Tabelul 1 arată o comparație între modelele de mănuși menționate în această secțiune. Se observă că, dintre mănușile analizate, 5DT Data Glove Ultra este singurul care utilizează senzori bazați pe fibra optică, cu un sensor pe deget. De remarcat, de asemenea, că prețul Cyber Glove III este cel mai mare dintre cele patru, datorită numărului mare de senzori utilizați, fiind de 13 ori mai scump decât cel utilizat pe 5DT Glove Ultra.

Mănuși	Interfață	Tip senzor	Nr. Senzori	Preț
5DT Data Glove Ultra	USB	Fibră optică	5	995 \$
Avatar VR	WiFi	IMU	6	1300\$
Cyber Glove III	WiFi si USB	Flexare	18 sau 22	1400\$
Manus VR	WiFi	IMU / Flexare	1 IMU / 5 Flexare	250\$

Tabelul 1 - Comparație între modelele de mănuși electronice

Una dintre marile dificultăți în evoluția mănușilor electronice se află în cantitatea mare de mișcări care pot fi exercitate de mâna umană, făcând complexă captarea și reproducerea fidelă a calculatoarelor sau a brațelor robotice. Un alt factor limitator este costul ridicat al dispozitivelor utilizate pentru fabricarea mănușilor electronice, care necesită senzori cu precizie ridicată și repetabilitate.

Mănușa propusă în această lucrare prezintă o soluție ieftină pentru captarea mișcărilor degetelor și a rotației mâinii.

2.2 Interfața grafică

Interfețe cu utilizatorul sunt modalitățile prin care un sistem (software) interacționează cu utilizatorii săi (umani).

Aceasta interacțiune poate fi reprezentată de:

- Linia de comandă Grafică (Graphical User Interface - GUI)
- Tactile (Touch User Interface - TUI)
- Multimedia (voce, animație, etc.)
- Inteligente (recunoașterea gesturilor, conversaționale, etc)

Etapele creării unei aplicații cu interfață grafică

- **Design** :crearea suprafețelor de afișare (containere) ,crearea și asezarea componentelor
- **Funcționalitate**: definirea unor acțiuni, “legarea” componentelor de acțiuni , ”ascultarea” și tratarea evenimentelor
- **Considerente**: programmatic, declarative, separare dintre nivelul GUI și logica aplicației.

Interfețele grafice au aparut datorită ideii adaptării calculatorului la om și de dezvoltare ulterioară a conceptului de interfață prietenoasă, asigurând comunicare vizuală între un program și utilizatorii săi.

Conceptul de interfață grafică

Interfața grafică este o interfață cu utilizatorul bazată pe un sistem de afișaj ce utilizează elemente grafice. Interfața grafică este numită sistemul de afișaj grafic-vizual pe un ecran, situat funcțional între utilizator și dispozitive electronice cum ar fi computere, dispozitive personale de tip hand-held (playere MP3, playere media portabile, dispozitive de jucat), aparate electrocasnice și unele echipamente de birou. Pentru a prezenta toate informațiile și acțiunile disponibile, un GUI

oferă pictograme și indicatori vizuali, în contrast cu interfețele bazate pe text, care oferă doar nume de comenzi (care trebuie tastate) sau navigația text.

Precursorul interfețelor grafice a fost inventat de către cercetătorii de la Institutul de Cercetare Stanford, SRI, condus de Douglas Engelbart. Ei au dezvoltat folosirea hiperlinkurilor bazate pe text, manipulate cu un maus. Conceptul de hiperlinkuri, realizat prin așa-numite „texte active” (care conduc la altă pagină web atunci când sunt „apăsate” cu mausul pe ecran), a fost rafinat și extins și spre elemente grafice, formând astfel interfața primară pentru computerul „Xerox Alto”. Majoritatea interfețelor grafice cu scop general sunt derivate din acest sistem. Ca rezultat, unii numesc această clasă de interfețe „interfața PARC” (PUI).

În 1963 Ivan Sutherland a dezvoltat un dispozitiv de indicare și desenare pe ecran numit sketchpad. Systemul utiliza un light pen (un fel de creion electronic care în locul minei de grafit avea în vârf un senzor de lumină) pentru a conduce crearea și manipularea obiectelor din desenele ingineresti de pe ecran.

[https://ro.wikipedia.org/wiki/Interfa%C8%9B%C4%83_grafic%C4%83]

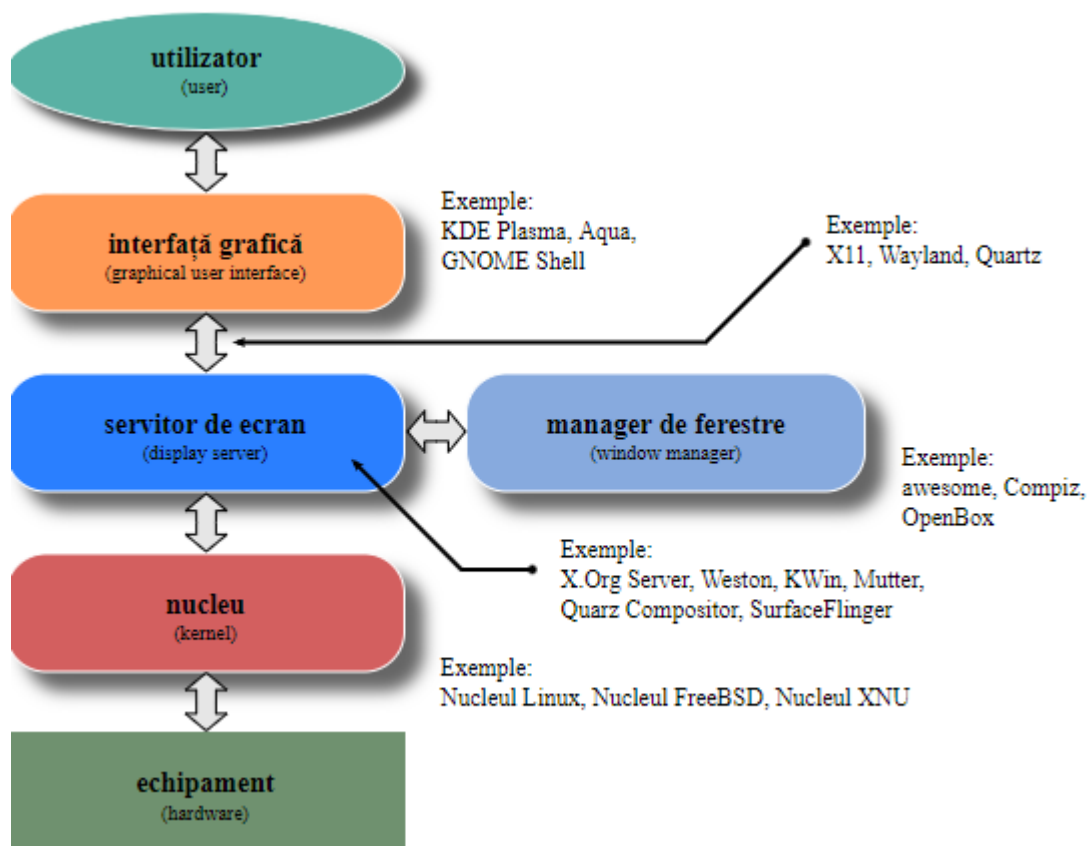


Figura 47 – Straturile interfetei grafice

3 Hardware

3.1 Senzori

Trăim într-o lume în care suntem înconjurați de senzori, fie că vrem sau nu. Îi găsim de diferite tipuri în case, birouri, mașini, stradă, etc. Detectarea prezenței, temperaturii, umidității, presiunii, accelerației toate aceste lucruri sunt posibile datorită senzorilor.

Senzorii dispozitive care măsoară o mărime fizică și o transformă într-un semnal care poate fi citit de către un observator printr-un instrument sau poate fi prelucrat. Originea cuvântului senzor este latină – *sensus* – care înseamnă simț.

3.1.1 Principiul de funcționare

Clasificarea senzorilor este următoarea:

- Senzori activi;
- Senzori pasivi.

Senzorii activi au nevoie de un semnal de excitație, de o sursă de energie adițională pentru a funcționa. Semnalul este apoi modificat pentru a putea produce un semnal de ieșire. Cu alte cuvinte senzorii activi necesită alimentare cu energie electrică deoarece ei au proprietatea de a-și mări sau micșora rezistența, atunci când stimulul extern este aplicat. De exemplu, un fotoresistor, atunci când este luminat, își poate scădea rezistența de la $10\text{k}\Omega$ până la $1\text{k}\Omega$. În funcție de cât de mare a fost căderea, se poate stabili cantitatea de lumină ce cade pe suprafața senzorului.

Senzorii pasivi nu necesită alimentare cu energie electrică deoarece, atunci când sunt excitați de un stimul extern, produc singuri energie electrică. Prin măsurarea cantității de energie produsă se poate stabili puterea stimulului aplicat (de exemplu o celulă fotovoltaică va produce mai mult curent când este soare față de când e înnorat). Uneori, curentul produs este foarte mic și pentru a fi amplificat, și acești senzori vor fi alimentați (dar nu pentru a funcționa ci pentru a li se amplifică semnalul). [4]

3.1.2 Clasificarea senzorilor & exemple de senzori

Clasificarea senzorilor după tipul mărimii de ieșire:

- Analogici

- Digitali (numerici).

Senzorii analogici au semnalul de ieșire proporțional cu mărimea fizică de intrare, pe când senzorii digitali, pot lua numai un număr limitat de valori discrete, care permit cuantificarea semnalului fizic de intrare.

Exemple de senzori conectați la pini analogici:

- Accelerometri, magnetism (polul nord)
- Torsiune, greutate
- Amprentă
- Umezeală pământ (soil moisture)
- Seismic (geophone)
- RFID reader
- Bătăile inimii
- Prezență (PIR)
- Recunoaștere comenzi vocale (voice recognition)

Alte tipuri de senzori:

- Temperatură, umiditate, barometrice (presiune atmosferică), altitudine, ploaie
- Lumină, culoare (RGB)
- Sunet, vibrații
- Distanță [ultrasonic, IR, laser]
- Diverse tipuri de gaze

3.1.3 Senzori de îndoire

Senzorii de îndoire sunt senzori ce care își modifică rezistență în funcție de gradul de îndoire.

Pe măsură ce senzorul este îndoit, rezistența senzorului (măsurată între cei doi pini metalici) se modifică.

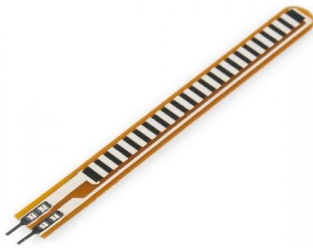


Figura 3 - Senzor de îndoire

Senzorii convertesc îndoirea în rezistență electric – cu cât senzorul este mai îndoit cu atât rezistența este mai mare. Astfel de senzori au fost utilizați la realizarea Nintendo Power Glove.

În stare normală, drept, acești senzori se comportă ca niște rezistori de 30k Ohm. Pe măsură ce sunt îndoiți, rezistența dintre cele două terminale va crește până la 70k Ohm la un unghi de 90°.

Senzori de îndoire funcționează astfel:

O față a senzorului este printată cu o cerneală ce are particule conductive înglobate în ea. Atunci când senzorul este drept, particulele oferă cernelii o rezistență de aproximativ 30k Ohm. Când senzorul este îndoit, particulele conductive se depărtează unele de altele, mărin d rezistența acestuia (la aproximativ 50k – 70k Ohm când senzorul este îndoit la un unghi de 90°, așa cum este arătat în imaginea următoare).

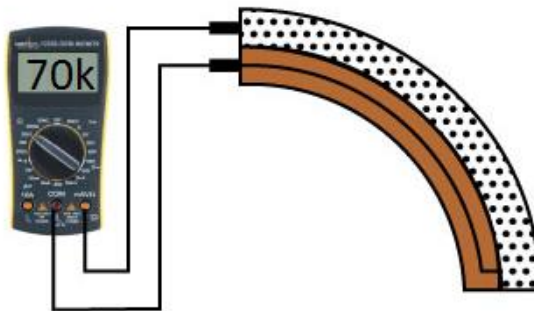


Figura 4 - Senzor de îndoire (particule conductive depărtate- 70k Ohm)

Când senzorul de îndoire este iarăși drept, rezistența senzorului revine la normal. Prin măsurarea rezistenței, se poate determina cât de mult a fost îndoit senzorul.

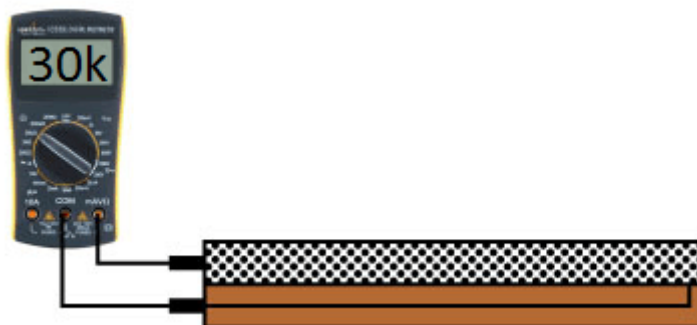


Figura 5 - Senzor de îndoire (particule conductive de apropiere- 30k Ohm)

Cel mai simplu mod de a încorpora acest senzor în proiectul nostru este prin folosirea unui divizor de tensiune. Acest circuit necesită un rezistor. Valorile cuprinse între 10k Ohm și 100k Ohm vor funcționa. O valoare cuprinsă între valoarea minimă și valoarea maximă este de obicei o alegere bună. Prin combinarea unui senzor de îndoire cu un rezistor static pentru a crea un divizor de tensiune, poți obține o tensiune variabilă ce poate fi citită de convertorul analog-digital al unui microcontroller.

3.1.4 Rezistența

Într-un circuit electric, valoarea rezistenței se calculează cu ajutorul legii lui Ohm, fiind egală cu raportul dintre tensiunea U aplicată la bornele sursei și intensitatea I a curentului care circulă prin conductor.

$$R = \frac{U}{I}$$

În schemele electronice se utilizează simbolul rezistorului fix în forma de dreptunghi (simbolizare conform standardului European IEC) sau simbolul “în zig-zag” (conform standardelor din America și Japonia).

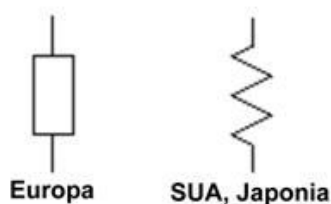


Figura 6 - Simbolul rezistenței electrice [12]

3.2 Modulul Bluetooth

Bluetooth-ul este un set de specificații (un standard) pentru o rețea personal fără fir, bazată pe unde radio. Prin tehnologia Bluetooth se elimină firele și cablurile între dispozitive atât staționare cât și mobile, facilitează atât comunicațiile de date cât și pe cele vocale și oferă posibilitatea implementării unor rețele ad-hoc și a sincronizării între diverse dispozitive.

Printr-o rețea Bluetooth se poate face schimb de informații între diverse aparate precum telefoane mobile, laptop-uri, calculatoare personale, imprimante, camere foto și video digitale sau console video prin unde radio criptate (sigure) și de rază mică, desigur numai dacă aparatele respective sunt înzestrate și cu Bluetooth.

Aparatele care dispun de Bluetooth comunică între ele atunci când se află în aceeași rază de acțiune. Ele folosesc un sistem de comunicații radio, așa că nu este nevoie să fie poziționate față în față pentru a transmite; dacă transmisia este suficient de puternică, ele pot fi chiar și în camere diferite.

Domeniu de aplicare:

- Dispozitive mobile: comunică wireless cu smartphone-uri iOS și Android, tablete, dispozitive de fitness;
- Audio și divertisment: MP3 player, controlul wireless al consolelor de jocuri (Nintendo Wii și Sony PlayStation 3 folosesc tehnologia Bluetooth);
- Industrie: au fost dezvoltate produse industriale bazate pe Bluetooth care sunt utilizate într-o mare varietate pentru automatizarea industrială și comunicarea fără fir între diferitele componente ale mașinilor.
- Industria auto: comunicare wireless între telefoane mobile cu transmițătoare GSM încorporate, sistemul stereo auto compatibil Bluetooth.
- Medicină: monitoarele de glucoză din sânge, pulsometrele, inhalatoarele pentru astm și alte dispozitive medicale care pot fi purtate, utilizează tehnologia Bluetooth pentru a ajuta la administrarea medicamentelor, la diagnosticarea leziunilor și la transmiterea în siguranță a informațiilor critice de la pacienți la furnizori.
- Domotică (o aplicație a calculatoarelor și roboților utilizați pentru aplicațiile casnice): Bluetooth permite controlul automat, centralizat al sistemelor esențiale ale unei clădiri, incluzând încălzirea, ventilația și aerul condiționat, iluminatul, sistemele de securitate.

Pentru acest proiect folosim modulul de Bluetooth HM-10 și este un mic modul 3.3-V BLE Bluetooth 4.0 bazat pe tehnologia TI CC2540 / CC2541 Bluetooth SoC. Acesta poate fi controlat prin intermediul comenzilor AT, care sunt trimise prin conexiunea UART (Universal Asynchronous Receiver/Transmitter). Module HM-10 se bazează pe chipul CC2541, cu o putere mai mică și o rază mai scurtă.



Figura 7 - Modulul Bluetooth HM-10

Caracteristici tehnice:

- Bluetooth Protocol: V4.0 BLE
- USB Protocol: USB V2.0
- Ecran tactil rezistiv cu 4 fire
- Tipul de modulare: GFSK (Gaussian Frequency Shift Keying)
- Rata de transmisie: 6kbps
- Securitate: Autentificare și criptare
- Alimentare: 5V DC
- Temperatură de operare: -5°C~ +65°C
- Dimensiune: 27mm x 13mm x 2.2 mm.

3.3 Unitatea de măsurare inerțială (IMU)

IMU-urile sunt dispozitive care măsoară și raportează diferitele forțe specifice ale corpului, viteză unghiulară și uneori, câmpul magnetic. Pentru asta se folosește o combinație de accelerometre, giroscopae și, în unele cazuri, magnetometre pentru diferitele forțe care acționează, cum ar fi accelerația, rotația sau câmpul magnetic.

Un IMU poate detecta translația și rotația pentru 3 axe X,Y și Z, în total pentru 6 grade de libertate. Făcând o mică paranteză, un IMU este de obicei utilizat, printre multe altele, pentru a manevra avioane, inclusive vehicule aeriene fără pilot. Datorită dezvoltărilor recente s-a permis fabricarea dispozitivelor GPS activate cu IMU.

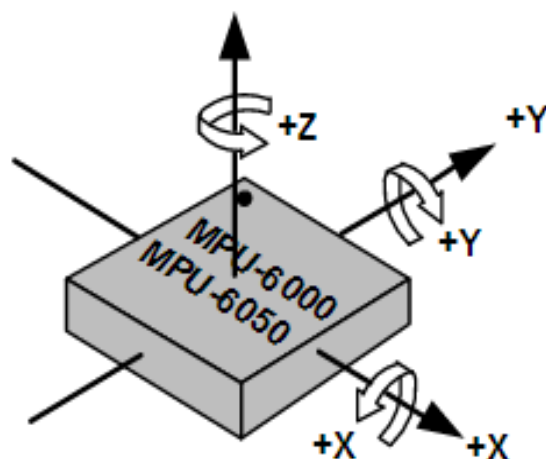


Figura 8 - Axele de detectare ale IMU

Tensiune de operare (VDD)	2.4 V-3.5 V
Tensiune logică (VDDIO)	de la 1.7 V sau VDD
I2C / SPI	ieșire digital
Sensibilitate Accelerometru	16384, 8192, 4096, 2048 LSB/g
Range Accelerometru	± 2 , ± 4 , ± 8 , ± 16 LSB/g
Viteza de zgomot a giroscopului	± 4800 Mf
Interval giroscop	± 250 , ± 500 , ± 1000 , ± 2000 ° / s
Componente Dimensiuni	$3 \times 3 \times 1$ mm

Tabelul 2 - Specificații ale IMU MPU-6050

3.3.1 Accelerometrul

Accelerometrul este un dispozitiv care măsoară intensitatea forței de accelerare exercitată asupra lui. În general, ele au o masă mobilă suspendată de izvoare, iar mișcarea dispozitivului provoacă deformarea arcurilor și generarea unui semnal care este măsurat pentru a informa în ce direcție și cu ce intensitate s-a produs accelerația. De exemplu, un accelerometru se va odihni pe suprafața Pământului cu o accelerație de aproximativ $9,81 \text{ m} / \text{s}^2$ pe axa Z, adică accelerația pe gravitație. Accelerometrele detectează mișcările de translație pe axele X, Y și Z din spațiu, deci are 3 DOF-uri (**Error! Reference source not found.**).

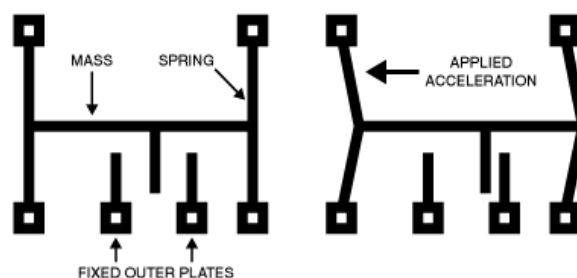


Figura 9 - Structura accelerometrului

3.3.2 Giroscopul

Giroscoapul detectează rotația în jurul fiecărei axe X, Y și Z, adică viteza unghiulară sau rata de schimbare a unghiului de rotație în grade pe secundă ($^{\circ} / s$). Există trei tipuri de bază de giroscopae: Rotary, Optical and Vibratory [5], acestea din urmă fiind cele mai frecvente în circuitele integrate, deoarece sunt mai mici decât alte tipuri. Ele au o masă conectată prin arc la o structură, care la rândul ei este conectată la o a doua structură fixă în circuit (**Error! Reference source not found.**). Masa centrală oscilează vertical, iar atunci când giroscopul este supus unei rotații, structura se mișcă orizontal, datorită forței inerțiale a lui Coriolis¹. Deoarece giroscopul nu are o referință fixă la măsurătorile sale, este susceptibil de erori cumulative în timp, în funcție de utilizare sau de temperatura acestuia.

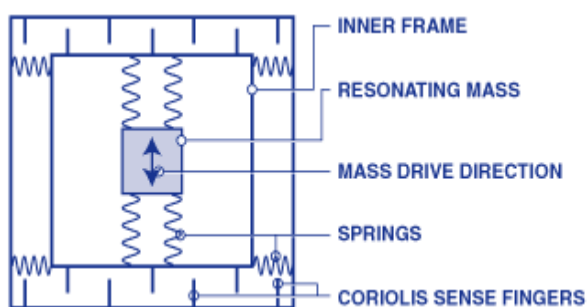


Figura 10 - Structura giroscopului [10]

¹ "Forța Coriolis este o forță care apare într-un sistem referențial rotativ care tinde să schimbe traiectoria corpurilor în mișcare [23]"

3.3.3 Magnetometrul

Magnetometrul este un senzor care detectează efectul *Hall*² (**Error! Reference source not found.**). Când IMU utilizează un magnetometru, este posibil să se detecteze intensitatea câmpului magnetic în jurul dispozitivului.

În IMU, datele de la accelerometru, giroscop și magnetometru sunt în general utilizate împreună pentru a furniza citiri corecte de orientare, deoarece datele accelerometrului sunt foarte zgomotoase, giroscopul este susceptibil de erori cumulative și magnetometrul este supus la interferențe provenite din câmpurile magnetice externe. Citirea lentă a giroscopului este combinată cu cea a accelerometrului pentru a reduce zgomotul, iar citirile magnetometrului sunt folosite pentru a detecta câmpul magnetic al pământului, folosit ca referință pentru a calcula orientarea și pentru a corecta erorile giroscopului.

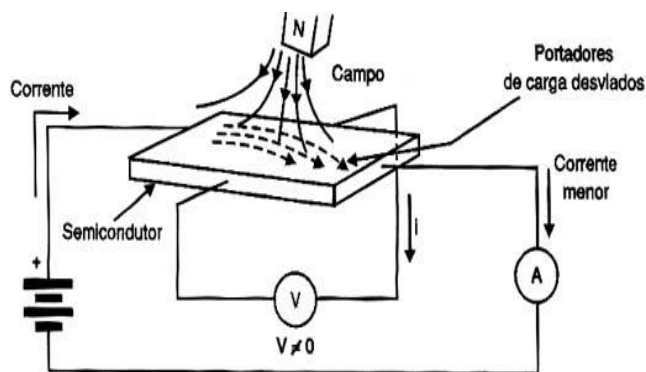


Figura 11 - Structura magnetometrului [11]

3.4 Arduino

Arduino este un instrument cu ajutorul căruia poți realiza sisteme informatice capabile să "perceapă" și să "controleze" lumea înconjurătoare. Acest instrument este open-source și este compus dintr-un mediu de dezvoltare (o variantă de Wiring – platforma folosită pentru procesare multimedia) și o placă de dezvoltare cu microcontroler AVR.

Arduino poate fi folosit pentru dezvoltarea de obiecte interactive. Informația este preluată de la o gamă variată de senzori și comutatoare, se procesează în interiorul microcontrolerului AVR și este transmisă către o gamă la fel de variată de lumini, motoare sau procesoare.

Arduino este una dintre cele mai simple de utilizat platforme cu microcontroller. În jurul lui Arduino există un ecosistem de dispozitive extrem de bine dezvoltat. Orice fel de informație îți ai

² "Efectul Hall se referă la abaterea traiectoriei normale a sarcinilor care curg într-un semiconductor atunci când este supus acțiunii unui câmp magnetic. Această abatere determină o diferență de potențial care poate fi măsurată perpendicular pe direcția mișcării curente. [24]"

dori să culegi din mediu, orice fel de conexiuni cu alte sisteme ai avea nevoie, există o șansă foarte mare să găsești un dispozitiv pentru Arduino capabil să îți ofere ceea ce ai nevoie.

O placă de dezvoltare Arduino este compusă dintr-un microcontroller Atmel AVR de 8-biți, 16-biți sau 32-biți (deși începând cu 2015 s-au folosit microcontrollere de la alți producători) cu componente complementare care facilitează programarea și încorporarea în alte circuite. Un aspect important la Arduino este că acesta dispune de conectori standard, care permit utilizatorului să conecteze plăcuța cu procesorul la diferite module interschimbabile numite shield-uri. Unele shield-uri comunică cu Arduino direct prin pinii digitali sau analogici, iar altele sunt adresabile individual prin magistrala serială I²C permițând utilizarea mai multor module în paralel. Până în anul 2015 plăcuțele Arduino oficiale au folosit cipuri Atmel din seria megaAVR, în special ATmega8, ATmega168, ATmega328, ATmega1280 și ATmega2560, iar în 2015 au fost adăugate cipuri de la alți producători. O multitudine de alte procesoare au fost folosite de dispozitive compatibile Arduino. Multe plăcuțe includ un regulator liniar de 5V și un oscilator cu cuarț de 16 MHz (sau un rezonator ceramic în unele variante), deși anumite plăcuțe, cum ar fi LilyPad, funcționează la 8 MHz și nu necesită regulator, datorită restricțiilor de formă. Un microcontroler instalat pe Arduino vine preprogramat cu un bootloader care simplifică încărcarea programelor pe memoria flash a cipului, în comparație cu alte dispozitive care necesită programatoare externe. Acest aspect face Arduino o soluție simplă, permițând programarea de pe orice computer. În prezent, bootloader-ul optiboot este bootloader-ul implicit instalat pe Arduino UNO.

La nivel conceptual, când se folosește mediul de dezvoltare integrat Arduino, programarea tuturor plăcuțelor se face prin conexiune serială. Implementarea acesteia diferă în funcție de versiunea hardware. Unele plăcuțe Arduino au implementate convertoare de nivel logic pentru a realiza conversia între nivelele logice RS-232 și cele TTL. Plăcuțele Arduino din prezent sunt programate prin USB, având integrate cipuri de conversie USB-serial, cum ar fi FTDI FT232. Unele modele ATmega2560, mai noi, folosesc un cip AVR separat programat să funcționeze ca un convertor USB-serial, care poate fi reprogramat printr-un port ICSP dedicat. Alte variante, cum ar fi Arduino Mini și versiunea neoficială Boarduino, folosesc adaptoare detașabile USB-serial, cabluri, Bluetooth sau alte metode.

Plăcuța Arduino are expuși mulți dintre pinii de intrare/ieșire ai microcontrolerului, pentru ca aceștia să fie folosiți de alte circuite. Placa de dezvoltare Arduino ATmega2560 oferă 54 pini digitali de intrare/ieșire, dintre care 15 pot produce semnale PWM și 16 intrări analogice care, de asemenea, pot fi folosite ca intrări/ieșiri digitale. Acești pini sunt accesibili prin partea superioară a plăcuței, prin intermediul unor barete mamă cu pasul între pini de 2,54 mm.

Plăcile de dezvoltare Arduino seamănă foarte mult între ele (din elementele comune am putea enumera: intrările/ieșirile digitale, intrările analogice, microcontrolerul etc.). Din acest motiv descriem în continuare doar placa de dezvoltare Arduino UNO R3:

- intrare analogică: este folosită pentru citirea semnalelor nondigitale. De exemplu senzori de temperatură, senzori de lumină, senzori de presiune, senzori de umiditate etc.
- intrare/ieșire digitală: imaginați-vă un întrerupător de la un bec. Acesta poate să aibă 2 stări: închis sau deschis adică 0 sau 1.
- PWM (Pulse-width modulation): modulația în durată a impulsurilor. Poate fi utilizat pentru a îndeplini o varietate de sarcini, de la iluminarea LED până la controlul vitezei motoarelor electrice.

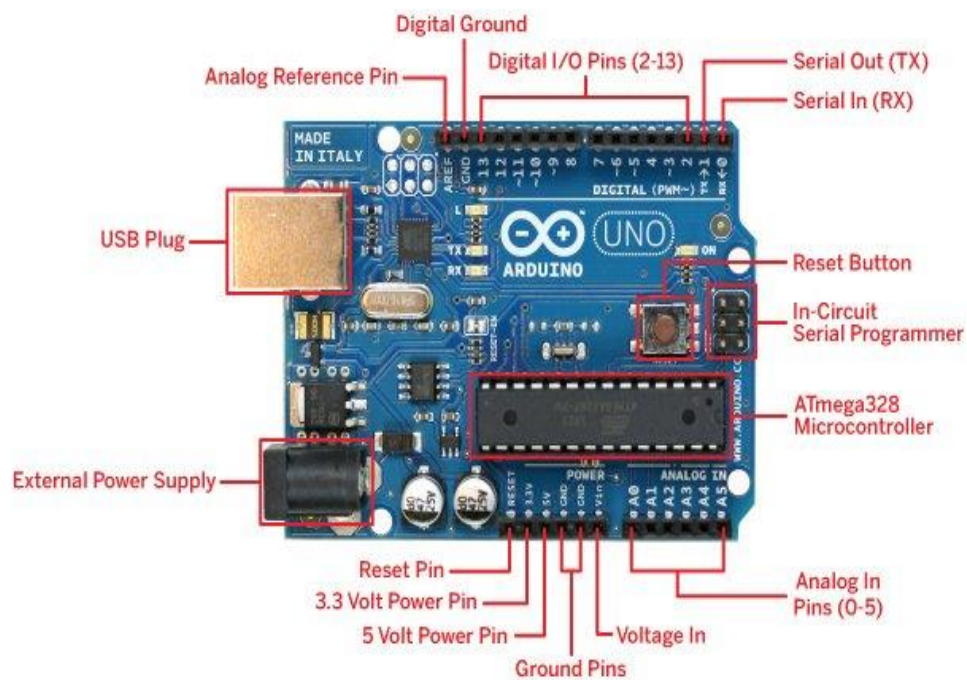


Figura 12 - Arduino UNO R3

Specificații :

- Microcontroler: ATmega328
- Tensiune de lucru: 5V
- Tensiune de intrare (recomandat): 7-12V
- Tensiune de intrare (limită): 6-20V
- Pini digitali: 14 (6 PWM output)
- Pini analogici: 6
- Curent per pin I/O: 40 mA
- Curent 3.3V: 50 mA
- Memorie Flash: 32 KB (ATmega328) 0.5 KB pentru bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

Limbajul de programare Arduino este o versiune simplificată de C/C++. Dacă se cunoaște C, programarea în Arduino va fi foarte ușoară.

O caracteristică importantă pe care o are Arduino este faptul că se poate realiza un program pe PC, se poate downloada pe plăcuța Arduino, iar acesta va funcționa automat. După îndepărtarea cablului USB, programul tot va funcționa de la capăt de fiecare dată când butonul de reset va fi apăsat. Însă adevărata putere a plăcii de dezvoltare Arduino este abilitatea de a interacționa cu lumea exterioară prin pinii săi de intrare-iesire (I/O). Arduino are 54 pini digitali I/O numerotați de la 0 la 53 ce pot fi folosiți pentru pornirea sau oprirea motoarelor și luminilor sau pentru a citi starea unui switch. Fiecare pin digital poate primi sau trimite o intensitate de 40mA. Pentru controlul altor dispozitive decât simple LED-uri sunt necesare circuite adiționale de conectare. Cu alte cuvinte, nu poți folosi un motor ce utilizează curentul disponibil pe un pin Arduino. Pentru a interacționa cu lumea exterioară, programul setează pinii digitali pe o valoare "high" sau "low" folosind instrucțiuni ale codului C, ce corespund valorilor de +5V sau 0V.

Secvența evenimentelor este prezentată în această **Error! Reference source not found.a.**

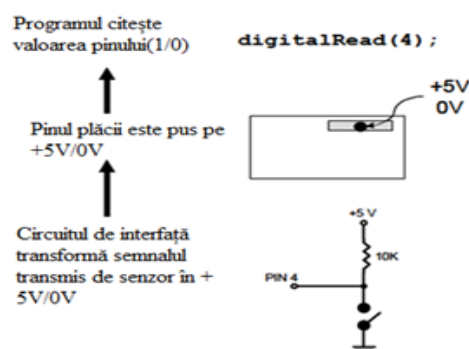


Figura 13 - Configurarea hardware Arduino (a)

Pentru a determina starea swichurilor și a altor senzori, Arduino poate citi valorile tensiunii aplicate pe pini ca numere binare. Circuitul de interfață traduce semnalul sensorului într-un semnal de 0 sau 5V aplicat pinului digital I/O. Arduino interoghează starea pinului. Dacă pinul este pe 0V, programul îl va citi ca 0 sau LOW. Dacă este pe +5V, programul îl va citi ca 1 sau HIGH. Dacă se aplică mai mult de 5V, plăcuța Arduino ar putea fi distrusă.

Secvența evenimentelor este prezentată în și în **Error! Reference source not found.b**.

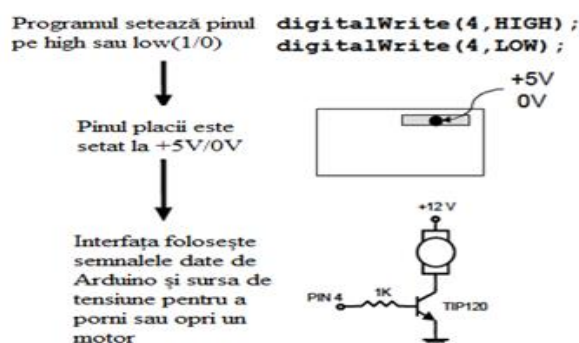


Figura 14 - Configurarea hardware Arduino (b)

Interacțiunea cu lumea exterioară are două părți. În primul rând, programatorul trebuie să realizeze circuitul electronic de interfață ce permite motorașelor și altor dispozitive să fie controlate de un semnal de curent (1 – 10mA) ce comută între 0 și 5V, și alte circuite ce convertesc citirile sensorului într-un semnal de 0 sau 5V. În al doilea rând programatorul trebuie să scrie programul folosind un set de comenzi Arduino ce setează și citește pinii I/O.

Atunci când se citesc intrările, pe pini trebuie să fie aplicată o tensiune de 0 sau 5V. Dacă un pin este lăsat "în aer", va citi tensiuni aleatorii și va determina rezultate eronate. Din această cauză întrerupătoarele au întotdeauna un rezistor de "pull up" de 10kΩ când sunt legate la un pin Arduino.

Pinii 0 și 1 se evită a se folosi deoarece acești pini sunt folosiți pentru comunicația serială între Arduino și calculatorul gazdă. Plăcuța Arduino are de asemenea 16 pini analogici de intrare folosiți pentru citirea tensiunilor în sfera 0 – 5V luate de la senzori, cum ar fi potențiometrul.

Arduino vine cu un mediu de dezvoltare integrat (IDE) bazat pe proiectul Processing, care include suport pentru limbaje de programare ca C și C++.

3.4.1 Microcontroller

Microcontroller-ul este o structură electronică destinată controlului unui proces al cărei mod de funcționare poate fi modificat prin programarea acesteia cu un soft special creat în acest scop. De asemenea un microcontroller poate fi considerat un microcalculator într-un singur cip, acesta încorporează pe același cip un microprocesor (CPU) împreună cu cele mai des utilizate periferice și o memorie, care-i permit interacțiunea cu mediul exterior. Microcontrollerul diferă de un microprocesor în multe feluri. În primul rând și cel mai important este funcționalitatea sa. Pentru a fi folosit, unui microprocesor trebuie să i se atașeze alte componente precum memoria și componente pentru primirea și trimiterea de date. Microcontrollerul fiind proiectat pentru a conține toate acestea într-unul singur. Nu sunt necesare alte componente externe pentru utilizarea sa pentru că toate perifericele necesare sunt deja incluse în el.

Un microcontroller tipic mai are facilități de prelucrare la nivel de bit, de acces direct și ușor la intrări/ieșiri și un mecanism de prelucrare a întreruperilor rapid și eficient.

În altă ordine de idei, toate aplicațiile care folosesc ca platformă de dezvoltare un microcontroller fac parte din categoria sistemelor încapsulate, la care existența unui sistem de calcul este aproape transparentă față de utilizator.

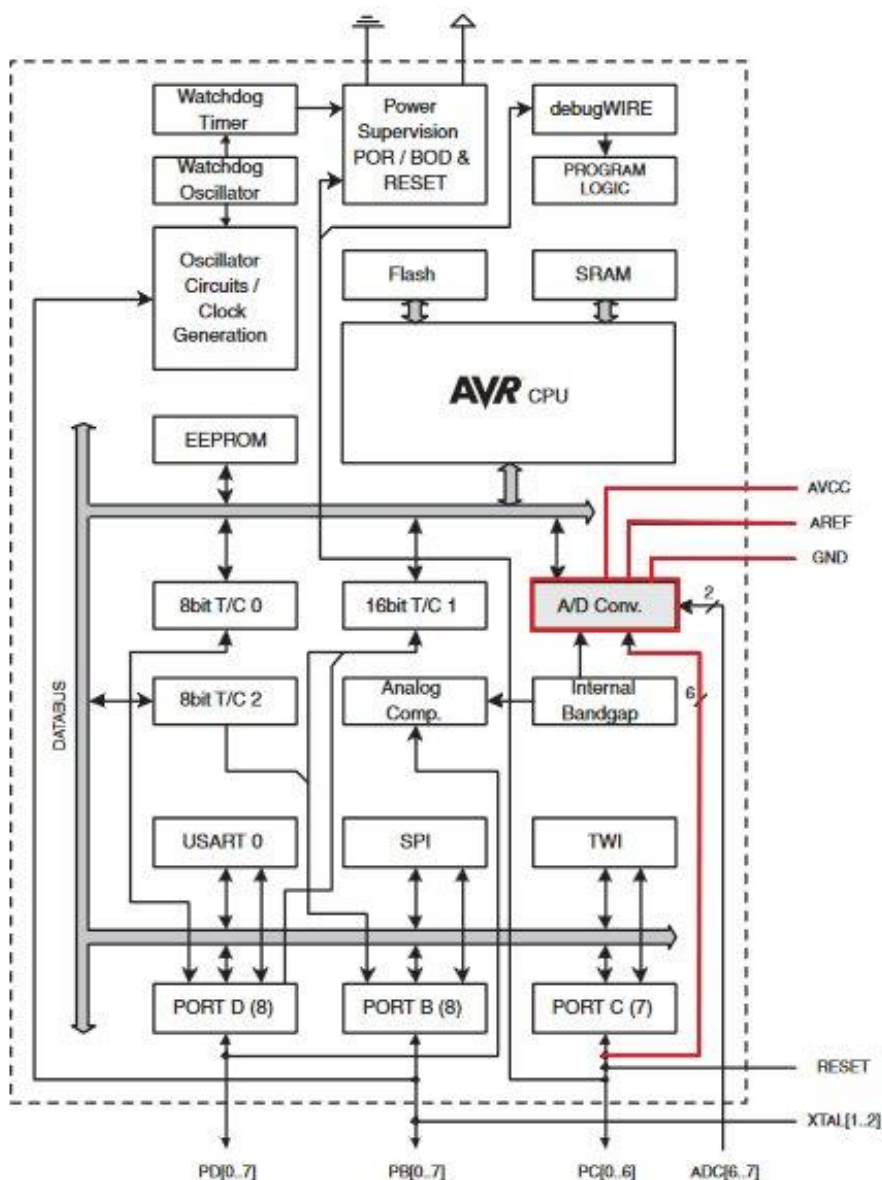


Figura 15 - Diagrama bloc a microcontrollerului ATmega328

Aplicații în care se folosesc microcontrollere sunt în următoarele industrii:

- automotive:
 - controlul confortului din mașină (climatizare, lumini ambientale, reglarea scaunelor etc.)
 - controlul sistemului de frânare
 - control al direcției
 - controlul pornirii și opririi automate a motorului
 - controlul la distanță al mașinii
- echipamente de zi cu zi:
 - GPS-uri
 - jocuri electronice

- telefoane mobile
- camere video
- televizoare
- electrocasnice
 - aspiratoare
 - mașini de spălat
 - cuptare cu microunde
 - frigidere
 - aparate de aer condiționat
- în aerospațială
- în realizarea de instrumente de măsurare
- în realizarea de periferice pentru calculatoare
- medicină

De exemplu orice automobil fabricat este echipat cu cel puțin un microcontroller care comandă motorul mașinii și adesea chiar cu mai multe pentru controlul sistemelor adiționale din automobile, ajungand pana in prezent la peste 120 microcontrollere.

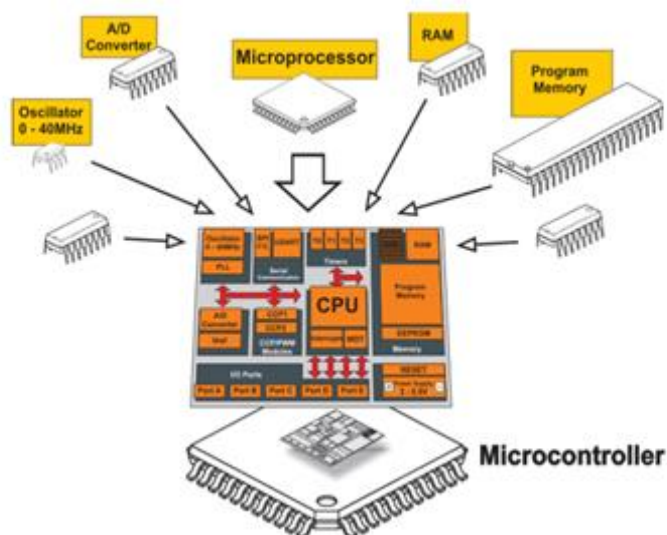


Figura 16 - Microcontroller vs. Microprocesor

3.4.2 Criterii de alegere a unui microcontrolle

Sunt mai multe aspecte de care trebuie ținut seama în vederea alegerii unui microcontroller pentru o anumită aplicație. Alegerea unui microcontroller potrivit poate duce la succesul proiectului, așa cum o alegere nepotrivită poate duce la eșecul proiectului.

În cele ce urmează voi face o clasificare a criteriilor pentru alegerea unui microcontroller, în ordinea importanței:

1) Posibilitatea folosirii în aplicația dată

- a. Este suficient un microcontroller sau sunt necesare circuite suplimentare.
- b. Microcontrollerul are viteza suficientă pentru dezvoltarea aplicațiilor. Se verifică timpul necesar rularii programului.
- c. Există capacitatea de memorare suficientă: RAM, ROM;
- d. Există toate interfețele solicitate de aplicație: I/O serial, convertoare A/D, D/A și nu există interfețe în plus;
- e. Alimentarea disponibilă în funcționarea aplicației și limitările acestei alimentări.
- f. Costurile permise pentru realizarea aplicației.
- g. Tipul conectivității necesare (Ethernet, USB, Wireless, etc.)
- h. Numărul și tipul de intrări și ieșiri.

2) Optenabilitatea microcontroller-ului

- a. Disponibilitatea în cantități suficiente.
- b. Trebuie să se afle în producția actuală, dar și în viitor pentru posibilitatea aprovizionării.
- c. Disponibilitatea accesoriilor (convertoare, alimentatoare, etc.).

3) Disponibilitatea suportului de dezvoltare

- a. Compilatoare;
- b. Debuggere;
- c. Module de evaluare;
- d. Emulatoare în circuit;
- e. Analizoare logice.

4) Suport din parte constructorului

- a. Documentatie tehnica;
- b. Service prin telefon (BBS);
- c. Software de utilizat;
- d. Existenta unor rapoarte despre problemele de functionare;
- e. Daca exista mai multi utilizatori de microcontroller, atunci sunt formate grupuri de lucru care pot oferi support.

Asa cum probabil ati observat, fiecare criteriu dintre cele enumerate, se afla in stransa legatura unul cu celalalt. Problema pe care o poate ridica acest aspect este legata de compromisurile pe care trebuie să le facă utilizatorul în dezvoltarea aplicației propuse. Procesul de cautare fiind foarte dificil din cauza numarului mare de microcontrollere de pe piata, dar si de alocarea unui buget cat mai avantajos pentru finalizarea proiectului. Fie ca vorbim de orice aplicatie practica sau de tehnologiile folosite, tot se fac compromisuri, dar ramane la latitudinea dezvoltatorului sa aleaga varianta cea mai potrivita.

3.5 Semnal digital

Un semnal digital este folosit în industria sistemelor încorporate având doar două valori logice, 0 și 1 care au câte o reprezentare în funcție de modul în care sunt transmise. În cazul microcontroller-ului folosit (ATMega328) acestea au pentru 0 logic tensiunea de 0 volți, iar pentru 1 logic tensiunea de 5 volți.

De cele mai multe ori în tehnică este de preferat să fie folosite semnale digitale în locul celor analogice deoarece transmisia acestora este mult mai puțin afectată de zgomote, iar conversia analog-digital este consumatoare de timp pe când trecerea unui semnal digital dintr-o stare logică în alta, este aproape instantanee, timpul petrecut între aceste tranziții fiind insesizabil ochiului uman.

Când semnalele digitale sunt folosite ca intrare pentru microcontroller pinul acestuia rămâne într-o stare de impedanță foarte ridicată dacă semnalul nu are una dintre valorile de 0 volți sau 5 volți și este între ele. Pentru a se evita această schimbare din cauza unui semnal inconsistent s-a folosit un trigger Schmitt.

Triggerul Schmitt asigură menținerea semnalului în 1 sau 0 în cazul unui semnal inconsistent și schimbând starea logică a acestuia doar când semnalul a depășit sau a scăzut peste pragurile de tensiune stabilite. Dacă tensiunea semnalului este între aceste praguri, starea logică a semnalului citit va rămâne neschimbată.

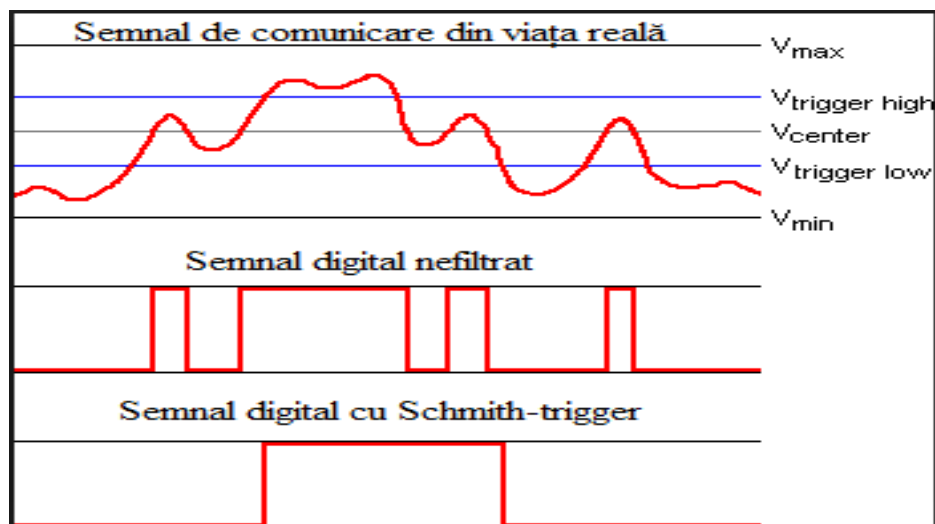


Figura 17 - Schmith Trigger

3.6 Semnal PWM

Termenul de PWM vine din engleză (pulse width modulation) și acest tip de semnal se folosește în general pentru a controla intensitatea luminoasă a unui LED\bec sau penru controlul vitezei de rotație a unui motor electric de curent continuu.

Caracteristici ale acestui semnal sunt:

- perioada
- frecvența
- factorul de umplere (duty cycle) – reprezentat de timpul pe care semnalul îl petrece în HIGH din perioadă.

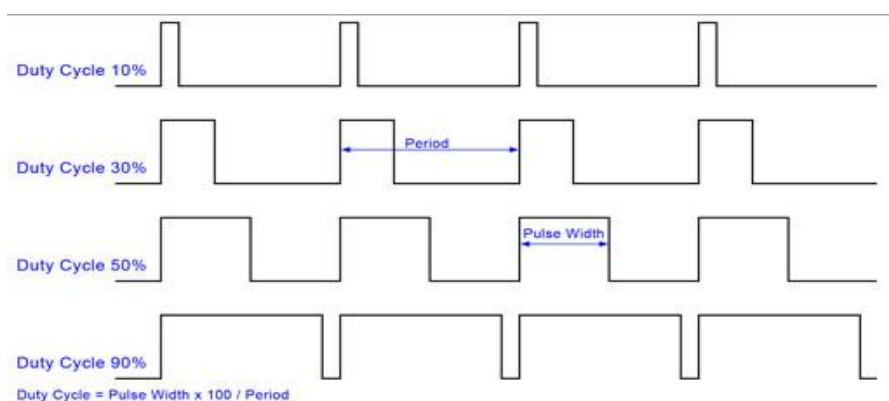


Figura 18 - Semnal PWM

3.7 Convertor Analog – Digital

Convertorul Analog-Digital reprezintă un bloc sau un circuit periferic al microcontrollerului care convertește un semnal analogic (tensiune, curent) într-o valoare numerică. Conversia se face pe baza aproximărilor succesive și stochează rezultatul într-un registru intern. Conversia analog-digital este consumatoare de timp, microcontroller-ul ATmega328 putând să facă această conversie într-un interval de 500 μ s până la 20 ms, aici rezultatul conversiei având cea mai mare acuratețe.

Acuratețea rezultatului conversiei constă în rezoluția convertorului. Microcontroller-ul ATmega328 are o rezoluție de 10 biți, astfel la o tensiune de 0 volți rezultatul conversiei va fi 0, iar la o tensiune de 5 volți rezultatul va fi 1023.

Se cunoaște faptul că un sistem pur analogic este capabil de o acuratețe mai bună decât un sistem analog-digital dar această acuratețe este foarte rar folosită în mod complet. Odată convertite, semnalele analogice pot fi procesate, sintetizate sau prelucrate matematic sau logic, pentru a putea fi interpretate de microcontroller, acesta neavând capacitatea de a scoate un semnal analogic.

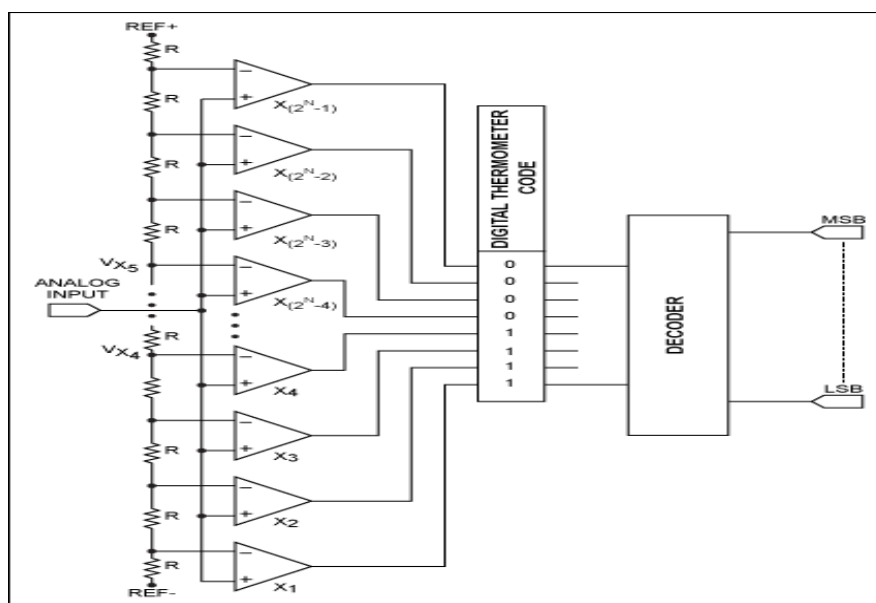


Figura 19 - Convertor Analog – Digital

4 Software

4.1 Comunicare I2C

Protocolul I2C a fost dezvoltat în 1982 de Philips pentru diferite cipuri Philips. Specificațiile originale permiteau comunicarea doar pe 100kHz și doar pe 7 biți de adresă, limitând numărul dispozitivelor conectate la magistrala bus, la 112. În 1992 spațiul de adrese a fost extins la 10 biți și comunicarea se realizează pe 500kHz. În prezent există trei moduri adiționale: fast plus (1MHz), high-speed (3.4MHz) și ultra-fast (5MHz).

În 1995 Intel a introdus o nouă variantă de I2C, numită "System Management Bus" (SMBus), care este mult mai bine controlată și realizată cu intenția de a maximiza predictibilitatea comunicării dintre suporturile I2C și plăcile de bază ale PC-urilor. Cea mai notabilă diferență dintre SMBus și I2C este că prima limitează viteza de la 10kHz la 100kHz, iar a doua poate suporta dispozitive de la 0kHz la 5MHz.

Protocolul Inter Integrated Circuit (I2C) este un protocol creat pentru a permite mai multor circuite integrate "slave" să comunice cu unul sau mai multe cipuri "master". Acest tip de comunicare poate fi folosit doar pe distanțe mici de comunicare și asemenea protocolului UART are nevoie doar de 2 fire de semnal pentru a trimite/primi informații.[32]

Dezavantajul porturilor seriale

Deoarece porturile seriale sunt asincrone, dispozitivele ce utilizează porturile seriale trebuie să conțină ceasuri ce au o rată apropiată de transmisie a datei, și comunicarea UART la bază, fiind o comunicare complexă și dificil de implementat.

Un alt minus în acest tip de comunicare este reprezentat de permisiunea de comunicare doar între două dispozitive. Chiar dacă este posibil să conectăm mai multe dispozitive la un singur port serial, apar conflicte la bus ("bus contention") deoarece două dispozitive tind să conducă aceeași linie în același timp și dispozitivele se pot strica.

În cele din urmă, rata datelor reprezintă o problemă. Teoretic nu este nicio limită în comunicarea serială asincronă, dar practic în comunicarea UART, dispozitivele suportă o rată de transfer fixată și cea mai mare rată este de aproximativ 230400 biți/secundă.[32]

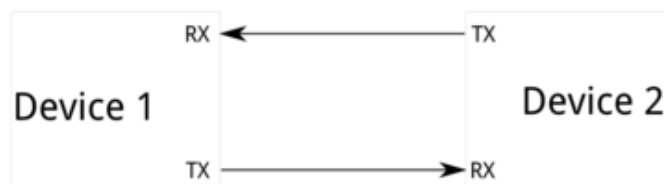


Figura 20 - Portul serial

Dezavantajul comunicării SPI

Cel mai mare minus pe care îl are un SPI este numărul mare de pini de care este nevoie. Pentru conectarea unui singur "master" cu un singur "slave" la un bus SPI, este nevoie de 4 linii, iar fiecare "slave" pe care-l conectăm după, necesită un cip aditional ce selectează un pin I/O de pe dispozitivul folosit ca "master". Prin urmare, în acest tip de comunicare este realizabilă conectarea a mai multor dispozitive "slave" la un singur dispozitiv "master", dar cu toate acestea, din cauza numărului mare de pini utilizați, transmiterea semnalelor este dificilă și încetă.

SPI este totuși util din cauza implementării ușoare și a ratei de transmitere a datelor (transmitere și recepționare simultană) prin conexiunea full-duplex, suportând o rată de transmisie de 10 milioane de biți/secundă.

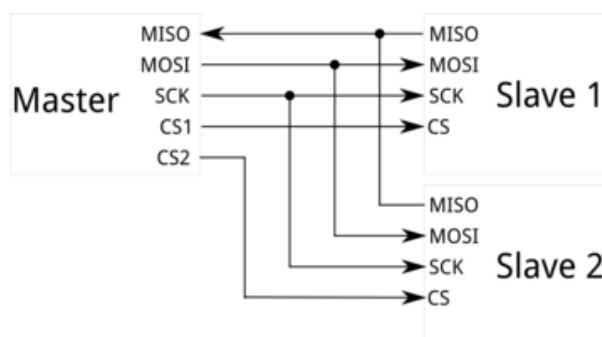


Figura 21 -Comunicarea SPI

Avantajul comunicării I2C

I2C necesită două fire ca și comunicarea serială asincronă, cu diferența că acest tip de comunicare poate suporta până la 1008 dispozitive de tip "slave". Mai mult decât atât, în comunicarea de tip I2C pot exista mai multe dispozitive de tip "master" la un bus, lucru nepermis în comunicarea SPI.

Rata datelor nu este foarte bună, fiind asemănătoare cu cea de la portul serial; cele mai multe dintre I2C-uri comunică cu o rată de transmisie cuprinsă între 100kHz și 400kHz.

În ceea ce privește implementarea, aceasta este mai complexă decât în cazul implementării SPI, dar mult mai ușoară decât în cazul comunicării asincrone, cu UART.

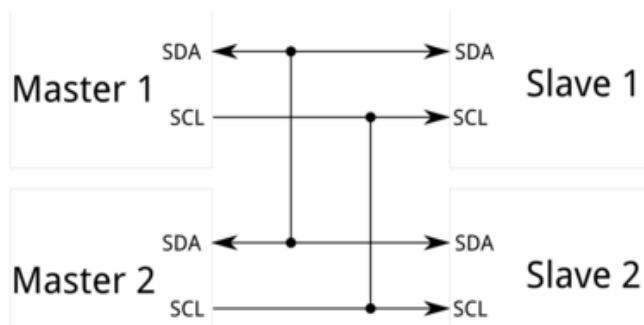


Figura 22 - Comunicarea I2C (a)

Fiecare bus I2C este compus din două semnale: SCL (semnalul de ceas) și SDA (semnalul de date). Semnalul de ceas este întotdeauna generat de bus-ul masterului curent. Fiecare bus I2C poate suporta până la 112 dispozitive, iar toate dispozitivele trebuie să distribuie GND.

Spre deosebire de alte metode de comunicare, precum UART, magistrala I2C este de tip "open drain", ceea ce înseamnă că poate trage o anumită linie de semnal în 0 logic, dar nu o pot conduce spre 1 logic. Așadar, se elimină problema de "bus contention", unde un dispozitiv încearcă să tragă una dintre linii în starea "high" în timp ce altul o aduce în "low", eliminând posibilitatea de a distruge componente. Fiecare linie de semnal are un rezistor pull-up pe ea, pentru a putea readuce semnalul pe "high" când nici un alt dispozitiv nu cere "low".[32]

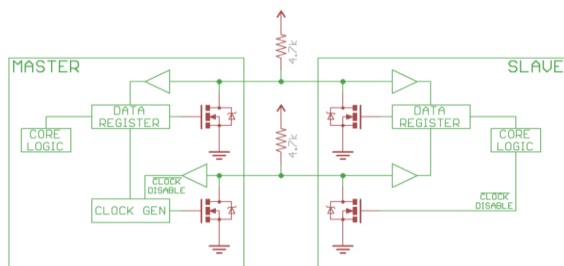


Figura 23 - Comunicarea I2C (b)

În imaginea alăturată se remarcă folosirea a două rezistențe pe liniile de semnal. Selecția rezistentelor variază odată cu dispozitivele care folosesc busul, dar o regulă bună este de a începe cu rezistențe de $4.7k\Omega$ și cu scăderea lor dacă e necesar.

4.1.1 Protocolul de comunicare I2C

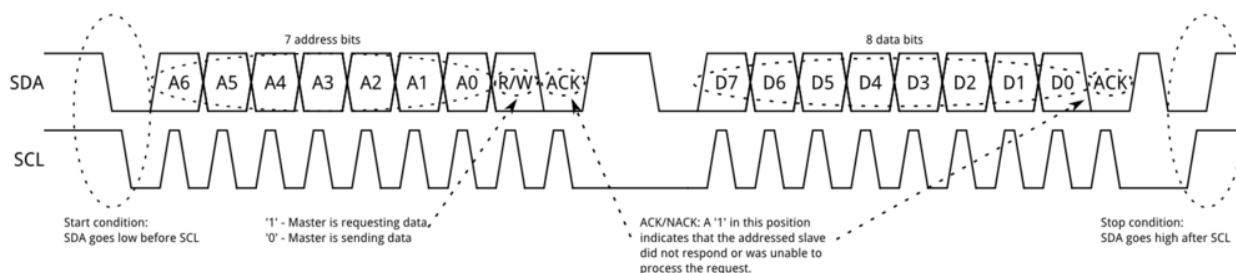


Figura 24 - Protocolul de comunicarea

Mesajele transmise sunt împărțite în două tipuri de cadre (frames): cadrele de date (conțin mesaje pe 8 biți direcționate de la dispozitivul de tip "master" la cel de tip "slave" și invers) și cadrele de adresă (conțin adresa dispozitivului de tip "slave", la care dispozitivul de tip "master" trimite mesajul).

Datele sunt puse pe linia SDA după ce SCL ajunge la nivel "low" și sunt eșantionate când SCL ajunge "high". Timpul între nivelul de ceas și operațiile de citire/scriere este definit de dispozitivele conectate pe magistrală și va fi diferit de la cip la cip.

Condiția de start

Pentru a iniția cadrul de adresă, dispozitivul "master" lasă SCL "high" și pune SDA pe "low", astfel toate dispozitivele "slave" sunt pregătite pentru a recepționa date.

Dacă două sau mai multe dispozitive "master" doresc să-și asume bus-ul la un moment dat, primul dispozitiv care ajunge de la "high" la "low" va prelua controlul bus-ului.

Cadrul de adresă

Cadrul de adresă este mereu primul atunci când o nouă comunicare începe. Mai întâi se vor trimite sincron biții adresei (primul fiind cel mai semnificativ), apoi se va transmite un semnal de tip R/W pe biți. Bitul 9 al cadrului este reprezentat de bitul NACK/ACK. El este prezent în ambele cadre (adrese și date).

După ce primii 8 biți ai cadrului sunt transmiși, dispozitivului receptor îi este dat controlul asupra SDA-ului. Dacă acest dispozitiv nu schimbă în 0 logic linia SDA înainte de al nouălea puls de ceas, se poate deduce că acesta nu a primit datele, ori nu a știut să interpreteze mesajul recepționat. În acest caz, schimbul de date se oprește și rămâne la latitudinea dispozitivului "master" cum va proceda mai departe.

Cadrul de date

După ce cadrul de adresă a fost transmis, datele/mesajele pot începe a fi transmise. Dispozitivul "master" va continua să genereze impulsuri de ceas la un interval regulat, iar datele vor fi plasate pe SDA, fie de dispozitivul "master" fie de cel "slave", în funcție de biții R/W transmiși în cadrul anterior. Numărul de cadre de date este arbitrar.

Cadrul de oprire

De îndată ce toate cadrele au fost trimise, dispozitivul "master" va genera o condiție de oprire. Condițiile de oprire sunt definite de tranziția de la "low" la "high" (0 -> 1) pe SDA, după o tranziție (0 -> 1) pe SCL, cu SCL pe "high".

În timpul operației de scriere, valoarea din SDA nu trebuie să se schimbe când SCL este "high" pentru a evita condițiile false de oprire.

Foarte mulți senzori utilizează I2C pentru a comunica, cum ar fi: barometrele, senzorii de temperatura, sonarele, MPU,etc. Este de precizat faptul că I2C nu este destinată cablurilor cu o lungime mare; în general cablurile de peste 2 metri pot cauza probleme.[32] I2C este un tip de comunicare complexă, dar este foarte folositoare.

4.2 Unity

Unity3d este un puternic motor 3D cross-platform, un mediu de dezvoltare ușor de utilizat. Destul de ușor pentru începători și suficient de puternic pentru experți. Unity ar trebui să fie interesant pentru oricine dorește să creeze cu ușurință jocuri și aplicații 3D pentru dispozitive mobile, desktop, web și console. Folosește în jur de trei limbaje de programare, precum C#, Boo și Javascript. Ofertă posibilitatea de a lucra în 3D și poate fi folosit de la Windows sau Mac până la sistemele de operare folosite pe telefoanele mobile, precum Android sau IOS. Unity ne permite să interacționăm cu spațiul de dezvoltare nu doar prin intermediul codului, ci și a componentelor vizuale.

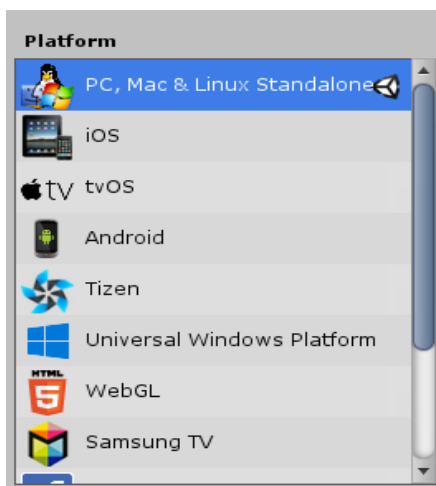


Figura 25 - Platforme disponibile în Unity[22]

După cum indică **Error! Reference source not found.**, Unity asigură suportul pe mai multe platforme și se pot schimba literalmente platformele cu un singur clic, deși este corect, de obicei există un efort minim necesar, cum ar fi integrarea cu fișierele necesare pentru fiecare platformă.

Microsoft și Unity lucrează în strânsă colaborare pentru a asigura suportul de platformă pe întregul nivel al Microsoft. Unity suportă executabile standalone Windows, Windows Phone, aplicații Windows Store, Xbox 360 și Xbox One.

Fiecare tip de conținut din Unity începe cu un GameObject. Orice obiect din joc este un GameObject: caractere, lumini, efecte speciale, elemente de recuzită, etc.

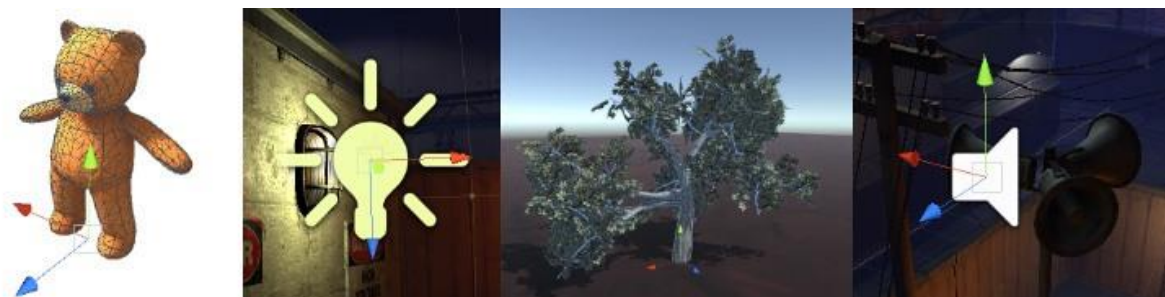


Figura 26 – GameObject [22]

GameObjects nu pot face nimic singure. Pentru a deveni ceva, pentru a avea o funcționalitate trebuie adăugată o proprietate GameObject-ului, lucru care se realizează adăugând componente pe obiectul respectiv.



Figura 27 - Proprietățile unui GameObject [23]

Componentele pot avea orice număr de proprietăți editabile care pot fi modificate prin fereastra Inspector din editor și / sau prin script. În exemplul de mai sus, unele proprietăți ale luminii sunt gama, culoarea și intensitatea. Componentele încorporate ale Unity sunt foarte versatile, dar pentru a implementa propria logică de joc se folosesc scripturile care trebuie aplicate obiectelor din joc.

Componentele de script vor permite obiectelor mai multe lucruri: declanșarea evenimentelor de joc, verificarea coliziunilor, aplicarea fizicii, răspunsul la intrarea utilizatorilor și multe altele.

Proiectele din Editorul Unity sunt alcătuite din mai multe gameplay-uri care conțin scripturi, sunete, și alte elemente grafice, cum ar fi lumini. Fereastra Inspector (uneori denumită "inspectorul") afișează informații detaliate despre GameObject-ul selectat în **Error! Reference source not found.**, inclusiv toate componentele atașate și proprietățile acestuia și permite să fie modificată funcționalitatea GameObject-ului în scenă.

Când este selectat un GameObject pentru vizualizare, Inspectorul arată proprietățile tuturor componentelor și materialelor acelui GameObject. Poate fi utilizat pentru a edita setările acestor componente și materiale.

Imaginea de mai sus arată inspectorul pentru GameObject-ul denumit "Point Light" la care s-a adăugat opțiunea Light. Toate proprietățile componente Light sunt disponibile pentru editare.

Când GameObject-urile au atașate componente de script personalizate, Inspectorul afișează variabilele publice ale acelui script. Se poate modifica valoarea acestor variabile în același mod în care se pot edita setările componentelor încorporate în Editor. Aceasta înseamnă că se pot seta cu ușurință parametri și valorile implicite în script-uri fără a modifica codul.

4.3 MonoDevelop

Un mediu de dezvoltare (integrated development environment - mediu integrat de dezvoltare) este un set de programe care ajută programatorul în scrierea programelor. Un mediu de dezvoltare combină toți pașii necesari creării unui program ca de exemplu: editarea codului sursă, compilarea, depanarea, testarea, generarea de documentație într-un singur soft, care, de regulă, oferă o interfață grafică cu utilizatorul destul de prietenoasă.

Principalele componente ale unui mediu de dezvoltare sunt editorul de cod sursă și depanatorul. Mediile de dezvoltare apelează compilatoare, sau interpretoare, care pot veni în același pachet cu mediul însuși, sau pot fi instalate separat de către programator. Printre facilitățile prezente în mediile de dezvoltare mai sofisticate se numără: exploratoare de cod sursă, sistemele de control al versiunilor, designere de interfețe grafice, sau unele din ingineria programării (ex.: generarea de diagrame UML) [30].

MonoDevelop este un mediu de dezvoltare integrat (IDE) furnizat împreună cu Unity. Acesta combină funcționarea familială a unui editor de text cu funcții suplimentare pentru depanare și alte sarcini de gestionare a proiectului. MonoDevelop este instalat în mod prestabilit cu Unity, deși există opțiunea de a-l exclude din instalare pe Windows.

Obiectivul său principal este dezvoltarea de proiecte care utilizează cadrele Mono și .NET. MonoDevelop integrează caracteristici similare cu cele ale NetBeans și Microsoft Visual Studio, cum ar fi completarea automată a codului, controlul sursei, interfața grafică (GUI) și designerul web. MonoDevelop integrează un designer grafic Gtk # numit Stetic [31]. Acesta susține Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Vala și Visual Basic.NET.

4.4 C#

Scrierea este un ingredient esențial în toate aplicațiile. Chiar și aplicațiile cele mai simple au nevoie de scripturi pentru a răspunde la intrarea utilizatorului și pentru a face ca evenimentele să se întâmple când ar trebui. Dincolo de aceasta, scripturile pot fi folosite pentru a crea efecte grafice, pentru a controla comportamentul fizic al obiectelor sau chiar pentru a implementa un sistem personalizat AI pentru personajele din jocurile video.

O mare parte a puterii Unity este în limbajul bogat de scripting, C#. Există posibilitatea de a fi utilizat pentru a gestiona intrarea utilizatorilor, pentru a manipula obiecte în scenă, pentru a detecta coliziuni sau pentru a crea noi GameObjects. Unity permite un script avansat de C# folosit pentru a compila jocuri video de tip cross-platform de compilatorul Unity.

C# este un limbaj de programare orientat pe obiecte care permite dezvoltatorilor să construiască o varietate de aplicații sigure și robuste. Sintaxa este una expresivă, simplă și ușor de învățat, fiind apropiată de C. C# este un limbaj de programare imperativ, obiect-orientat. Este foarte asemănător cu Java și C++, motiv pentru care, curba de învățare este foarte lină. C# suportă conceptul de încapsulare, ierarhie și polimorfism. Deși seamănă foarte mult cu limbajul de programare Java, ambele fiind limbaje orientate pe obiecte, C# simplifică conceptul de polimorfism prin înlăturarea moștenirii multiple.

Făcând o comparație între C# și Java, ambele limbaje au librării asemănătoare și se aseamănă și prin sintaxă, însă există și diferențe. De exemplu, două caracteristici majore din Java care lipsesc în C# sunt clasele imbricate și „checked exeptions”. În schimb C# a introdus clasele delegat.

4.5 ARDUnity



ARDUnity are un sistem comun de cod numit "ArdunityApp" și "ArdunityController", pentru ca componentele Arduino Sketches și Unity să interacționeze. Prin urmare, trebuie să fie utilizat sistemul comun pentru a face o aplicație ARDUnity.

Una dintre cele mai mari caracteristici ale ARDUnity este generarea automată a unei schițe utilă în activitatea Arduino. Unity este un mediu de dezvoltare bazat pe C#. Cu toate acestea, Unity are mai multe caracteristici care pot da rezultat, în cazul în care programatorul nu se descurcă foarte bine cu acest limbaj. Deoarece Arduino este un mediu de dezvoltare bazat pe C / C++, trebuie să știi limba C/C++ pentru a obține rezultate. De asemenea, este foarte dificil de programat, deoarece Unity și Arduino trebuie să creeze protocoale pentru a lucra împreună.

Modul de funcționare este prezentat în **Error! Reference source not found..**

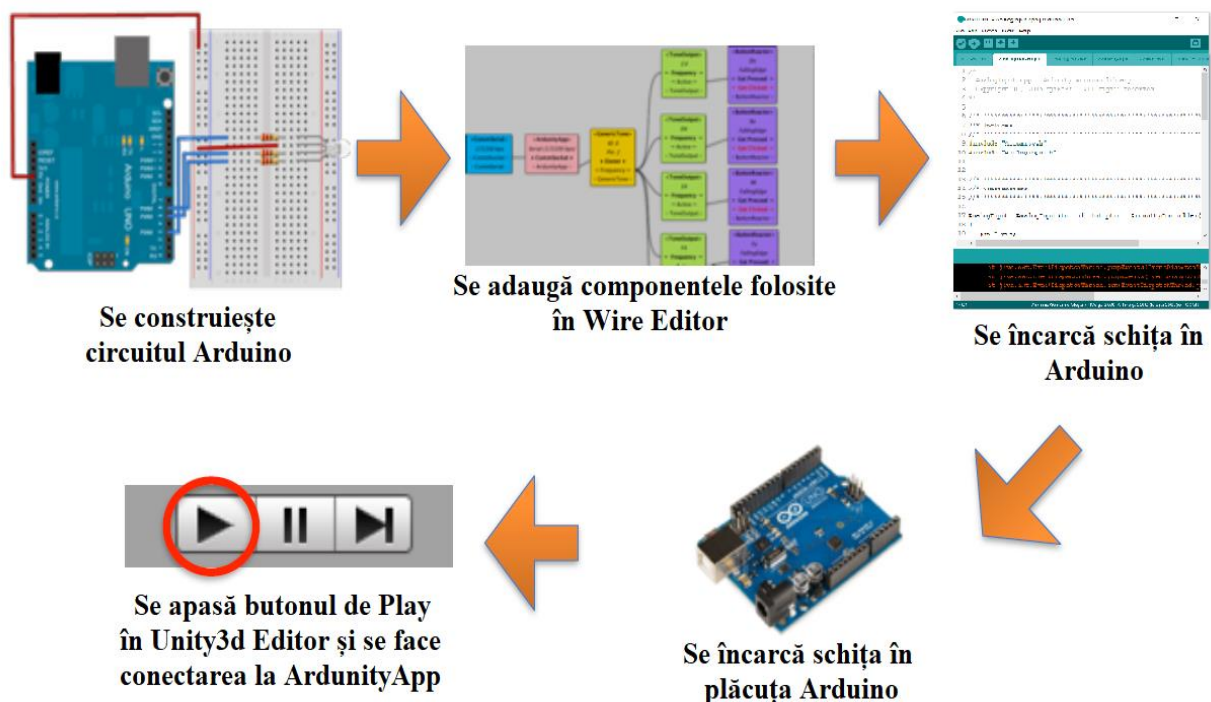


Figura 28 - Funcționare Ardunity

Pentru a extinde această structură, utilizatorul poate crea un cod derivat prin conceptul de moștenire. Sursele Arduino sunt scrise în C++ și creează fișiere Sketch la export pe baza

informațiilor de cablare. Clasa Arduino folosește biblioteca Arduino, când controlează hardware-ul. În consecință, aceste coduri funcționează bine fără hardware dependent de placă.

Arduino Sketch folosește o bibliotecă denumită "Stream" pentru toate bibliotecile ARDUnity care comunică cu exteriorul. Unity utilizează o varietate de metode pentru a comunica cu exteriorul, cum ar fi Serial, Bluetooth, WiFi etc. Pentru a le folosi cu ușurință, ARDUnity oferă o componentă, numită "CommSocket". Metoda și protocolul de comunicare, sunt complet separate, metoda de comunicare putând fi schimbată cu ușurință prin schimbarea "CommSocket". Asta înseamnă că, comunicațiile s-au schimbat cu ușurință schimbând doar "CommSocket", nefiind nevoie să se realizeze o nouă aplicație. [6]

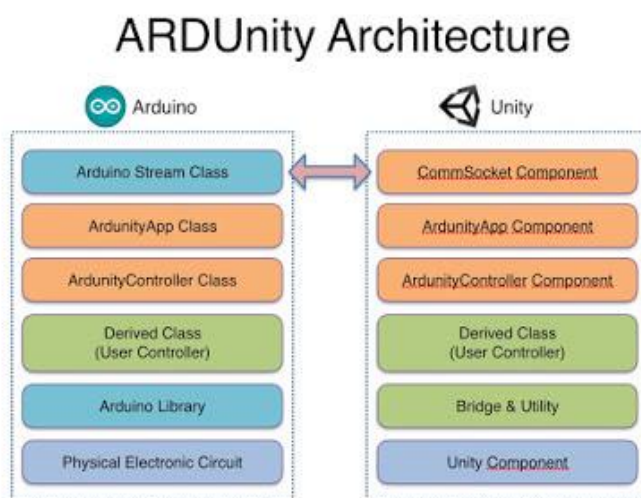


Figura 29 - Arhitectura ARDUnity [15]

Pentru a utiliza ARDUnity trebuie să se utilizeze "ArdunityApp" și "ArdunityController". "ArdunityController" poate fi derivat, de fapt, din funcții care sunt utilizate în mod obișnuit. Controlul fiind furnizat prin implementarea unei clase derivate. ARDUnity oferă funcții utile care pot procesa cu ușurință datele schimbate cu plăcile Arduino.

Caracteristicile sunt după cum urmează:

- Bridge: folosită pentru a procesa datele cu ușurință prin conectarea directă la controller în cea mai mare parte
- Reactor: folosit pentru a controla direct componentele Unity fără o programare separată
- Utility: caracteristică utilă în diverse aplicații

ARDUnity susține PlayMaker, introducând cu ușurință începătorii în acest mediu de dezvoltare. Odată ce programul este instalat, se pot vedea următoarele funcții:

Proxy: delegatul care transferă comanda Arduino la eveniment

Action: funcție ce trebuie utilizată pentru controller-ul în starea PlayMaker FSM

4.6 Android



Android este singurul sistem de operare mobil creat de Google. Transformă dispozitivul mobil într-un computer personal de buzunar. Android este echipat cu un browser Web complet și capacități de navigare pe Internet, acces la peste 80.000 de aplicații prin Android Market, inclusiv Gmail, Pandora și Facebook, plus capacitatea de a juca jocuri, de a trimite mesaje de tip SMS și de a efectua apeluri de pe telefonul.

Android este un software open source, ceea ce înseamnă că oricine poate face sistemul de operare mai bun. În acest fel, se beneficiază nu numai de cunoștințele dezvoltatorilor Google, ci și de cele ale dezvoltatorilor terță-partea. Se poate actualiza și telefonul Android existent la versiunea 2.2 și se pot obține aceleași beneficii. Android este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, dezvoltată inițial de compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance. Android permite dezvoltatorilor să scrie cod gestionat în limbajul Java controlând dispozitivul prin intermediul bibliotecilor Java dezvoltate de Google. Aplicațiile scrise în C și în alte limbaje pot fi compilate în cod mașină ARM și executate, dar acest model de dezvoltare nu este sprijinit oficial de către Google.

Android este structurat pe 4 niveluri, enumerate mai jos, de la bază spre vârf:

- **kernel-ul Linux;**
- **librăriile software** – conțin și modulul “runtime” ce permite rularea proceselor software pentru aplicațiile **Java**. Toate procesele sunt rulate în mașini virtuale independente (**Dalvik Virtual Machine**) optimizate pentru dispozitivele mobile. Aplicațiile pentru **Android** sunt compilate în formatul **.dex** – **Dalvik Executable** – ce este rulat de mașinile virtuale în momentul execuției.
- **application framework** – cadrul de dezvoltare pentru aplicații – conține clasele de obiecte necesare programatorilor în procesul de dezvoltare al aplicațiilor cum ar fi : clase de obiecte folosite pentru afișare – views, clase de obiecte folosite pentru accesarea datelor din alte aplicații – content providers, etc;

- **nivelul aplicație** – aici se găsesc aplicațiile software livrate odată cu sistemul de operare: clientul email, managerul SMS, managerul de apeluri și contacte, browser-ul web, etc. Toate aceste aplicații sunt scrise în Java.

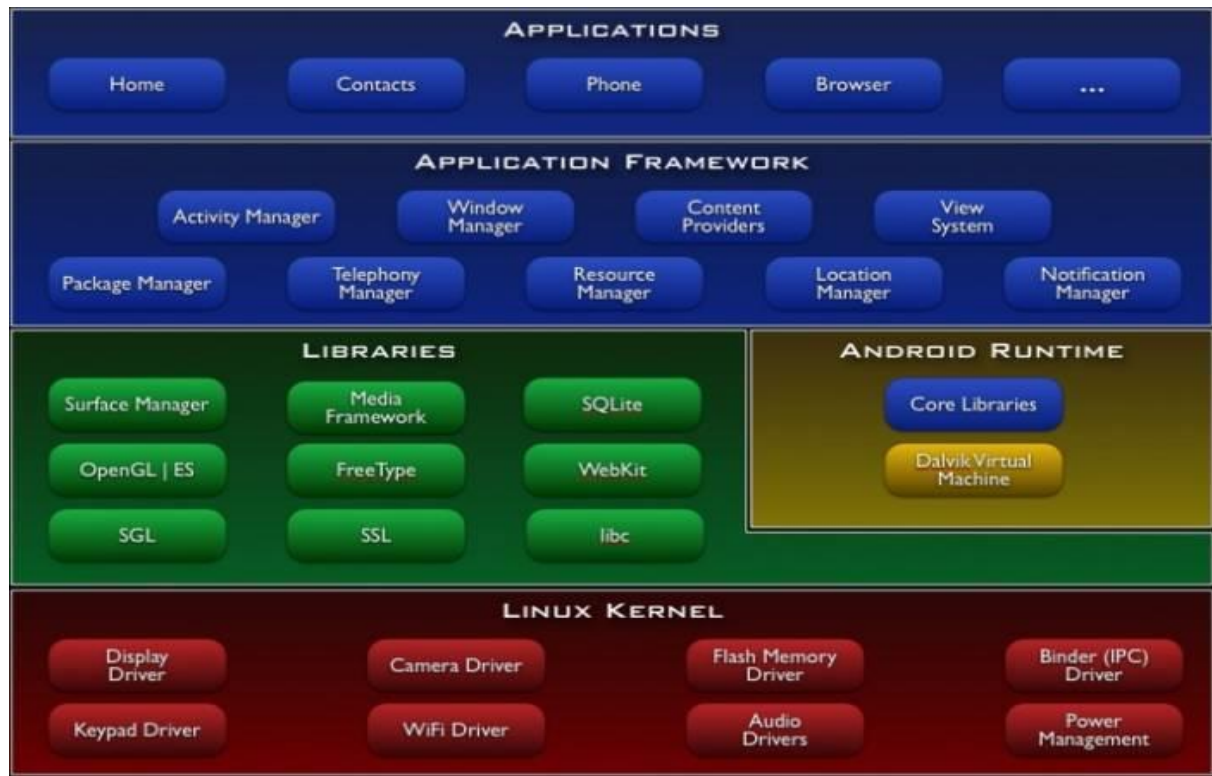


Figura 30 - Android structurat pe 4 niveluri

Faptul că **Android** este un sistem de operare cu sursă deschisă și licență gratuită, permite programatorilor să dezvolte rapid și cu ușurință aplicații care să aducă inovația în utilizarea dispozitivelor mobile.

4.6.1 Caracteristici și specificații ale sistemului Android

- **Configurații dispozitive.** Platforma este adaptabilă la configurații mai mari, VGA, biblioteci grafice 2D biblioteci grafice 3D bazate pe specificația OpenGL ES 1.0 și configurații tradiționale smartphone.
- **Stocare de date** Software-ul de baze de date SQLite este utilizat în scopul stocării datelor.
- **Conectivitate.** Android suportă tehnologii de conectivitate incluzând GSM/EDG, CDMA, EVDO, UMTS, Bluetooth și Wi-Fi.
- **Mesagerie instant** SMS și MMS sunt formele de mesagerie instant disponibile, inclusiv conversații de mesaje text.

- **Navigatorul de web.** Navigatorul de web disponibil în Android este bazat pe platforma de aplicații open source WebKit.
- **Mașina virtuală Dalvik** Software-ul scris în Java poate fi compilat în cod mașină Dalvik și executat de mașina virtuală Dalvik care este o implementare specializată de mașină virtuală concepută pentru utilizarea în dispozitivele mobile, deși teoretic nu este o Mașină Virtuală Java standard.
- **Suport media** Android acceptă următoarele formate media audio/video/imagine: MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF.
- **Suport hardware adițional** Android poate utiliza camere video/foto, touchscreen, GPS, accelerometru, și grafică accelerată 3D.
- **Mediu de dezvoltare** Include un emulator de dispozitive, unelte de depanare, profilare de memorie și de performanță, un plug-in pentru mediul de dezvoltare Eclipse.

4.6.2 Extensia APK

Fișierul Android cu extensia .apk este un pachet recunoscut de sistemul de operare Android pentru distribuirea și instalarea aplicațiilor mobile și a middleware-ului. Un fișier .apk conține tot codul unui program (cum ar fi fișiere .dex), resursele, activele, certificatele și fișierul manifest.

Fișierele .apk sunt un tip de fișier tip arhivă, în special în pachete de tip zip-type, bazate pe formatul de fișier JAR, cu extensia de nume.apk de fișier. Fișierele APK pot fi instalate pe dispozitivele Android, la fel ca instalarea software-ului pe un PC. Când un utilizator descarcă o aplicație Android utilizând un dispozitiv Android, dintr-o sursă oficială (cum ar fi Google Play Store), este instalată automat.

5 Dezvoltarea aplicației practice

5.1 Diagrama Use Case

Figura 31 prezintă schema de utilizare a proiectului, reprezentând fluxul de schimb de date între utilizator, Arduino și Unity.

În primul rând, utilizatorul mișcă mânușa, generând citirile pozițiilor degetelor și unghiul de orientare al mâinii. Aceste informații sunt primite mai apoi de Arduino care la rândul său citește datele de la senzorii de îndoire și de la MPU pentru a le trimite, prin intermediul Bluetooth -ului la aplicația Unity ce rulează pe un dispozitiv cu Android. Informația primită determină deplasarea modelului 3d cât și emiterea unui sunet împreună cu o imagine corespunzătoare a poziției degetelor.

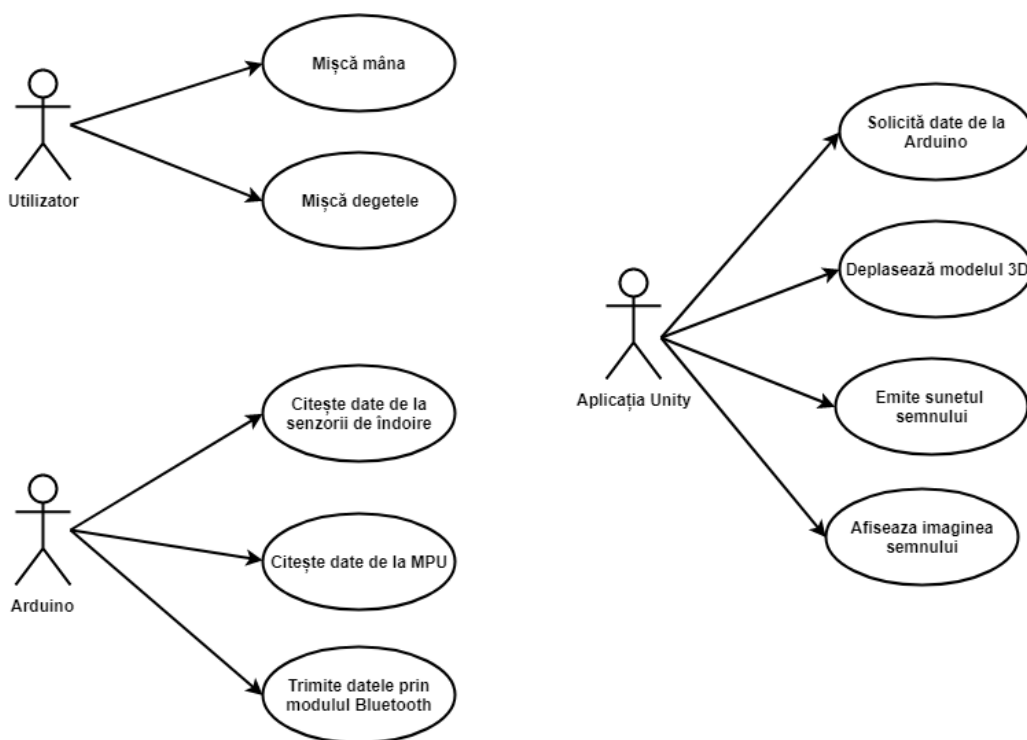


Figura 31 - a.Diagrama Use Case

5.2 Diagrama de secvență

Diagrama de secvență prezentată în Figura 32b, prezintă secvența de schimb de date dintre senzorii Arduino și Unity, de la captarea datelor, la afișarea rezultatelor pe ecran. De asemenea, se poate observa că cele mai recente date sunt trimise numai la Unity când are loc actualizarea cadrului. Acest lucru se face pentru a prevenirea blocării în tamponul de intrare al portului serial a valorilor senzorilor, ceea ce poate duce la un răspuns lent al modelului 3d.

Mișcarea mâinii utilizatorului și a degetelor în raport cu antebrațul generează poziția modelului tridimensional (3d). Valorile de îndoire sunt citite de Arduino prin `Analog.read()` și valorile unghiurilor de orientare ale mâinii sunt recepționate de către funcția `mpu.dmpGetQuaternion()` în Arduino de la MPU-6050. În timpul actualizării fiecărui cadru în Unity, un caracter este trimis prin portul software serial care are funcția de a solicita datele senzorilor curenți. Arduino citește apoi acest caracter și trimite datele prin portul serial pentru Unity, ele fiind folosite pentru a manipula obiectul 3d pe ecranul dispozitivului, prin intermediul funcției `transform.localEulerAngles()`.

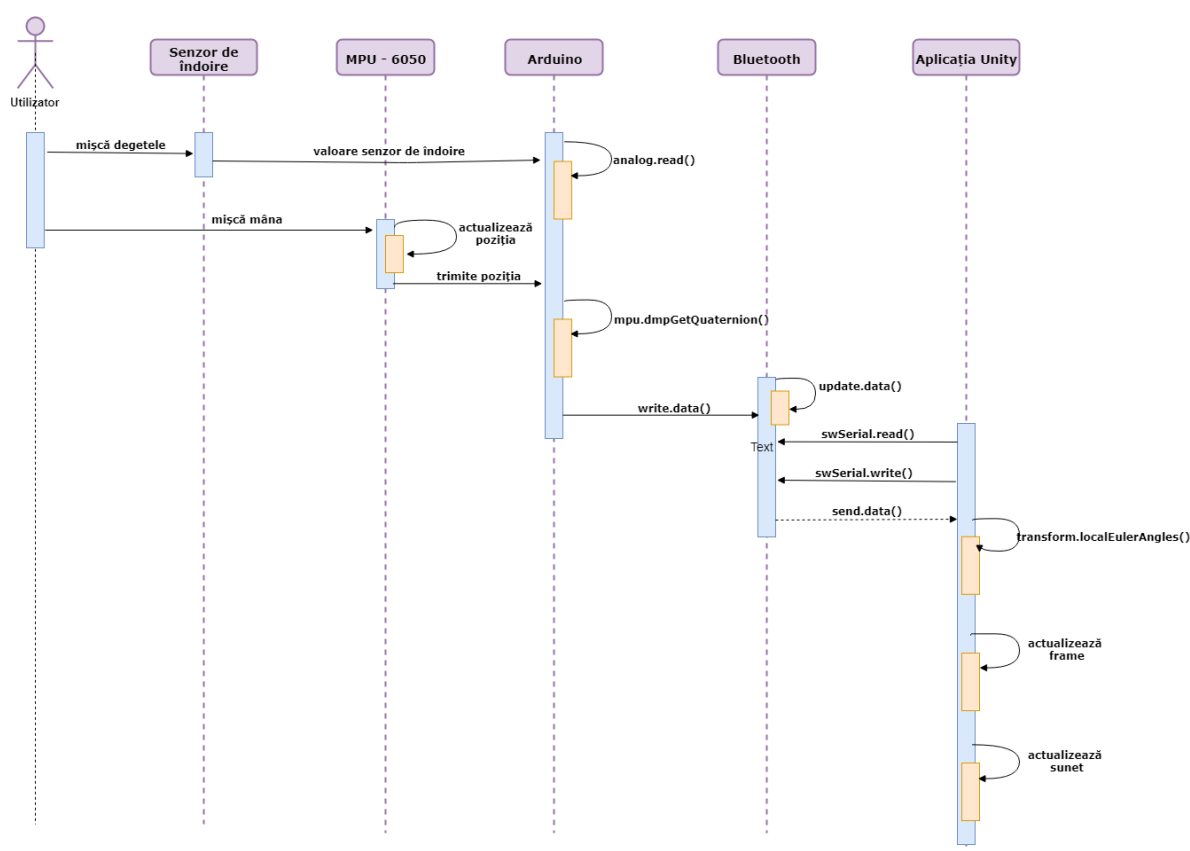


Figura 32 - b. Diagrama de secvență

Semnalul de orientare trimis de Arduino este destul de zgomotos, provocând instabilitate în modelul 3D și tremurând. Acest zgomot provine de la accelerometru, deoarece este inexact în a furniza date privind unghiul de orientare a senzorului, astfel încât citirile accelerometrului nu pot fi folosite singure. Pe de altă parte, giroscopul oferă date de unghi stabile, dar erorile mici de citire se acumulează în timp, furnizând date incorecte (cunoscute sub numele de drift).

Soluția este de a folosi valorile accelerometrului și giroscopului împreună pentru a obține valori mai apropiate de cele reale. IMU MPU-6050 are un procesor intern numit Digital Motion Processor™ (DMP), care realizează îmbinarea și filtrarea valorilor obținute de la accelerometru, giroscop și

magnetometru, asigurând orientarea IMU fără necesitatea de a utiliza procesorul Arduino pentru aceasta.

Funcționarea corectă a mânușilor și reducerea zgomotului este efectuată prin observarea valorilor mediei, deviației standard (DS), amplitudinii și deviației medii absolute (DMA) a valorilor brute primite de la senzori.

5.3 Mâna fizică

Dezvoltarea acestei lucrări a avut loc în două etape: asamblarea Hardware și implementarea Software-ului. Fiecare dintre acești pași va fi descris în continuare.

Cerințele au fost concepute cu scopul de a defini domeniul de activitate, oferind o bază pentru efectuarea testelor și pentru a ajuta la verificarea rezultatelor atunci când dispozitivul este complet.

Cerințele hardware pentru acest proiect sunt:

- a) detectarea mișcărilor de întindere și strângere a degetelor cu o eroare maximă de $\pm 15^\circ$;
- b) detectarea mișcării de rotație a mâinii utilizatorului cu o eroare maximă acceptată de $\pm 10^\circ$;
- c) Nici o componentă a mânușilor nu poate împiedica mișcarea mâinii utilizatorului.

Cerințele software pentru acest proiect sunt:

- a) reproducerea mișcărilor degetului utilizatorului de către modelul 3D;
- b) rotația modelului 3D în funcție de orientarea mâinii utilizatorului;
- c) mișcări netede ale modelului 3D.
- d) interacțiunea naturală a modelului 3D cu celelalte obiecte din spațiul virtual.

6 Dezvoltarea Hardware

Pentru a realiza mănușa electronică s-au folosit următoarele componente:

- Placa de dezvoltare Arduino Uno R3;
- O baterie de 9V folosită ca sursă de current;
- Un modul Bluetooth;
- Cinci senzori de îndoire la care s-a atașat fiecare câte o rezistență de 10k Ω ;
- Un IMU-MPU6050;
- Un breadboard;
- Fire de conexiune, arcuri, mănușa din neopren, fire care sa acționeze cursorul, carcasa;

În continuare va fi descris modul de utilizare al fiecărei componente:



Figura 33 - Arduino Uno R3

La placa de dezvoltare **Arduino Uno R3** vor fi conectate toate componentele mănușii electronice. Alimentarea se face printr-o baterie de 9V. Codul sursă prin care se programează microcontroller-ul și citirea informațiilor obținute de senzori se realizează prin intermediul modulului Bluetooth.

Mai multe informații despre Arduino și modul de utilizare au fost prezentate în Capitolul 3.3.4- Arduino.



MPU6050 este o unitate de măsurare inerțială. Acesta are rolul de a detecta mișcările mâinii utilizatorului ca în figura prezentată mai jos (**Error! Reference source not found.**). Mai multe informații despre MPU6050 au fost prezentate în capitolul 3.3 - Unitatea de măsurare inerțială (IMU).

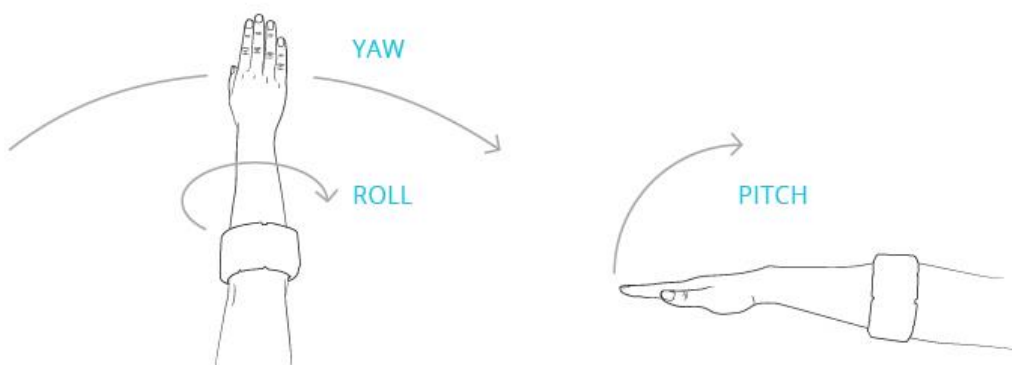


Figura 34 - Rotația [17]

Pinii de la MPU6050 au fost conectați la placuța Arduino astfel:

- VCC conectat la 5V;
- GND conectat la GND;
- SCL conectat la SCL (pinul 21);
- SDA conectat la SDA (pinul 20);
- INT conectat la pinul 2 (pin PWM);

Modulul Bluetooth

Modulul Bluetooth HC-05 / HC-06 comunică cu Arduino prin interfața UART. Fiecare mesaj pe care Arduino dorește să-l trimită este dat mai întâi modulului Bluetooth, care trimite mesajul fără fir. Pentru a evita problemele cu UART, Arduino și Bluetooth-Module trebuie să utilizeze aceeași baud-rate (la 9600 implicit). Este posibil să modificați baud-rate și parola modulului HM-10.

Cablare HM-10:

- GND din HM-10 la GND Arduino
- VCC de HM-10 la 3.3V Arduino sau 5V cum permite modulul
- TX HM-10 la Arduino Pin 10 (RX)
- RX HM-10 la Arduino Pin 11 (TX)

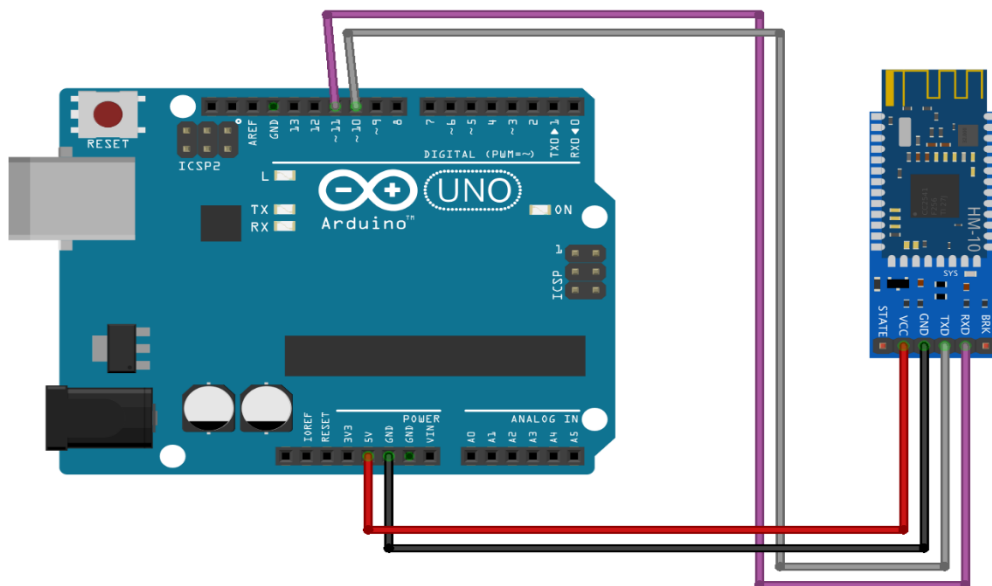
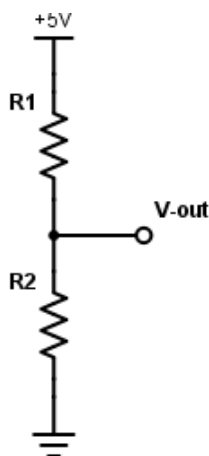


Figura 35 - Schema de conectare Android si Bluetooth HM-10

Senzorii de îndoire



Mai multe informatii se pot gasii in capitolul Senzori de la pagina 6.

Cel mai simplu mod de a încorpora acesti senzori în proiectul nostru este prin folosirea unui divizor de tensiune. Astfel acest circuit necesită un rezistor suplimentar. Prin combinarea unui senzor de îndoire cu un rezistor static se obține un divizor de tensiune, pentru a obține o tensiune variabilă ce poate fi citită de convertorul analog-digital al unui microcontroller. Divizorul de tensiune este foarte important cand vine vorba despre Arduino.

Cu ajutorul celor doua rezistențe se poate obtine orice voltaj între voltajul de intrare și GND.

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$

Formula de mai sus este aplicată divizorului de tensiune. V-in reprezinta alimentarea circuitului și are o tensiune de 5V. Rezistența R_1 va avea o valoare diferită în funcție de poziția senzorului de îndoire, mai exact în punctul V-out se pote citi orice o valoare cuprinsă între 625mV

și 1.25V deoarece rezistența R_1 va avea o valoare diferită în funcție de poziția senzorului de îndoire. Rezistența R_2 are valoarea constantă de 10k Ohm.

Astfel se va citi un semnal analogic diferit la pinul Arduino când degetul utilizatorului se va mișca. Semnalul analogic va fi procesat de Arduino, convertit în semnal digital, iar valoarea respective va fi transmisă în spațial virtual al aplicației ce rulează pe Android prin intermediul modulului Bluetooth.

În imaginea următoare este reprezentată schema bloc a unui sensor de îndoire:

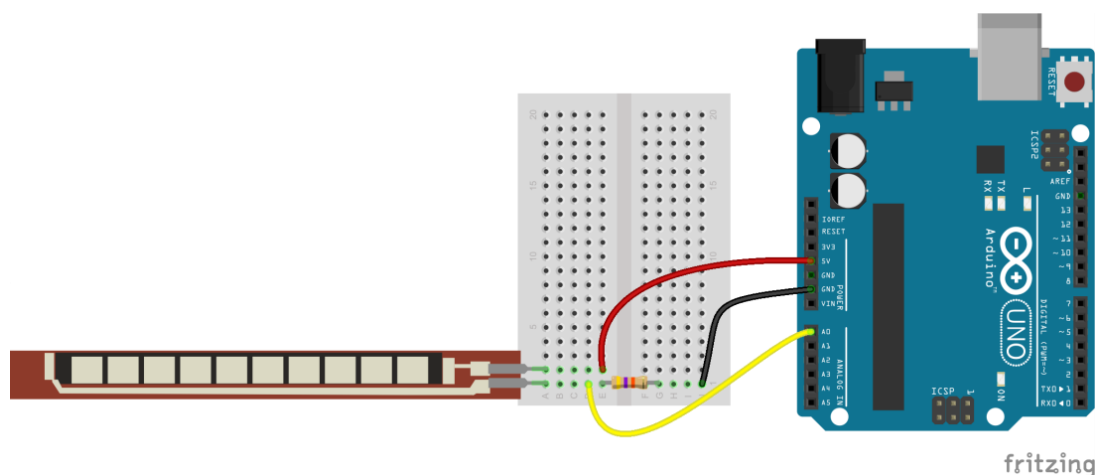


Figura 36 - Schema de conectare a Senzorului de îndoire la Arduino

6.1 Schema bloc

Schema din Figura 37 reprezintă arhitectura de proiectare, prezentând conexiunile între fiecare componentă. Arduino citește datele de la fiecare senzor și datele de orientare de la IMU, și le trimite prin modulul bluetooth la aplicația care rulează pe Android.

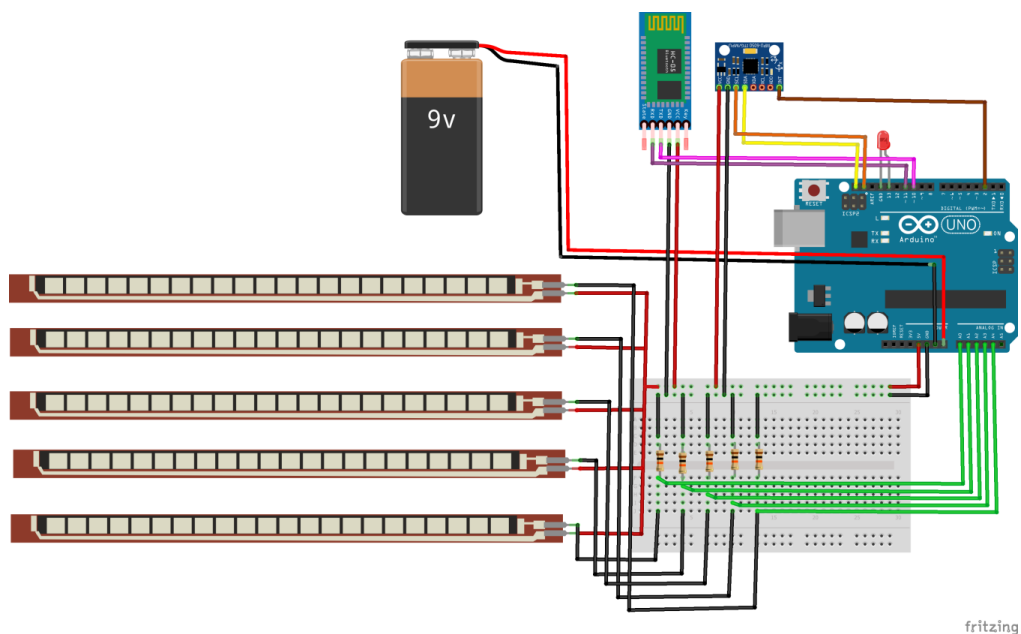


Figura 37 - Schema bloc

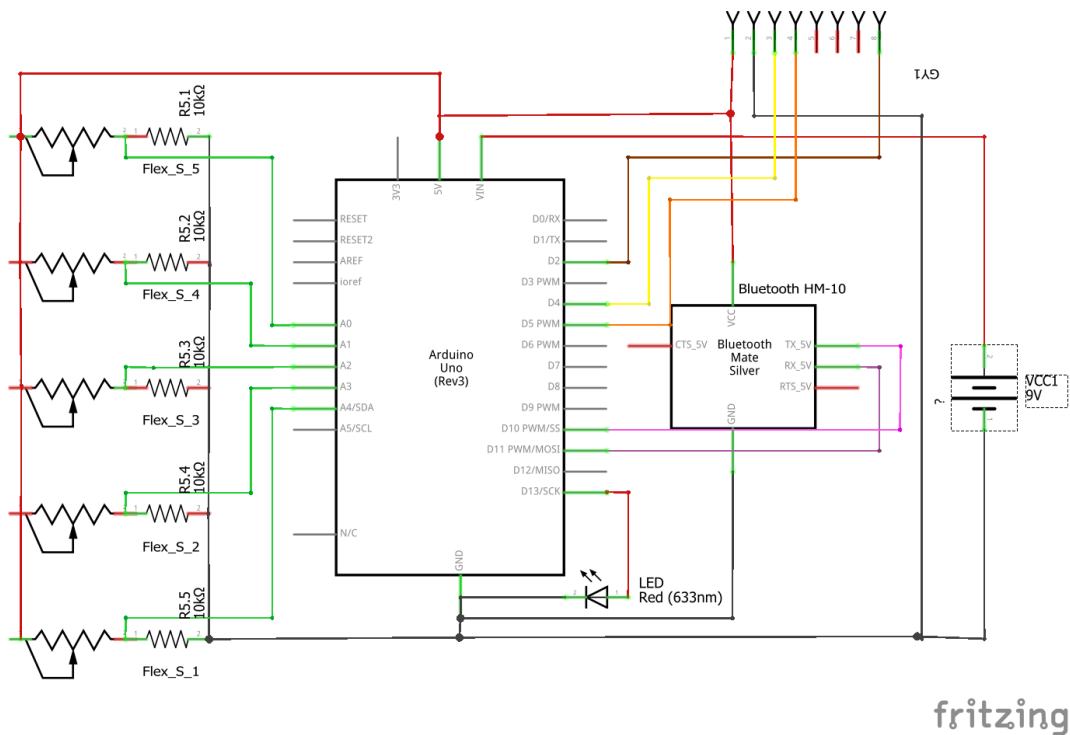


Figura 38 - Schema electrică

6.2 Mănușă electronică

În imaginea de mai jos este prezentată mănușă electronică. Se pot observa senzorii de flexare atașați pe fiecare deget, dar și celelalte componente electronice (modulul HM10, modulul MPU6050, bateria dar și firele de conexiune)



Figura 39 - Mănușă electronică - Privire de sus



Figura 40 - Mănușă electronică - Privire laterală

7 Dezvoltarea Software

În această etapă, au fost implementate codurile Arduino și Unity. **Error! Reference source not found.**, arată funcționarea sistemului în ansamblu.

Arduino inițializează unitatea IMU și DMP-urile lor respective. Atunci când sunt inițializate, datele de la senzorii de îndoire sunt actualizate până la o întrerupere din partea MPU-6050. Când se întâmplă acest lucru, valorile de orientare sunt actualizate și trimise către Unity, dacă s-au cerut cele mai actuale valori. Unity începe executarea prin deschiderea unei conexiuni a modului Bluetooth pentru primirea datelor Arduino. Fiecare actualizare de cadru trimite un caracter prin portul software serial indicând faptul că Unity are nevoie de cele mai recente date pentru a actualiza aplicația. După primirea și împărțirea șirului care conține datele, poziția fiecărui element este actualizată.

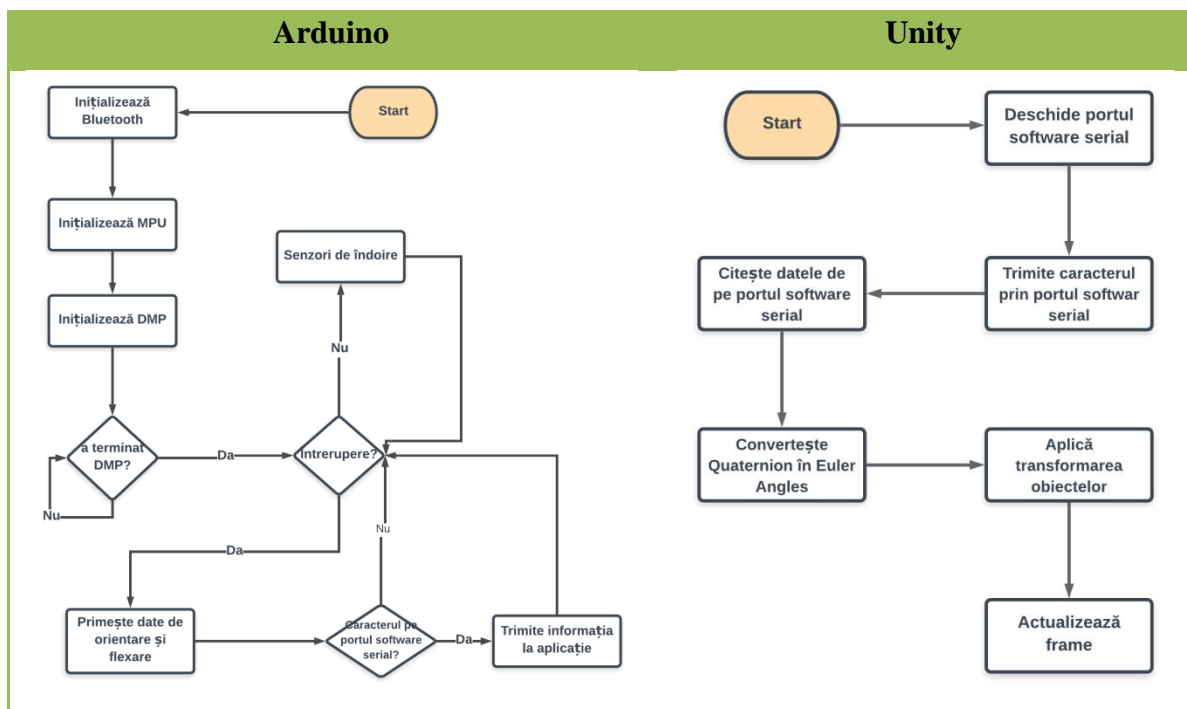


Figura 41 - Diagrama de procesare

Un alt aspect esențial care contribuie la experiența dezvoltării în Unity este prezența Asset Store-ului. Asset Store este o platformă de cumpărare a unor elemente concepute de comunitate. Un asset poate fi un grup de modele, sunete, scripturi, texturi, etc ce formează un tot unitar, ce poate fi atașat cu ușurință într-o aplicație Unity. În această aplicație este folosit asset-ul ARDUnity Deluxe.

7.1 Codul Arduino

Codul Arduino este responsabil pentru citirea valorilor senzorialor de îndoire, a IMU-ului și de trimiterea acestora prin portul software serial creat de Bluetooth către aplicația Unity ce rulează pe Android. Pentru achiziția de date de la IMU, a fost folosită biblioteca ArdunityI2C oferită de plug-in [7], bibliotecă asemănătoare cu I2Cdevlib, produsă de Rowberg (2016) [8], ce facilitează comunicarea prin protocolul I2C, fiind un protocol utilizat de IMU și biblioteca Kharlashkin (2016) [9], care permite utilizarea și setarea DMP-ului din MPU-6050.

În **Error! Reference source not found.**, se verifică dacă s-au putut citi informațiile de la MPU-6050, iar dacă acest lucru s-a efectuat cu succes, se salvează în variabile predefinite unghiul de rotație al quaternionilor XYZW.

```
void MPUSeries::OnProcess()
{
    if(started && _initialized)
    {
        if(update()) // Successful MPU DMP read
        {
            _qX = mympu.xyzw[0];
            _qY = mympu.xyzw[1];
            _qZ = mympu.xyzw[2];
            _qW = mympu.xyzw[3];

            dirty = true;
        }
    }
}
```

Codul 1 - MPU process()

Pentru a citi valorile obținute de la senzorii de îndoire s-a folosit "AnalogInput", clasă gândită și pentru a reduce zgomotul produs de intrările analogice ale Arduino, cu reducere minimă de receptivitate a semnalului. Funcția detectează dacă s-a produs o schimbare a senzorului, iar dacă s-a produs, salvează noua valoare și anunță modificarea efectuată.

```
void AnalogInput::OnProcess()
{
    if(started)
    {
        FLOAT32 newValue = (FLOAT32)analogRead(_pin) /
(FLOAT32)ArdunityApp.maxADC;
        if(_value != newValue)
        {
            _value = newValue;
            dirty = true;
        }
    }
}
```

Codul 2 - SenzorÎndoire process()

Comunicare între componentele ce sunt conectate la Arduino se realizează prin protocolul de comunicare I2C. Mai multe informații despre acest protocol se găsesc la capitolul 4.6. Codul de mai jos prezintă corpul funcțiilor principale utilizate pentru a realiza comunicarea dintre component.

```
boolean ArdunityI2C::Read(uint8_t* buffer, uint8_t length)
{
    Clear();

    Wire.requestFrom(address, length);

    unsigned long time = millis();
    int num = Wire.available();
    while(millis() - time < 2)
    {
        if(num >= length)
        {
            for(uint8_t i=0; i<length; i++)
                buffer[i] = Wire.read();

            return true;
        }

        if(num != Wire.available())
        {
            num = Wire.available();
            time = millis();
        }
    }

    return false;
}

boolean ArdunityI2C::Write(uint8_t reg, uint8_t* data, uint8_t length)
{
    Wire.beginTransaction(address);
    Wire.write(reg);
    for(uint8_t i=0; i<length; i++)
        Wire.write(data[i]);
    if(Wire.endTransmission() > 0)
        return false;

    return true;
}
```

Codul 3 – Corpul funcției principale()

În Codul 4 sunt prezentate inițializările obiectelor necesare pentru Arduino.

```
SoftwareSerial swSerial(10, 11);
AnalogInput aInput2(2, A4);
AnalogInput aInput5(5, A0);
AnalogInput aInput3(3, A2);
AnalogInput aInput4(4, A1);
AnalogInput aInput1(1, A5);
signed char orient_mpu6050_0[9] = { 1, 0, 0, 0, -1, 0, 0, 0, -1 };
```



```

MPUSeries mpu6050_0(0, MPU6050, false, orient_mpu6050_0);
DigitalOutput dOutput6(6, 13, LOW, true);

void setup()
{
  ArdunityApp.attachController((ArdunityController*)&aInput2);
  ArdunityApp.attachController((ArdunityController*)&aInput5);
  ArdunityApp.attachController((ArdunityController*)&aInput3);
  ArdunityApp.attachController((ArdunityController*)&aInput4);
  ArdunityApp.attachController((ArdunityController*)&aInput1);
  ArdunityApp.attachController((ArdunityController*)&mpu6050_0);
  ArdunityApp.attachController((ArdunityController*)&dOutput6);
  ArdunityApp.resolution(256, 1024);
  ArdunityApp.timeout(5000);
  swSerial.begin(9600);
  ArdunityApp.begin((Stream*)&swSerial);
}

void loop()
{
  ArdunityApp.process();
}

```

Codul 4 - Inițializări Arduino

În codul de mai sus au fost de asemenea declarate variabilele globale necesare monitorizării IMU-ului, calcularea poziției cursorului pe rezistența senzorilor și trimiterea șirului prin portul software serial.

În funcția setup (), prezentată în Codul 4 - Inițializări Arduino

, s-a folosit:

- ArdunityApp.resolution(256, 1024) – configurează rezoluția pentru semnalul maxim PWM = 256 și semnalul maxim ADC=1024;
- ArdunityApp.timeout(5000) – timpul de așteptare, după care se vor citi noile informații de la senzori;
- ArdunityApp.begin(9600) – comunicarea software serială a fost inițializată cu o rată de transfer de 9600 bps;

În funcția loop(), prezentată în **Error! Reference source not found.**, se activează toate procesele, lucru care se repetă la infinit.

```

void loop()
{
  ArdunityApp.process();
}

```

Codul 5 - Arduino loop()

7.2 Spațiul virtual

Pentru a cerceta și dezvolta aplicabilitatea modelului 3d am stabilit să dezvolt o aplicație cu scopul de a ajuta persoanele cu dizabilități să comunice mai ușor. Acest lucru s-a realizat cu ajutorul codului disponibil în ArdUnity dar și prin construirea de scripturi în limbajul C# pentru a programa mișcarea mâinii și implicit a degetelor. De asemenea, în funcție de mișcare a mâinii și a degetelor în spațiul virtual va fi emis un sunet și va fi afișată o imagine aferentă.

Pentru a anima și a afișa modelul 3D, am ales să folosesc platforma Unity3d, versiunea 2019.1.0. Fiind un mediu de dezvoltare pentru jocuri digitale, este compatibil atât cu Oculus VR, cât și cu dispozitivele Arduino.

Alegerea acestei platforme s-a bazat pe ușurința de învățare, utilizare și pe varietatea de modele 3D puse la dispoziție de comunitate.

Modelul 3d utilizat a fost preluat din proiectul lui Semihsmg cu numele GloVR, acest proiect fiind disponibil aici [21].

Pentru mișcarea modelului, a fost implementat plug-in-ul ArdUnity Deluxe[18], disponibil în magazinul Unity. Prin utilizarea sau modificarea pachetului de scripturi C# oferite de acest plugin, modelul 3d primește datele fiecărui senzor și valorile obținute de la IMU prin Bluetooth și aplică transformările degetului corespunzător și rotația modelului 3d. Aplicația Unity afișează reprezentarea mâinii utilizatorului într-un model 3d care reproduce mișcările efectuate. Astfel, se poate realiza o comunicare cât mai eficientă cu orice persoană.

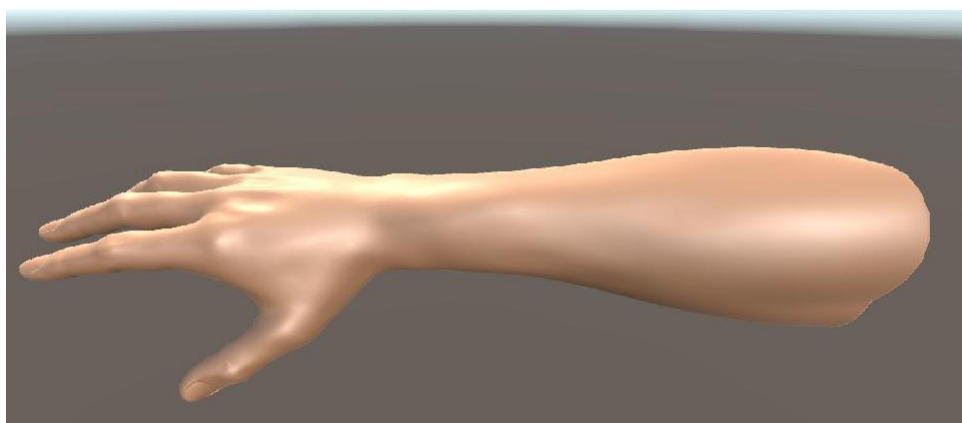


Figura 42 - Modelul 3d din Unity (1)

Mâna virtuală este concepută din trei obiecte principale (GameObject), la care s-au adăugat scripturi pentru a avea comportamentul dorit atunci când se citesc valorile trimise de Arduino.

Pe primul obiect este atașată componenta Skinned Mesh Renderer, componentă predefinită în Unity. Aceasta folosește și componenta Renderer pentru a face animații osoase, acolo unde forma obiectului este deformată de secvențe de animație predefinite. Această tehnică este utilă pentru

caractere și alte obiecte a căror articulații se îndoaie (spre deosebire de o mașină în care articulațiile sunt mai mult ca balamalele)[22].

Un mesh este un grup de triunghiuri aranjate într-un spațiu tridimensional astfel încât se creează imaginea unui obiect solid. Un triunghi este definit de trei puncte sau vârfuri. Noțiunii de mesh îi corespunde și o clasă numită sugestiv "Mesh". În această clasă, triunghiurile sunt stocate printr-un vector ce conține toate vârfurile; fiecare triunghi este definit de câte trei elemente ale vectorului. Astfel elementele 0, 1 și 2 ale vectorului ar reprezenta primul triunghi, 3, 4 și 5 ar reprezenta al doilea triunghi, etc[23].

La al doilea obiect principal este atașată componenta Animator, predefinită în Unity. Această componentă este utilizată pentru a atribui animație unui GameObject în scenă. Componenta Animator necesită o referință la un Controller Animator care definește clipurile de animație pe care le folosește și controlează când și cum să interacționeze între ele.

Al treilea obiect principal este construit din mai multe GameObject, care îndeplinesc funcția de schelet al mâinii. Atunci când un deget este îndoit, se produce rotirea a trei GameObject care alcatuiesc acel deget, iar prin componenta Skinned Mesh Renderer se produce efectul de întindere al pielii.

Conectarea la plăcuța Arduino este realizată de un obiect la care s-au adăugat scripturile "HM10" și "Ardunity App".

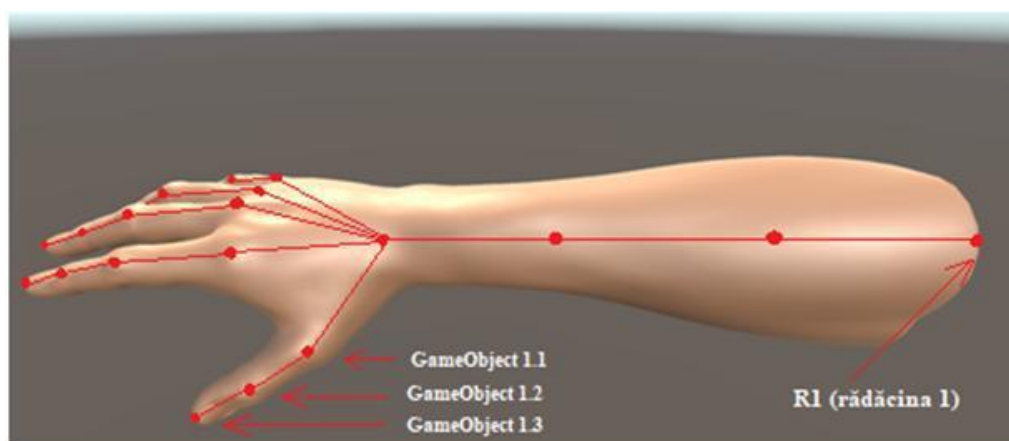


Figura 43- Modelul 3d din Unity (2)

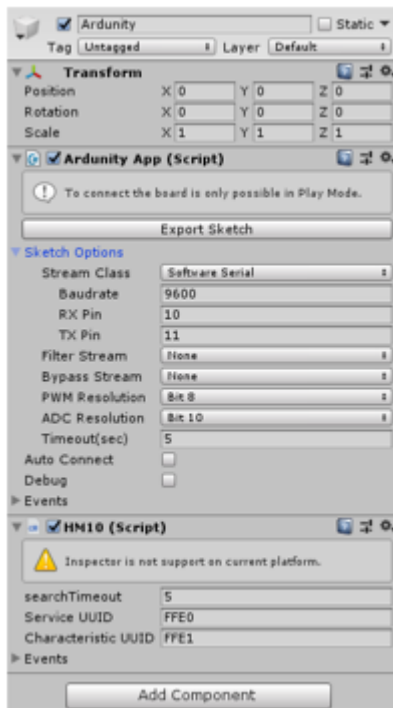


Figura 44 – Comunicarea

Butonul "Export Sketch" generează codul pentru Arduino configurat după modul în care au fost făcute toate setările, inclusiv citirea pinilor pentru senzorii de îndoire, modulul Bluetooth și MPU-6050.

Pentru a seta pinul de pe care se va face citirea semnalului analogic de la senzorul de îndoire s-a folosit un obiect pe care s-a atașat scriptul "Analog Input". Valoarea acestui semnal fiind mapată în intervalul [1:0]. Aici se alocă un id unic senzorului de îndoire.

Prin intermediul scriptului "Input Filter" se poate specifica intervalul rezistenței pe care se acționează atunci când este îndoit senzorul. În exemplul din **Error! Reference source not found.**, la pinul analogic va fi citită valoarea doar din intervalul [0.6:0.27]. Acest Interval fiind și el mapat la rândul lui între [1:0], pentru a fi utilizat de scriptul pentru rotația articulațiilor degetelor, acest script fiind discutat mai jos.

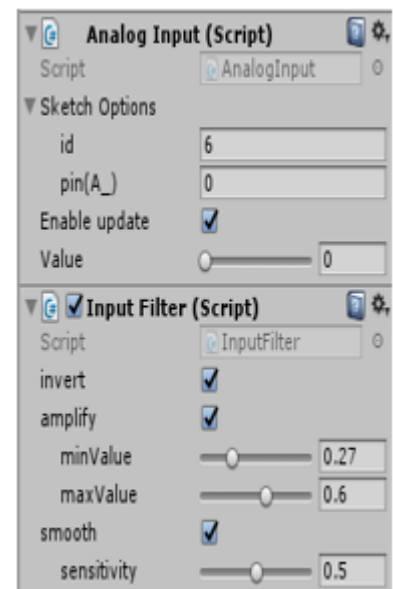


Figura 45 - Input

Acestea au cel mai important rol deoarece aici se setează modul de comunicare și modul de lucru pentru Unity și Arduino.

Prin intermediul scriptului "HM10" se setează:

- portul pe care se va face comunicarea între Unity și plăcuța Arduino

Prin intermediul scriptului "Ardunity App" se setează:

- rata de transfer a informației (bps) ce va fi folosită de Unity pentru a comunica prin Bluetooth cu modulul conectat la Android, ea fiind setată la 9600 bps.
- rezoluția pentru semnalul PWM, informație ce va fi utilizată de codul Arduino
- rezoluția pentru semnalul ADC, informație ce va fi utilizată de codul Arduino
- timpul pe care Arduino trebuie să îl aștepte, apoi se vor citi informațiile de la senzori

La fiecare GameObject cu rolul de articulație au fost alocate scripturile "Mapping Input", cu rolul de a modifica valoarea semnalului mapată și scriptul "Rotation Axis Reactor" ce stabilește axa pe care se va produce rotația și în ce mod.

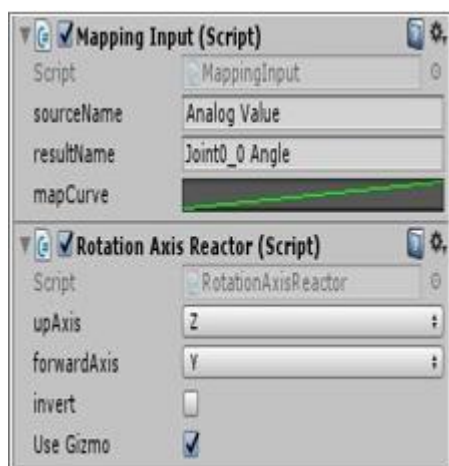


Figura 46 - maparea și rotația

De exemplu: valoarea obținută de la senzorul de îndoire a fost mapată în intervalul [1:0], și se dorește o rotație maximă de 30° a unei articulații. Pentru asta se va modifica curba mapării pe intervalul [30:0], curbă care poate lua mai multe forme. Când senzorul de îndoire este în poziția 0(degetul este întins), GameObject-ul corespunzător acelei articulații va avea 0° la rotație pe axa respectivă, iar dacă senzorul de îndoire ajunge în poziția 1(degetul este strâns), GameObject-ul se va roti 30° pe axa predefinită.

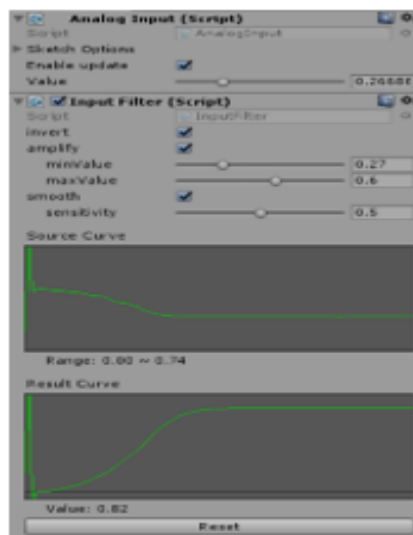


Figura 47 – Curba semnalului

În imaginea alăturată este prezentată forma curbei semnalului recepționat de la Arduino și noua curbă care va fi rezultată.

În "Input Value" s-a ales intervalul [0.6-0.27] pentru că mișcarea degetului produce flexarea senzorului de îndoire doar pe acest interval. Astfel aceste două puncte vor fi minimul și maximum semnalului sursă.

Pentru că este activat modul "invert", când valoarea semnalului recepționat scade de la maxim la minim, curba rezultată va crește de la minim la maxim.

În momentul în care nu se acționează asupra senzorului de îndoire, linia semnalului recepționat trebuie să fie cât mai dreaptă, dar acest lucru nu se întâmplă de fiecare dată din cauza materialelor de slabă calitate utilizate în producerea senzorilor.

În **Error! Reference source not found.** este prezentat punctul de maxim și punctul de minim de pe care se vor citi valorile atunci când se acționează asupra senzorului de îndoire.

Deget (pin Arduino)	Valoarea minimă	Valoarea maximă	invert activat
mare (A1)	0.27	0.60	da
arătător (A2)	0.60	0.76	da
mijlociu (A3)	0.14	0.26	da
inelar (A4)	0.15	0.24	da
mic (A5)	0.18	0.24	da

Tabelul 3 - punctul de maxim și minim

Interpretarea și utilizarea valorilor de la MPU-6050 la Arduino se realizează în scriptul "MPU Series". O problemă întâlnită, a fost păstrarea rotației pe axa Y atunci când utilizatorul rotește mâna pe axa Z. Rezolvarea pentru această problemă este prezentată în Codul 6.

```
Quaternion q = new Quaternion(_qX, _qY, _qZ, _qW);
// Convert Axis XYZ -> XZY and Rotation direction right -> left
Vector3 angles = q.eulerAngles;
float tmp_y = angles.y;
angles.y = angles.z;
angles.z = tmp_y;
q.eulerAngles = angles;
q = Quaternion.Inverse(q);
```

Codul 6 - Rotirea axelor YZ

Rotația mâinii este realizată prin adăugarea scriptului "Rotation Reactor" unui GameObject care reprezintă prima articulație a modelului 3d, în cazul acesta rotația se realizează la nivelul cotului.

În continuare este prezentat codul care realizează rotația modelului 3d. Valorile utilizate pentru rotație sunt obținute de la scriptul "MPU Series".

Funcția Update() este apelată la fiecare frame de Unity.

```
// Update este apelat la fiecare frame
void Update () {
    if(_time < 1f && smoothFollow == true)
    {
        _time += Time.deltaTime;
        _curRotation = Quaternion.Lerp(_fromRotation, _toRotation,
        _time);
    }
    else
    {
        _curRotation = _toRotation;
    }
    transform.localRotation = _initRot * _curRotation;
}
// S-au detectat valori noi la IMU
private void OnRotationChanged(Quaternion q) {
```

```

_fromRotation = _toRotation;
_toRotation = q;
_time = 0f;

```

Codul 7 - Rotația modelului 3d



Figura 48 - Spațiul Virtual

În imaginea alăturată este prezentată interfața grafică a aplicației Unity ce rulează pe un dispozitiv cu Android.

În partea de sus-dreapta a spațiului virtual sunt afișate două butoane:

-Butonul “**Quit**” este folosit de utilizator pentru a închide aplicația care rulează pe divace.

-Butonul “**Connect**” alternează cu butonul “**Disconnect**”. Atunci când utilizatorul dorește să utilizeze aplicația, el trebuie să se conecteze prima dată la Arduino. Atunci când se va apăsa butonul “Connect” va afișa o listă cu dispozitivele care au activat modulul Bluetooth. Dacă este ales modul corespunzător mănuii electronice, aplicația se va conecta la aceasta și astfel va începe comunicarea cu aceasta. Butonul “Disconnect” va apărea atunci când aplicația comunică cu mănua electronică și poate fi apăsat de utilizator dacă se dorește întreruperea comunicării.

În partea de sus-stanga a spațiului virtual este afișată o **image** corespunzătoare semnului realizat de utilizator. În aceasta imagine, prin ținerea mâinii în poziția de vertical și cu degetul mare și arătător întins, iar celelalte degete strânse, aplicația va afișa imaginea unui medic. Când este îndeplinită condiția afișării acestei imagini, aplicația va emite prin intermediul dispozitivului și sunetul “Am nevoie de medic”.

Mâna virtuală din imagine va copia mișcările mâinii utilizatorului care poartă mănua electronică.

Codul 8 prezintă modul în care este activată condiția în care utilizatorul solicită ajutorul unui medic.

```

if(mare == false && aratator == false && mijlociu == true &&
    inelar == true && mic == true && pozitie_mana == 3){
    if(timer > timp_difuzare){
        counter++;
        if(counter == 1){
            Debug.Log ("Semn: nevoie medic");
            img_nevoieMedic.SetActive(true);

```

```

        playerAudio.clip = _nevoieMedic;
        playerAudio.Play ();
    }
} //end-> am nevoie de medic (palma in sus)

```

Codul 8 - condiție pentru solicitarea medicului

Atunci când un pivot al degetului este rotit si trece de un anumit prag se va seta un fleag. Acest fleag reține astfel poziția degetului (întins sau îndoit).

```

if((R_Deget0 <= 340) && (R_Deget0 >= 200)){
    mare = true; //deget indoit
}
else{
    mare = false; //deget intins
}

```

Codul 9 - verificarea poziției degetului

Dacă mâna este rotită pe o anumită axa peste un prag anume, o variabilă își va schimba valoarea. Această variabilă este utilizată mai tarziu în funcția loop() pentru a se activa o imagine și un sunet corespunzător, tinandu-se cont și de poziția degetelor(codul 10).

```

if( ((R_Man_X <= 40) && (R_Man_X > -180)) &&
    ((R_Man_Z <= 90) && (R_Man_Z >= -90)) ){
    //mana intinsa orizontal dar cu palma in jos
    Pozitie_Man = 1;
}

```

Codul 10 - conditie pentru setarea pozitiei mainii

7.3 Platforma de dezvoltare

Această lucrare a fost dezvoltată pe un laptop Samsung, model NP300E52. Specificațiile sistemului sunt prezentate în **Error! Reference source not found.** de mai jos:

Procesor	Intel® Pentium® CPU B950 @ 2.10GHz
Memorie RAM	8 GB
Placa grafică	AMD Radeon R5 M330
Hard disk	500GB (5400RPM)
Sistem de operare	Windows 10 64 biți
Specificațiile platformei de dezvoltare	

Tabelul 4 - Specificațiile platformei de dezvoltare

8 Testare

Modul de testare a fost compus din 18 poziții, efectuate cu obiectivul de a analiza fiecare caz. Pozițiile 1-6 ale degetelor sunt prezentate în (**Error! Reference source not found.**). Fiecare poziție este repetată pentru fiecare grad de rotație al mâinii prezentat în (**Error! Reference source not found.**). Toate pozițiile a fost ținute timp de 10 secunde și repetate de 3 ori pentru a obține valorile medii ale DS, DMA și amplitudinii, pentru fiecare caz.

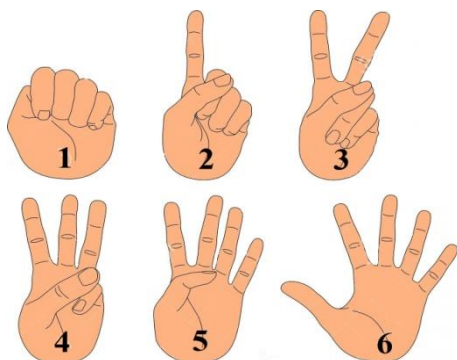


Figura 49 - Articulația degetului

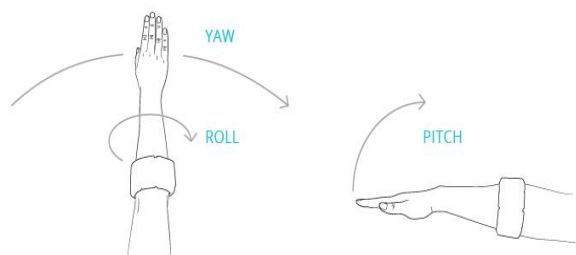









Figura 50 - Gradele de rotație

În imaginile următoare este prezentată comparația dintre mișcările utilizatorului și cele ale modelului virtual, dar și modul în care interacționează modelul 3d al mâinii cu celelalte obiecte.

Se poate observa un rezultat satisfăcător al modului de comunicare între dispozitive și interacțiunea obiectelor.



	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Eu sunt Laurentiu”</p>	
---	---	---

	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Am nevoie de medic”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Mă simt bine”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Am nevoie de ajutor”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Vino cu mine”</p>	

	<p>Mâna este în poziția orizontală cu palma orientată în sus, iar aplicația va emite suntele:</p> <p>“Cât costă?”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite suntele:</p> <p>“Vreau mâncare”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite suntele:</p> <p>“Bună ziua !”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite suntele:</p> <p>“Cheamă”</p>	

	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite suntele:</p> <p>“Poliția”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite suntele:</p> <p>“Pompierii”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite sunetul:</p> <p>“Nu !”</p>	
	<p>Mâna este în poziția orizontală cu palma orientată în jos, iar aplicația va emite sunetul:</p> <p>“Da !”</p>	

	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite sunetul:</p> <p>“ 1 ”</p>	
	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite sunetul:</p> <p>“ 2 ”</p>	
	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite sunetul:</p> <p>“ 3 ”</p>	
	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite sunetul:</p> <p>“ 4 ”</p>	

	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite sunetul:</p> <p>“ 5 ”</p>	
	<p>Mâna este în poziția verticală cu palma orientată în față, iar aplicația va emite suntele:</p> <p>“ Vă mulțumesc! ”</p>	

Tabelul 5 - Poziția reală vs poziția virtuală

9 Concluzii și modificări pentru viitor

A fost atins obiectivul general de dezvoltare a o aplicație ce rulează pe Android, aplicație ce conține un model virtual al unei mâini care să execute aceleași mișcări pe care le face mână electronică, dar și emiterea sunetului corespunzător dar și a imaginii corespunzătoare pe telefonul mobil.

Am reușit să capturez și să interpretez semnalul de la MPU-6050 folosit pentru a capta rotațiile mâinii, dar și mișcarea degetelor prin intermediul senzorilor de îndoire. Cerințele software au fost îndeplinite, deoarece mișcarea de orientare a mâinii a fost satisfăcătoare, cu rezultate precise și mișcări fluide. Modul în care mână electronică se mișcă în timp real cu mâna virtuală este unul natural, oferind plăcere și satisfacție utilizatorului pe parcursul utilizării acestei interfețe.

9.1 Dezvoltarea viitoare

În ceea ce privește dezvoltarea acestui proiect m-am gândit la un set de îmbunătățiri pentru viitor, atât pe partea de software cât și pe partea de hardware.

Aceste îmbunătățiri oferă utilizatorului o mișcare cât mai naturală a mâinii prin realizarea unui circuit de o dimensiune mai mică pentru a fi ușor de folosit și pentru a permite o libertate în mișcare pe de-o parte, iar pe de altă parte un circuit cât mai robust care să-i ofere utilizatorului posibilitatea de a se folosi de această mână electronică o perioadă îndelungată de timp.

Pe partea de software se dorește a avea accesibilitate sporită în vederea modificării și adăugării unor noi funcții ale aplicației în momentul utilizării. Acestea constă în lărgirea câmpului de semne printr-o introducere simplă prin apăsarea unui buton aflat în aplicația ce rulează pe Android sau prin introducerea unor noi semne folosind aceeași metodă.

Potrivit unor cercetări recente în România numărul persoanelor surdo-mute este foarte mare. Aceste fiind persoane diagnosticate cu un handicap în ceea ce privește capacitatea acestora de a vorbi sau de a auzi zgomotele din mediul înconjurător, deoarece aceste persoane sunt de sex feminin cât și masculin, voi aborda problematica introducerii unor funcții care îți oferă posibilitatea alegerii vocii de emisie în funcție de sex-ul persoanelor care utilizează aplicația.

10 BIBLIOGRAFIE

- [1] - KIM, J.-H.; THANG, N. D.; KIM, T.-S. 3-d hand motion tracking and gesture recognition using a data glove. In: IEEE. 2009 IEEE International Symposium on Industrial Electronics. [S.l.], 2009. p. 1013–1018.
- [3] - FAHN, C.-S. and SUN. Development of a data glove with reducing sensors based on. s.l. : IEEE Transactions on Industrial Electronics, IEEE, v. 52, n. 2, p. 585–594, 2005.
- [5] - SUTHERLAND, J. W. An Introduction to Sensors and Transducers. 2004, traducerea mea. Disponibil aici: <<http://www.mfg.mtu.edu/cyberman/machtool/machtool/sensors/intro.html>>. Accesat în anul 2017.
- [9] - FRADEN, J. Handbook of modern sensors: physics, designs, and applications. [S.l.]: Springer Science & Business Media, 2004.

REFERINȚE WEB

- [4] - FIFTH DIMENSIONAL TECHNOLOGIES. 5DT Data Glove 5 Ultra. 2011. Disponibil aici: <<http://5dt.com/products/pdataglove5u.html>>. Accesat în anul 2017.
- [6] – Varlad Cosmin. material didactic.<https://profs.info.uaic.ro/~vcosmin/pagini/resurse_arduino/Cursuri_2016/1/Arduino_1.pdf>. Accesat în anul 2018
- [7]- INVENSENSE. MPU-9250 Datasheet. <<https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>>.
- [8]- ELECTRONIC PRODUCTS. MEMS: A Brief Overview. 2013. Disponibil aici:<https://www.electronicproducts.com/Sensors_and_Transducers/Sensors_and_Transducers/MEMS_A_Brief_Overview.aspx?id=2546>. Accesat în anul 2017.
- [10] - Gyroscope, Sensor Wiki. Disponibil aici:<<http://sensorwiki.org/doku.php/sensors/gyroscope>>.
- [11] - BRAGA, N. C. Cum funcționează senzorii. Efeito Hall. 2014. Disponibil aici: <<http://www.newtonbraga.com.br/index.php/como-funciona/6640-como-funcionam-os-sensores-de-efeito-hall-art1050>>. Accesat în anul 2017.
- [12] - Cum funcționează senzorii. Efeito Hall. 2014. <http://www.newtonbraga.com.br/index.php/como-funciona/6640-como-funcionam-os-sensores-de-efeito-hall-art1050.com>
- [13] – Imagine preluată de aici: <<http://www.creeaza.com/tehnologie/electronica-electricitate/Potentiometrul851.php>>.

- [14] - Imagine preluată de aici: <https://www.magaudio.ro/index.php?main_page=index&cPath=5_49_67>.
- [15] – Documentația Oficială Ardunity. <<https://sites.google.com/site/ardunitydoc/more-in-depth/add-ardunity-controller.com>>.
- [16] – Schemele au fost realizate aici: <<https://www.draw.io/>>.
- [17] - Imagine preluată de aici: <<https://developer.thalmic.com/forums/topic/4094/?page=1#post-12936.com>>.
- [18] – Plug-in preluat de aici: <<https://assetstore.unity.com/packages/tools/input-management/ardunity-deluxe-62912>>.
- [19] - ROWBERG. I2Cdevlib. Accesat în anul 2016. <<https://github.com/jrowberg/i2cdevlib/>>.
- [20] - KHARLASHKIN, V.MPU-9250 DMP. Corecție magnetometru Arduino. Accesat în anul 2016.< <https://github.com/kharlashkin/MPU-9250DMP> >.
- [21] – Model 3d preluat de aici: <<https://github.com/semihsng/GloVR>>.
- [22] [23] – Documentația Oficială Unity. <<https://docs.unity3d.com/Manual/classSkinnedMeshRenderer.html>>.
- [24] - Imagine preluată de aici: <<https://www.dreamstime.com/royalty-free-stock-image-hand-motions-image12717306>>.
- [26] - Imagine preluată de aici: <http://cursuri.flexform.ro/courses/L2/document/Cluj-Napoca/grupa4/Rus_Maria/site/tp.html>.
- [27] - CAMPOS. O efeito de Coriolis. 2007. Disponibil aici:<<http://web.archive.org/web/20161207221235/tocampos.planetaclix.pt/astron/coriolis/coriolis.htm>>. Accesat în anul 2017.
- [28] – Informație tradusă de aici: <<https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro>>.
- [29] – Informație preluată de aici: <<https://www.vrstudio.ro/ce-este-realitatea-virtuala>>. Accesat în anul 2018.
- [30] - Informație preluată de aici: <https://ro.wikipedia.org/wiki/Mediu_de_dezvoltare>. Accesat în anul 2018.
- [31] – Stetic Gui Designer: <<http://www.monodevelop.com/documentation/stetic-gui-designer/>>.
- [32] – Informație și imagini preluate de aici: <https://profs.info.uaic.ro/~arduino/index.php/Comunicare_I2C/>. Accesat în anul 2018.

Codul sursă

https://github.com/Pistritu/Limbajul_Semnelor

CD / DVD

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.

