Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem de asistență inteligentă pentru persoanele cu dizabilități de auz și vorbire

Lucrare de disertație

prezentată ca cerință parțială pentru obținerea titlului de
Master în domeniul *Inginerie electronică și telecomunicații*
programul de studii de masterat *Comunicații Mobile*

Conducător științific:
Prof. Dr. Ing. Corneliu BURILEANU

Absolvent:
Ing. Maria-Mădălina ANDRONACHE

2020

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Programul de masterat **Comunicaţii mobile**

Anexa 2

TEMA LUCRĂRII DE DISERTAȚIE

a masterandului **ANDRONACHE Gh. Maria-Mădălina , 421-CMOB.**

1. Titlul temei: Sistem de asistență inteligentă pentru persoanele cu dizabilități de auz și vorbire

2. Descrierea contribuţiei originale a masterandului (în afara părţii de documentare) și specificații de proiectare:
Se va realiza un sistem ce va folosi atât o parte hardware, cât și o parte software, în scopul utilizării acestuia de către persoane cu dizabilități de auz și vorbire.
Practic, se va crea un mod de comunicare exprimat cu ajutorul gesturilor:
1. partea hardware - va fi bazată pe achiziția de date prin intermediul unor senzori și translatarea semnalelor furnizate de aceștia în semnal vocal sau sub formă de text.
2. partea software - va fi bazată pe o aplicație Android ce va conține gesturi predefinite pe care utilizatorul le poate folosi.
Sistemul va cuprinde componente electronice cunoscute precum senzori, microcontrolere, etc., dar și componente software precum Android Studio, Java SE Development Kit, etc.
Acest sistem va fi portabil, pentru a ușura aplicabilitatea acestuia în viața de zi cu zi.

3. Resurse folosite la dezvoltarea proiectului:
Senzori, Pachet CAD, Arduino, Java, Android Studio

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Programare în Java și Android, Planificare radio și sisteme IoT, Sisteme în cip în comunicații

5. Proprietatea intelectuală asupra proiectului aparţine: studentului

6. Data înregistrării temei: 2019-11-27 16:24:53

Conducător(i) lucrare,
Prof. dr. ing. Corneliu BURILEANU

semnătura: .....

Student,

semnătura:.....

Responsabil program master,
Prof. dr. ing. Octavian FRATU

semnătura:.....

Decan,
Prof. dr. ing. Mihnea UDREA

semnătura:.....

Cod Validare: **c0807fb7ec**

Declarație de onestitate academică

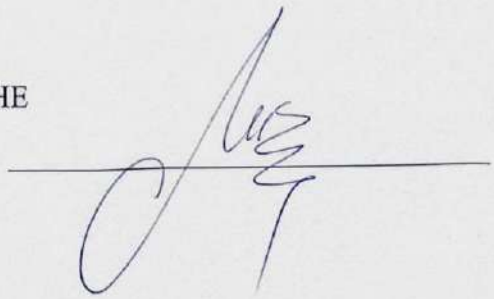
Prin prezenta declar că lucrarea cu titlul " Sistem de asistență inteligentă pentru persoanele cu dizabilități de auz și vorbire", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de Master în domeniul Telecomunicațiilor, programul de studii Comunicații Mobile este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, iunie 2020

Absolvent Maria-Mădălina ANDRONACHE



CUPRINS

LISTA FIGURILOR

LISTA TABELELOR

LISTA ACRONIMELOR

A

ADC – Analog to Digital Converter – Convertor analog-digital

ALU – Arithmetic Logic Unit – Unitate aritmetico-logică

AP – Access Point **–** Punctele de acces wireless

ASCII – American Standard Code for Information Interchange **–** Sistem de codificare a caracterelor bazat pe alfabetul englez.

ASL – American Sign Language – Limbajul semnelor american

AVD – Android Virtual Device – Dispozitive virtuale Android

B

BSL – British Sign Language – Limbajul semnelor britanic

C

CE – Chip Enable – Pin de activare a chip-ului

CPU – Central Processing Unit – Unitate Centrală de Prelucrare

CSN – Chip Select Not – Pin pentru dezactivarea chip-ului

E

EPROM – Erasable Programmable Read-Only Memory **–** memorie nevolatilă care își păstrează datele chiar și când se întrerupe alimentarea cu curent electric

F

FTJ – Filtru trece jos

G

GND – Ground Pin – Pin de masă

GPIO – General-purpose input/output – Intrare / ieșire cu scop general

GPR – General Purpose Register – Registru de uz general

I

INT – Interrupt Signal Pin – Pinul de întrerupere

IoT – Internet of Things – Internetul obiectelor

IP – Internet Protocol – Protocol de transmisie a datelor prin intermediul Internetului

IR – Instruction Register – Registrul de instrucțiuni

ISL – Irish Sign Language – Limbajul semnelor irlandez

ISM – Industrial, Scientific and Medical Band Specifications – Bandă de frecvență pentru diverse tipuri de comunicații

I2C – Inter-Integrated Circuit – Magistrală pentru transmisie de date serială master-slave

L

LCD – Liquid Crystal Display – Afișajul cu cristale lichide

LED – Light Emitting Diode – Diodă luminescentă

M

MEMS – Micro-electro-mechanical systems – Sistemele de tip micro-electro-mecanic

MISO – Master In Slave Out – Pin de transmisie a datelor către microcontroler

ML – Machine Learning – Învățare automată

MOSI – Master Out Slave In – Pin de recepție a datelor de la microcontroler

MPU – Microprocessing Unit

P

PC – Program Counter – Contorul de programe

PCB – Printed Circuit Board – Circuit imprimat / Cablaj imprimat

PWM – Pulse Width Modulation – Modulația lățimii pulsului / Modularea duratei pulsului

R

RAM – Random Access Memory – Memoria cu acces aleator

RF – Radio frequency – Frecvență radio

ROM – Read Only Memory – Memorie nevolatilă

RS – Register Select – Registru de selecție

S

SCL/ SCK – Serial Clock Line – Pin pentru ceas serial

SDA – Serial Data Line – Pin pentru date seriale

SDK – Software Development Kit – Set de dezvoltare a programelor

SFR – Special Function Register – Registru cu funcții speciale

SP – Stack Pointer – Registru de stivă

SPI – Serial Peripheral Interface – Interfață sincronă de mare viteză

T

TCP – Transmission control protocol – Protocolul de Control al Transmisiei

TSL – Taiwan Sign Language – Limbajul semnelor din Taiwan

U

UART – Universal Asynchronous Receiver-Transmitter – Transmițător-Receptor asincron universal

UC – Control Unit – Unitate de control

UDP – User Datagram Protocol – Protocolul Datagramelor Utilizator / Protocol de comunicație fără conexiune

V

VCC – Voltage Common Collector – Pini de alimentare

VoIP – Voice over Internet Protocol **–** Voce peste Protocolul de Internet

W

Wi-Fi – Wireless Fidelity – Standard de comunicație fără fir

X

XCL – Auxiliary Serial Clock – Pin pentru ceas auxiliar

XDA – Auxiliary Serial Data pin – Pin pentru date seriale auxiliare

INTRODUCERE

MOTIVAȚIA ALEGERII TEMEI

Unul dintre dezavantajele majore ale societății noastre este bariera care se creează între persoanele cu handicap și persoanele normale. Comunicarea este singurul mijloc prin care putem să ne împărtășim gândurile sau să transmitem mesaje, dar o persoană cu dizabilități se confruntă involuntar cu dificultăți în realizarea acesteia. Impactul acestor dificultăți crește în momentul în care acestea se întâlnesc în anumite domenii, cum ar fi serviciile bancare, spitalul sau instituții ce dețin servicii de apărare ale drepturilor cetățenilor.

Limbajul semnelor este un limbaj bine structurat, cu morfologie și sintaxă, folosind diferite moduri de exprimare. De aceea, a devenit modul de bază al comunicării pentru persoanele cu dizabilități implicând semne manuale (degetele, mâinile, brațele) și non-manuale (fața, capul, ochii și corpul). Există limbaje ale semnelor diferite în funcție de regiunile lumii: limbajul semnelor american (ASL), limbajul semnelor britanic (BSL), limbajul semnelor din Taiwan (TSL), limbajul irlandez al semnelor (ISL) și multe altele.

Limbajul semnelor este cea mai importantă metodă prin care persoanele cu deficiențe pot interacționa cu restul lumii. Însă, conversația devine complicată dacă ascultătorul ignoră limbajul semnelor, apărând dificultăți de înțelegere. O persoană care poate vorbi și aude în mod corespunzător (persoană normală) nu poate comunica cu persoana cu dizabilități dacă nu este familiarizată cu limbajul semnelor. Același caz este valabil atunci când o persoană cu dizabilități de auz și vorbire dorește să comunice cu o persoană normală sau cu o persoană cu deficiențe de vedere.

Sistemul de recunoaștere a limbajului semnelor transferă comunicarea de la interacțiunea inter persoane la cea dintre persoane și calculator. Acest proiect își propune să ofere o comunicare

bidirecțională între oamenii cu dizabilități și oamenii normali și se bazează pe nevoia de a dezvolta un dispozitiv electronic care să traducă limbajul semnelor în vorbire.

Avantajele principale ale sistemului vor fi:

1. Poate ajuta persoanele cu dizabilități să interacționeze cu oamenii obișnuiți
2. Poate oferi un instrument pentru ca părinții să-și învețe copiii o altă formă de comunicare
3. Introducerea de jocuri pentru instruirea copiilor cu dizabilități

OBIECTIVELE LUCRĂRII

În cadrul acestei lucrări se vor realiza două abordări – o abordare bazată pe senzori și o abordare bazată pe vizualizare:

1. Abordarea bazată pe senzori: se folosesc diferite tipuri de senzori ce se plasează pe mâna subiectului, iar, când mâna efectuează orice gest, datele sunt înregistrate și analizate.
   În cadrul acestui scenariu, se folosesc senzori de flexiune ce sunt atașați unei mănuși. Persoanele cu deficiențe pot folosi aceste mănuși prin efectuarea unor gesturi de îndoire a senzorilor de flexie ce generează un semnal audio.
   Procedura bazată pe senzori dăunează mișcării naturale a mâinii din cauza utilizării hardware-ului extern, iar dezavantajul principal este că gesturile complexe nu pot fi realizate.

2. Abordarea bazată pe vizualizarea gestului: se folosește o cameră ce captează imaginea gestului, extrage caracteristica principală și o recunoaște. În această lucrare, imaginea mâinii este capturată folosind o cameră fotografică simplă.

STRUCTURA LUCRĂRII

Lucrarea constă din două părți principale:

Prima parte cuprinde informații detaliate cu privire la teoria și revizuirea literaturii referitoare la proiect atât pentru partea hardware, cât și pentru partea software, împărțind aceste aspecte în două mari capitole. Acestea cuprind, mai departe, mai multe subcapitole despre componente hardware sau diverse părți ale dezvoltării software.

În partea a doua, se explică metodele folosite în realizarea proiectului. Procesul include proiectarea circuitului și dezvoltarea de software-ului prin diverși algoritmi. Se menționează că există două tipuri de software: cel realizat pentru partea hardware și cel independent.

În capitolul următor, se reliefează rezultatele proiectului și se analizează eficiența proiectului comparativ cu datele existente în literatură.

Ultimul capitol acoperă concluziile, problemele, recomandările și îmbunătățirea suplimentară a proiectului.

CAPITOLUL I

Aproximativ nouă miliarde de oameni din lume sunt surzi și mut. Cât de des întâlnim acești oameni care comunică cu lumea normală?

Limbajul semnelor este singurul instrument de comunicare folosit de persoanele cu deficiențe de auz și vorbire pentru a comunica între ele. Cu toate acestea, oamenii normali nu înțeleg limbajul semnelor, iar acest lucru va crea o barieră mare de comunicare. În plus, limbajul semnelor nu este ușor de învățat datorită diferențelor sale naturale în structura propoziției și în gramatică. Prin urmare, este necesar să se dezvolte un sistem care să poată ajuta la traducerea limbajului semnelor în text și/sau semnal audio, pentru a se asigura o comunicare eficientă.

În România, numărul total de persoane cu dizabilități a fost de 759.019 persoane (în 2015), și 737.885 (în 2014), reprezentând 3,41% (în 2015) și, respectiv, 3,47% (în 2014) din populația României [1, 2].

Marea majoritate (741.516 persoane, respectiv 97,7%) se află în grija familiei lor și/sau trăiesc independent și doar 2,3% (17.457 de persoane) sunt instituționalizate în instituțiile publice rezidențiale de asistență socială pentru adulți cu dizabilități [1, 2].

Femeile reprezintă 53,12% dintre persoanele cu dizabilități. Numărul persoanelor cu vârsta peste 50 de ani reprezintă 68,40% din numărul total al persoanelor adulte cu dizabilități.

Centralizarea datelor pe grupe de vârstă arată că persoanele în vârstă (peste 65 de ani) reprezintă 37,2% (282.158 persoane) în numărul total de adulți cu dizabilități; aproximativ 54,8% (415 852 persoane) sunt între 18-64 de ani, o gamă de vârste care este considerată legal capabilă să lucreze.

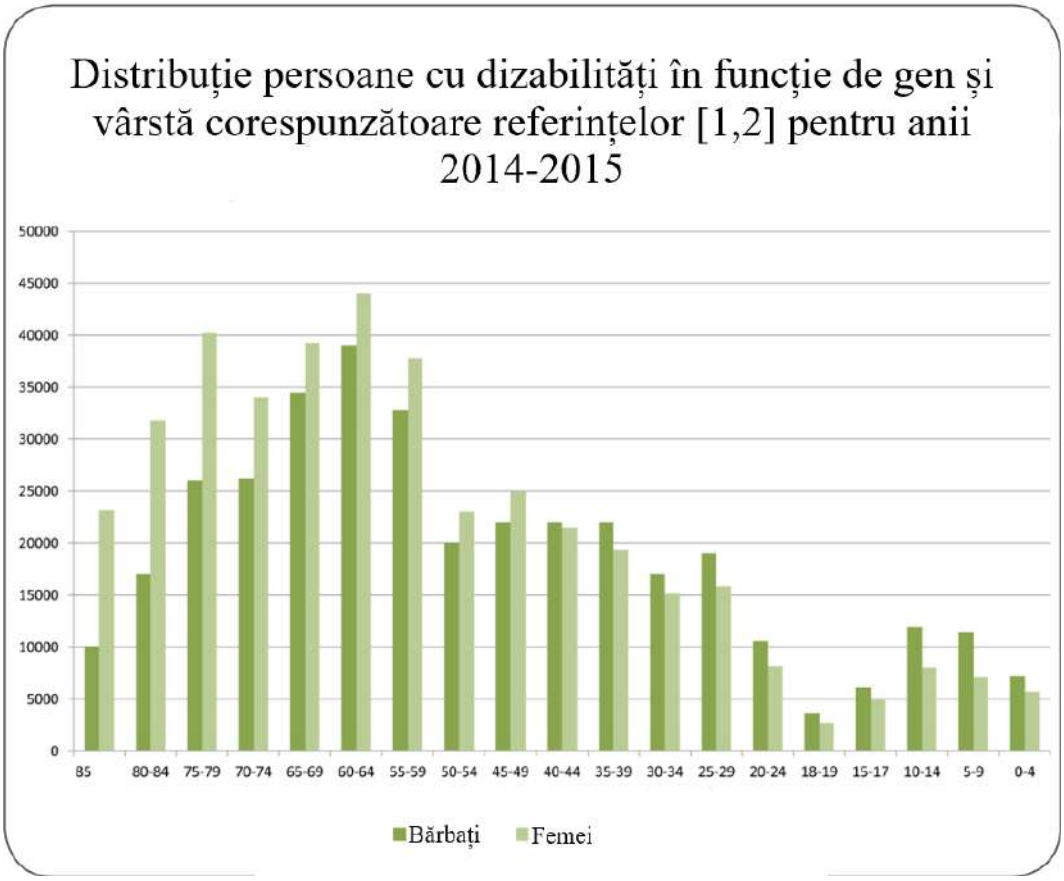


Figura I. 1: Diagramă distribuție persoane cu dizabilități în funcție de gen și vârstă [3]

Scenarii: Recunoașterea gesturilor este clasificată, în principal, în abordare bazată pe recunoaștere vizuală și abordare pe bază de senzori:

1. Abordarea pe bază de senzori – folosește o mănușă electronică unic asamblată, ce are senzori care ne oferă forma mâinii. Este o metodă des utilizată deoarece modul de achiziție a datelor prin îndoirea degetelor și accelerometru este unul destul de simplu, iar rezultatul acoperă o gamă variată de gesturi. Complexitatea intervine în modul de procesare a acestor date, transmiterea acestora în condiții de securitate și recunoașterea eficientă a gestului transmis de sistem.
   Se menționează că, crescând numărul de senzori, se va îmbunătăți acuratețea datelor, iar numărul de gesturi posibile va crește considerabil, indiferent dacă se folosește un scenariu bazat pe alfabet sau bazat pe gesturi predefinite.
2. Abordarea bazată pe recunoaștere vizuală - imaginile utilizatorului [cel care folosește limbajul semnelor] sunt capturate prin intermediul unei camere video. Acestea sunt procesate pentru a se realiza recunoașterea gesturilor. Este o metodă cu o complexitate superioară, dar este ușor adaptabilă deoarece, ulterior, în cadrul sistemului, se poate realiza și o recunoaștere de facială, ceea ce ar îmbunătăți considerabil acuratețea.
   Se menționează că această abordare depinde de caracteristicile de mediu[condițiile de fundal și luminozitate] și de poziția camerei video.
   Această metodă va fi implementată cu prin intermediul unei aplicații Android în limbajul Java sau Kotlin.

I.1 Abordarea bazată pe achiziție de date cu ajutorul unor senzori

A) Considerații generale

Sistemul de recunoaștere bazat pe date extrase cu ajutorul unor senzori a apărut pentru prima dată în literatură sub semnătura lui Sayre Glove în 1977. Deși această tehnologie este încă în stadiul său emergent, o serie de aplicații au fost implementate în timp real. Conceptul de bază implică utilizarea sistemelor sub formă de mănuși ce realizează achiziții de date purtate de persoanele cu dizabilități.

În cadrul acestui scenariu, această lucrare se concentrează pe dezvoltarea unui sistem cu senzori de flexie și accelerometru ce este folosită pentru a capta mișcarea utilizatorului. Senzorii de flexie schimbă valoarea rezistenței în funcție de unghiul de îndoire. Semnalele acestora, care indică activitățile mușchilor înrudiți în timpul executării unui gest, au avantaje în captarea de mișcări [de la încheietura mâinii sau degete - în cazul nostru]. Practic, este un dispozitiv care are câțiva senzori care se pot purta și sunt montați pe un suport format de o mănușă care măsoară diferiții parametri analogi asociați cu mișcarea degetelor și orientarea mâinii în timpul oricărui gest particular. Acești senzori achiziționează valori analogice, pe care le codifică, pentru a recunoaște anumite gesturi.

Când un gest este făcut de persoana cu dizabilități, microcontrolerul începe conversia analogică digitală a datelor de intrare pe care le primește de la senzorii de flexie și MPU6050. Datorită mișcărilor mâinii, există o cădere potențială pe rezistența variabilă, care este senzorul de flexie, din care se obține o valoare analogică a tensiunii. Pentru a transforma aceste date în formă digitală, se vor folosi un divizor de tensiune alături de un comparator sau ADC-ul din pinii de intrare ai microcontroler-ului. Dacă valoarea tensiunii trece peste prag, aceasta este recunoscută ca intrare. Accelerometrul implicat în partea de achiziție are scopul de clasificare a gesturilor de înclinare a mâinii și este montat în partea superioară a mâinii. Microcontroler-ul este utilizat pentru a achiziționa datele transmise din senzori, pentru a le prelucra și transmite prin intermediul unui modul RF. Practic, dacă gestul se potrivește cu valoarea stocată anterior în memoria principală a microcontrolerului, atunci se declanșează transmisia în modulul de comunicații. După procesarea în sistemul de bază, ieșirea corespunzătoare este produsă în format text pentru persoana surdă și pentru ieșirea vocală pentru persoana mută sau în ambele moduri pentru a uniformiza aplicabilitatea. Practic, microcontroler-ul verifică ieșirea senzorilor de flexie și apoi se calculează lățimile pulsurilor pe care le și salvează. Astfel, gesturile sunt transformate în mesaje text și în semnal audio. O serie de tehnici sunt folosite pentru a converti aceste gesturi în rezultatele necesare, iar aceste tehnici sunt fie bazate pe imagini, fie bazate pe dispozitive anexe, fie bazate pe ambele direcții. Alimentarea sistemului se realizează prin intermediul unui circuit de alimentare ce realizează o tranziție din 12V către 5V, iar din 5V către 3,3 V.

Acest dispozitiv poate fi considerat a fi unul portabil, deoarece datele sunt deja stocate în memoria microcontrolerului. Cu ajutorul senzorului de accelerometru și giroscop, sistemul este mai sensibil împotriva determinărilor incorecte deoarece evită achiziția de date când există doar o flexie puțin sesizabilă.

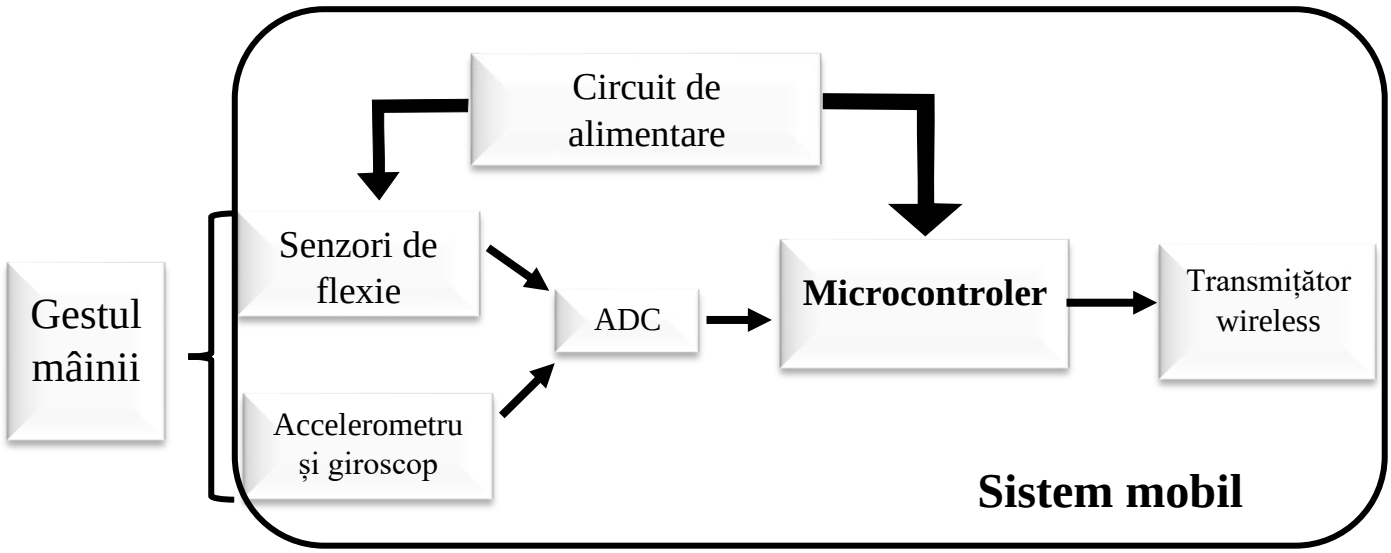


Figura I. 2: Sistem aplicabil mănușă

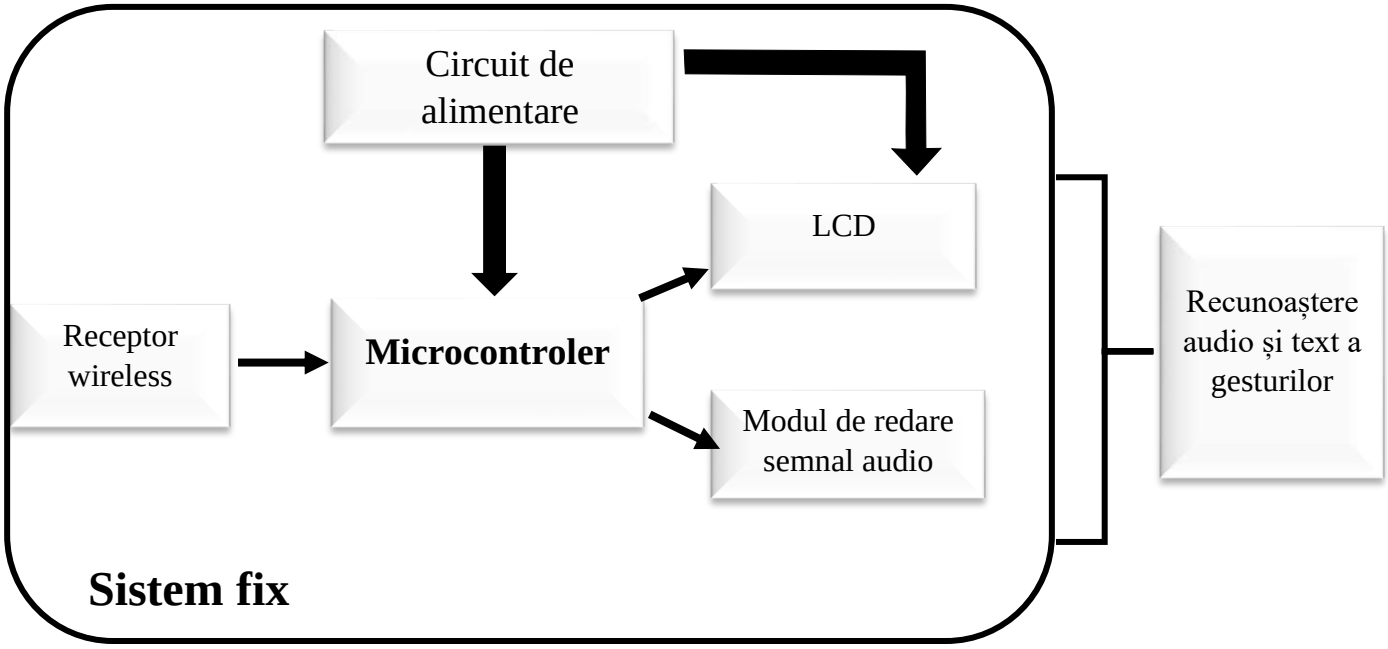


Figura I. 3:Sistem de recunoaștere a gesturilor

Sistemul utilizează un emițător și un receptor pentru a reduce greutatea componentelor ce vor fi utilizate pe mănușă și pentru a face ușoară mișcarea mâinii. Impactul imediat dat de această abordare prin utilizarea a două procesoare va fi cel de cost suplimentar. Acest sistem nu reprezintă o abordare complexă deoarece achiziția de date se realizează prin îndoirea degetului și orientarea 3D a mâinii cu ajutorul unor senzori ce se prezintă sub forma unei mănușii. Aceasta este foarte potrivită pentru a percepe atât mișcarea degetelor cât și înclinările semnului ce urmează a fi detectat.

Aspectele de discuție principale ar putea fi vulnerabilitatea sistemului în caz de zgomot și confortabilitatea purtării acestuia de către utilizator. Citirea senzorului de flexie nu este foarte stabilă și sensibilă la zgomot, iar precizia de testare trebuie să fie destul de mare deoarece cuvintele implică mișcare care trebuie detectată de accelerometru.

Figura I. 4: Mod grafic de implementare al proiectului pentru partea de transmisie

În unele cazuri, asemănarea dintre măsurători referitoare la gesturi diferite este foarte mare. Această asemănare înșelătoare determină o detectare greșită cunoscută sub numele de problemă de identificare.

Mănușa de date este formată din două tipuri diferite de senzori: senzori de flexie și senzor pentru dispozitiv de măsurare, care, în implementarea aceasta, este un modul integrat de accelerometru și giroscop, ambele măsurând orientări în toate cele trei axe posibile X, Y, Z – MPU6050. Deci, prin diferite combinații formate prin îndoirea senzorilor flexori și diferite orientări date de MPU6050, se creează combinații diferite ale valorilor de rezistență pentru ca pinul de ieșire al microcontroler-ului să reprezinte o entitate diferită.

Ieșirea senzorilor dispozitivului de măsurare este detectată în de modul de detecție slabă, în timp ce ieșirea senzorilor flexori și, prin urmare, gestul general este procesat în modulul de detecție a gesturilor.

Algoritmul de realizare al proiectului

Finalizarea acestui proiect va necesita îndeplinirea mai multor etape, după cum este descris mai jos:

- Dezvoltarea sistemului hardware – proiectarea circuitului și a PCB-ului
- Dezvoltarea modulului de detectare care să încorporeze și să diferențieze diferitele tipuri de gesturi.
- Achiziția de date
- Dezvoltarea unui program de recunoaștere a gesturilor prin intermediul unei baze de date
- Transmitere RF a semnalelor
- Interpretarea semnalelor primite și procesarea acestora
- Detecția gesturilor prin intermediul unei baze de date
- Afișarea detecției pe LCD
- Declanșare semnal audio pe baza detecției

B) Descriere componente hardware

1. Senzori de flexie – folosiți pentru recunoașterea gesturilor

Un senzor de flexie, numit și senzor de îndoire, măsoară deviația cauzată de îndoirea senzorului. Dezvoltat de la sfârșitul anilor 80, există trei tipuri de senzori de flexie. Inițial, au fost creați senzori de flexie optici, iar, ulterior, senzori conductivi pe bază de cerneală și senzori capacitivi. Deși sunt utilizați pentru același tip de detecție, fiecare tip de senzor de flexie este diferit atât din punct de vedere al construcției, cât și al principiului de funcționare.

Cele mai comune trei tipuri de senzori de flexie întâlnite în practică sunt [4]:

- fibra optică
- pe bază de cerneală conductivă
- țesătură conductivă sau pe bază de polimeri

Un senzor tipic de îndoire are următoarele specificații de bază [4]:

- raza de deviere - determină unghiul maxim de deviere care poate fi măsurat (spre deosebire de unghiul maxim pe care senzorul poate fi îndoit).
- senzor uni- vs. bi-direcțional - rezistența crește atunci când este îndoit în oricare din cele două direcții opuse, cu toate acestea nu există nicio diferență în ceea ce privește direcția.
- senzor uni- vs. bi-polar - măsoară deviația în două direcții opuse, oferind măsurători diferite.
- gama de rezistență (nominală până la deplasare completă) – gama este măsurată ca diferență de la rezistența nominală la rezistență cu deplasare completă.

Senzorul de flexie optic este format dintr-un tub flexibil cu două capete, un perete interior reflectorizant în interiorul tubului flexibil și o sursă de lumină plasată într-un capăt și un detector fotosensibil plasat în celălalt capăt al tubului flexibil pentru a detecta o combinație de direct raze de lumină atunci când tubul flexibil este îndoit [5].

Cel mai comun senzor de flexie de tip optic este fibra optică. Îndoirea fibrei optice determină o pierdere de lumină (intensitate). Pierderea de lumină este adesea îmbunătățită prin tăierea, lustruirea sau abraziunea unei părți din partea de plastic. Datorită principiului de detecție, senzorii de îndoire a fibrei optice unice sunt dispozitive unipolare. *[6]*

Senzorul pe bază de cerneală conductivă este un dispozitiv pasiv rezistiv fabricat prin punerea unei benzi de cerneală conductivă pe un substrat din rășină fenolică pe acesta este așezat un conductor segmentat pentru a forma un potențiometru flexibil în care rezistența se schimbă la deviere. [5] În repaus (când este așezat plat), senzorul de îndoire este caracterizat printr-o rezistență intrinsecă. Pe măsură ce senzorul este îndoit, materialele rezistive din interior vin în contact, crescând astfel rezistența. De obicei, rezistența nominală se situează între 10kΩ și 50kΩ [4].

Pentru a conduce electricitatea, această cerneală conține particule de carbon sau argint amestecate într-un mediu pigmentat. De obicei, particulele de carbon sunt suspendate în cerneală pentru a evita decolorarea pigmentului în timp. Acest tip de cerneală poate fi, de asemenea, aplicat în

siguranță pe hârtie pentru a evita absorbția în fibre, modificând astfel proprietățile hârtiei. Cu toate acestea, particulele de carbon pot fi sensibile la modificările umidității.

Majoritatea senzorilor de îndoire bazați pe cerneală ce se pot găsi pe piață sunt dispozitive unipolare, adică rezistența crește pe măsură ce flexia crește într-o direcție și este neschimbată dacă este îndoită în cealaltă direcție. Plasarea a două dispozitive va permite măsurători bipolare pentru captarea devierii în ambele direcții. Zgomotul ce intervine în valoarea rezistenței este mic, dacă nu sunt complet neglijabil [4].

Banda senzorilor poate avea lungimi de 1, 2 sau 3 inch în funcție de conectorii care pot fi interfațați. Pe măsură ce lungimea crește, rezistența intrinsecă crește, la fel și rezistența la îndoire. De asemenea, există diferite opțiuni de laminare și acoperire pentru a crește durabilitatea și rigiditatea [4].

Senzorii de flexie pe bază de țesut, filet sau polimer constau, de obicei, din două straturi de material conductor cu un strat de material rezistiv între ele. Pe măsură ce se aplică presiune (direct sau prin îndoire), cele două straturi de material conductor se strâng și rezistența senzorului scade. Acest mecanism de detecție este similar cu rezistențele sensibile la forță sau presiune.

Senzorii de flexie convertesc o **energie fizică** într-una de tip **electric**. Acest tip de senzori s-a dovedit a fi eficient în sistemele dedicate urmăririi mișcărilor umane, atât la membrele superioare și inferioare, cât și la nivelul capului și toracelui, dar au aplicații importante și în alte contexte, cum ar fi domeniul auto, robotică și proteză, instrumente muzicale și instrumente de măsurare. De menționat este ca trebuie luate în considerare și acuratețea, repetabilitatea și reproductibilitatea măsurării, erorile medii și întârzierea de timp pe care acești senzori le pot crea.

Un senzor de flexie rezistiv este un dispozitiv pasiv care nu necesită alimentare pentru a funcționa. Atunci când senzorul este îndoit, substratul este comprimat și stratul conductor se întinde, crescând astfel rezistența totală până la o valoare maximă corespunzătoare unghiului de îndoire maxim măsurabil.

Valoarea electrică poate avea o componentă reactivă (în special capacitivă), cu valoare neglijabilă, dar care poate fi să afecteze măsurătoarea la o deformare rapidă. pentru a reia răspunsul electric al senzorilor de flux în mișcări de deformare rapidă. Astfel, senzorul se poate caracteriza prin valoarea de rezistență R față de unghiul de îndoire. Ca o soluție intermediară, se va folosi un divizor de tensiune de tip senzor de flexie și rezistor cu valoare fixă, dar tensiunea de ieșire va fi considerată tot în funcție de unghiul de îndoire.

Cerneală conductivă	Fibră optică	Țesatură / polimer
robust și durabil	măsurători precise, repetabile	calități fizice atractive
lungime fixă	necesită sursă de lumină și detector	performanță slabă și variabilă

Tabelul I. 1: Comparație între cele mai frecvente tipuri de senzori de flexie

$$V_o = V_g * \frac{Senzor}{R1 + Senzor} \quad (1)$$

2. MPU-6050 – Modulul de giroscop și accelerometru cu 3 axe – folosit ca mod de recunoaștere a mișcării mâinii

Datele precizate în această parte sunt preluate din Lucrarea de Diplomă unde am abordat același tip de dispozitiv.

1. Senzorii sunt, adesea, definiți ca fiind dispozitive "care detectează sau măsoară unele condiții sau proprietăți și înregistrează, indică sau uneori răspund la informația primită" [7].

Astfel, conform [7], aceștia au "funcția de a converti un stimul într-un semnal măsurabil", deținând atât circuite care "transformă mărimea de intrare în semnal electric util, cât și circuite pentru adaptarea și conversia semnalelor."

2. Alegerea senzorilor trebuie făcută ținând cont de proprietate de monitorizat, de domeniu în care variază acesta, de dimensiunile ce trebuie respectate sau de geometria sistemului, de condiții speciale de mediu sau de lucru, de tipul mărimii de ieșire și nu în ultimul rând de cost [8].

Cuvântul "senzor" este derivat din cuvântul latin „sentire" care înseamnă "a percepe" [8].

3. Definiția din dicționar atribuie cuvântului "senzor" semnificație de "dispozitiv care detectează schimbarea într-un stimul fizic și o transformă într-un semnal ce poate fi măsurat sau înregistrat" [9].

Clasificarea generală a senzorilor se face din următoarele puncte de vedere:

1. **După proprietățile obiectelor pe care le pun în evidență deosebită**:
   a) Senzori pentru evaluarea formei și dimensiunilor geometrice (deplasarea); [7]
   b) Senzori pentru determinarea proprietăților fizice ale obiectelor;); [7]
   c) Senzori pentru determinarea proprietăților chimice ale obiectelor (concentrație, compoziție, analizoare chimice, etc)); [7]

2. **Din punct de vedere constructiv deosebim:**
   a) Senzori activi (generatori), la care se realizează conversia energiei mărimii de măsură în energie electrică;
   b) Senzori pasivi (parametrici), la care se utilizează o sursă auxiliară de energie, ai cărui parametri depinde de caracteristicile mărimii de măsurat.

3. **După tipul semnalului furnizat la intrare deosebim:**
   a) Senzori pentru mărimi fizice (deplasare, viteză, efort, cuplu, presiune, câmp magnetic, temperatură);
   b) Senzori pentru mărimi chimice (concentrație, analiza gazelor, pH);

c) Senzori biologici (tactili, vizuali, auditivi, zaharuri, proteine).

4. **După timpul semnalului furnizat la ieșire deosebim :**
   a) Senzori analogici, la care semnalul continuu de ieșire urmărește variațiile mărimii aplicate la intrare:
   b) Senzori numerici, la care semnalul discontinuu de ieșire sub formă de impulsuri, reprezintă modul de variație a mărimii de măsurat.

Modulul senzorului MPU-6050 este un dispozitiv complet de urmărire a mișcării în 6 axe și combină un giroscop cu 3 axe, accelerometru cu 3 axe, un senzor de temperatură și procesorul digital de mișcare, toate într-un singur chip. De asemenea, are o interfață I2C pentru a comunica cu microcontrolerul. El dispune și de o magistrală I2C auxiliară pentru a comunica cu alte dispozitive de senzor cum ar fi magnetometru cu 3 axe, senzor de presiune, senzor de carburant. Dacă magnetometrul cu 3 axe este conectat la magistrala I2C auxiliară, atunci MPU-6050 poate furniza o ieșire completă de mișcare în 9 axe.

Modulul MPU-6050 are 8 pini:

- **INT**: pin de ieșire pentru o întrerupere digitală.
- **AD0**: pinul de adresare a LSB pentru modul I2C Slave - Acesta este bitul 0 în adresa slave a dispozitivului pe 7 biți.
  Dacă este conectat la VCC atunci este citit ca 1 logic și schimbă adresa slave.
- **XCL**: pin pentru ceas serial auxiliar. Acest pin este utilizat pentru a conecta un alt senzor I2C cu senzor SCL la MPU-6050.
- **XDA**: pin pentru date seriale auxiliare. Acest pin este utilizat pentru a conecta alte senzori I2C cu senzori SDA la MPU-6050.
- **SCL**: pin pentru ceasul serial - Se va conecta acest pin la pinul SCL al microcontrolerului.
- **SDA**: pin pentru date seriale – Se va conecta acest pin la pinul SDA al microcontrolerului.
- **GND:** pini de masă
- **VCC**: pin de alimentare

După obținerea datelor brute de la senzor putem calcula accelerația și viteza unghiulară prin împărțirea datelor brute cu factorul de scală al sensibilității după cum urmează:

Valorile accelerometrului în g:

Accelerația de-a lungul axei X = (date accelerometru pe axa X / 16384) g.

Accelerația de-a lungul axei Y = (date accelerometru pe axa Y / 16384) g.

Accelerația de-a lungul axei Z = (accelerația axei Z / 16384) g.

Valorile giroscopului în ° / s (grad pe secundă):

Viteză unghiulară de-a lungul axei X = (date giroscop pe axa X / 131) ° / s.

Viteza unghiulară de-a lungul axei Y = (date giroscop pe axa Y/ 131) ° / s.

Viteză unghiulară de-a lungul axei Z = (date giroscop pe axa Z/ 131) ° / s.

Valoarea temperaturii în ° C (grade Celsius)

Temperatura în grade C = ((date senzor de temperatură) / 340 + 36,53) ° C.

a) Accelerometrul

Unul dintre cei mai obișnuiți senzori inerțiali este accelerometrul, un senzor dinamic capabil să dea o gamă destul de largă de valori. Accelerometrele sunt de mai multe feluri, dar criteriul principal pe baza căruia se realizează clasificarea este dat de numărul de axe ortogonale pe care se realizează achiziția de date[10].

Principiul de funcționare:

Cele mai multe accelerometre sunt senzori micro-electro-mecanici (MEMS). Principiul de bază al operării în spatele accelerometrului MEMS este deplasarea unei structuri de masă gravate în suprafața siliciului circuitului integrat și suspendate de marginile de susținere ale accelerometrului cu ajutorul unor arcuri. Astfel, se realizează o măsurătoare bazată pe poziția masei inerțiale relativ la carcasa accelerometrului.

În concordanță cu a doua lege a mișcării lui Newton ($\vec{F} = m * \vec{a}$), pe măsură ce se aplică o forță ce modifică accelerația aplicată dispozitivului, se dezvoltă o forță care deplasează masa cu aceeași viteză prin comprimarea unuia dintre arcuri. În această formulă, m este masa, $\vec{a}$ este accelerația, iar $\vec{F}$este forța aplicată. Astfel, fluidul (de obicei aerul) prins în interior acționează ca un amortizor, iar poziția actuală a masei e proporțională cu accelerația aplicată.

Accelerometrul poate detecta mișcarea pe baza integrării duble a accelerației măsurate și adăugarea poziției inițiale și a vitezei. Cu toate acestea, deoarece Pământul exercită o accelerație gravitațională asupra tuturor corpurilor, putem folosi și accelerometrul pentru a măsura înclinarea. În cadrul acestui proiect, înclinarea este măsurată folosind un senzor individual, dar toate acestea sunt integrate în modulul MPU6050.

Pentru o mai bună înțelegere a senzorilor pe cele 3 axe, s-a instalat o aplicație open-source din Google Play și au fost date valori aleatoare în funcție de mișcări aleatoare ale telefonului mobil. Câteva grafice realizate în cadrul experimentelor sunt redate în continuare:

Accelerometrele analogice transmit o tensiune variabilă constantă în funcție de cantitatea de accelerație aplicată.

Accelerometrele digitale mai vechi produc un semnal cu frecvență variabilă cunoscut sub numele de modulație în lățime a impulsurilor. Un accelerometru modulat cu lățime în impulsului are citiri la o rată fixă, de obicei 1000 Hz, iar valoarea accelerației este proporțională cu lățimea impulsului (sau ciclul de sarcină) al semnalului PWM.

Noile accelerometre digitale au o probabilitate mai mare de a-și valorifica valoarea utilizând protocoale digitale cum ar fi I2C sau SPI.

Accelerometrele cu ieșire PWM pot fi utilizate în două moduri diferite. Pentru cele mai precise rezultate, semnalul PWM poate fi introdus direct către un microcontroler. Un dezavantaj al unei ieșiri digitale este că necesită mai multe resurse de temporizare și tact ale microcontrolerului.

Măsurarea accelerației are o varietate de utilizări - senzorul poate fi implementat într-un sistem care detectează viteza, poziția, șocul, vibrația sau accelerația gravitației pentru a determina orientarea.

b) Giroscopul

Termenul "giroscop", referit în mod convențional la clasa mecanică de giroscoape, derivă din limbajul grecesc antic, fiind Fizica mișcării de precesie, un fenomen observat și în societatea antică grecească [12].

Giroscopul este un “aparat care, antrenat de o mișcare de rotație în jurul uneia dintre axe, se poate deplasa astfel încât să nu modifice direcția axei sale de rotație” [13].

Senzorul de giroscop măsoară viteza de rotație de-a lungul celor 3 axe principale Roll, Pitch și Yaw. Aceasta depinde de proprietatea masei rotative, așa cum este ilustrat în următoarea schiță schematică a giroscopului mecanic clasic.

Giroscoapele pot fi împărțite în trei categorii principale:

- mecanice
- optice;
- vibrante (MEMS).

Desigur, la fel ca în cazul accelerometrului, senzorii moderni de giroscop utilizează tehnologia MEMS conținută într-un pachet electronic. Același chip poate include atât giroscopul cât și accelerometrul – ca și în cazul nostru - și uneori chiar magnetometrul.

Inițial, domeniul giroscopului făcea referire numai la cele ce implicau rotația unei mase inerțiale, extinzându-se ulterior și incluzând toți senzorii ce măsoară viteze unghiulare și nu necesită un sistem de referință extern [12].

Sistemele de tip micro-electro-mecanic (MEMS) sunt senzori de mișcare care detectează și măsoară mișcarea unghiulară a unui obiect. Ele măsoară rata de rotație a unui obiect în jurul uneia, a două sau a trei axe particulare.

Dacă un giroscop este instalat pe ansambluri care permit masei să navigheze liber în cele trei direcții ale spațiului, axa sa principală de centrifugare va rămâne identică, chiar dacă se schimbă direcția.

Principiul giroscopiei mecanice: Efectul de bază pe care se bazează un giroscop constă în acela că o masă izolată tinde să își mențină poziția unghiulară față de un cadru de referință inerțial și atunci când un cuplu extern permanent (respectiv o viteză unghiulară constantă) este aplicată pe masă, axa sa de rotație suferind o mișcare la o viteză unghiulară constantă (respectiv, cu un cuplu constant de ieșire), într-o direcție care este normală față de direcția cuplului aplicat (respectiv la viteza unghiulară constantă). Practic, când are loc o rotație, rotorul își păstrează orientarea față de sistemul de referință global, modificându-se unghiurile dintre cardane[29]. Deși inițial au fost folosite pentru aplicații militare scumpe, în prezent sunt adoptate și pentru aplicații comerciale cu cost redus, a electronicii de consum pentru aplicații pentru automobile, apărare, industriale și medicale.

Giroscoapele de tip MEMS nu sunt o clasă specifică, ci sunt giroscoape care sunt tipărite pe circuite cu ajutorul fotolitografiei. Fiecare tip de giroscop MEMS are o anumită formă de componentă oscilantă de unde poate fi detectată înclinarea și, prin urmare, schimbarea direcției. Acest lucru se datorează, conform legii privind conservarea mișcării, faptului că unui obiect vibrant îi place să continue să vibreze în același plan și orice abatere vibrațională poate fi utilizată pentru a obține o schimbare de direcție. Aceste deviații sunt cauzate de forța Coriolis, care este ortogonală față de obiectul vibrator. Când o masă (m) se mișcă în direcția $\vec{v}$ și se aplică o viteză de rotație unghiulară $\vec{\Omega}$, atunci masa va avea o forță perpendiculară în direcția săgeții galbene ca urmare a forței Coriolis.

3. Microcontroler – folosit ca unitate de recunoaștere a gesturilor

Datele precizate în această parte sunt preluate din Lucrarea de Diplomă unde am abordat un dispozitiv asemănător.

Un microcontroler este un sistem integrat folosit pentru a prelua anumite caracteristici a mediului înconjurător și pentru a comanda anumite stări ale unui proces în funcție de acestea.

Potrivit [30], resursele integrate includ următoarele componente :

1. Unitate centrală de procesare cu un oscilator intern pentru ceasul de sistem;
2. Memorie de tip ROM, EPROM, Flash și RAM;
3. Porturi I/O – intrări/ieșiri numerice;
4. Un sistem de întreruperi;

Vom defini un microcontroler pornind de la reprezentarea sa în blocuri funcționale:

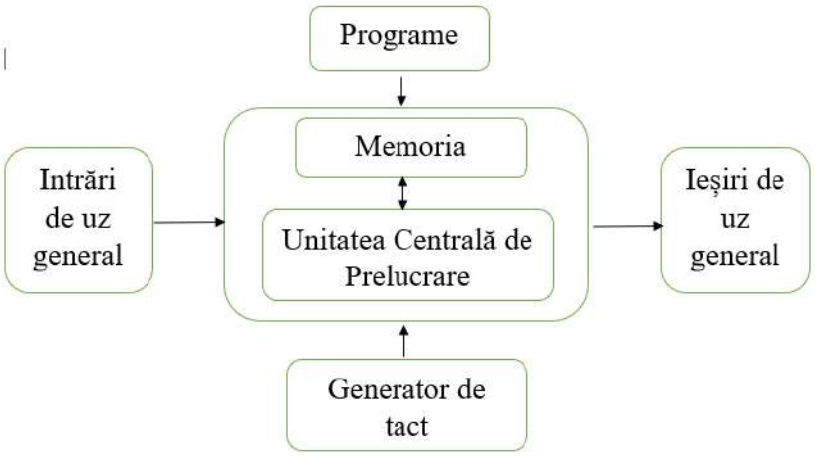


Figura I. 5: Schema simplificată a unui microcontroler

a) Unitatea centrală de prelucrare

Unitatea centrală de prelucrare este secțiunea procesorului care procesează datele, preluând instrucțiunile din memorie, decodificându-le și executându-le. Se compune dintr-o unitate de control (UC), o unitate aritmetică și logică (ALU) și mai multe tipuri de registre.

Unitatea de control (UC) determină momentul începerii unei instrucțiuni și succesiunea operațiilor, astfel generându-se semnalele de sincronizare folosite pentru a prelua o instrucțiune de

program din memorie și pentru a o executa. Unitatea de control este responsabilă cu decodificarea instrucțiunilor conținute în cadrul programului. Astfel, unitatea de control elaborează semnalele pentru a comanda celelalte blocuri funcționale spre a finaliza executarea întregului program sau doar a unei instrucțiuni.

Consultând [16], Unitatea Aritmetică și Logică este responsabilă (așa cum este precizat și în denumire) cu efectuarea operațiilor aritmetice și/sau logice asupra operanzilor furnizați. " Modul de implementare a operațiilor este transparent pentru utilizator. Este important însă timpul de execuție al fiecărei operații pentru a aprecia dacă timpul necesar procesării complete satisface cerințele de timp ale unei aplicații" [16]. ALU este responsabilă pentru efectuarea manipulării efective a datelor. Datele interne pe care unitatea centrală de prelucrare le utilizează în prezent sunt temporar deținute într-un grup de registre în timp ce instrucțiunile sunt executate.

Toate tipurile de microcontrolere prezintă următoarele registre:

- ✓ **Acumulator (registrul acumulator)** – este locul în care sunt stocate temporar datele dintr-o intrare către unitatea aritmetică și logică. Pentru ca procesorul să poată accesa, adică să citească, instrucțiunile sau datele din memorie, trebuie să folosească magistrala de adrese. După ce s-au adus datele din memorie, instrucțiunile pot fi citite și prelucrate de unitatea de control. Întrucât o singură locație de memorie poate fi adresată simultan, trebuie să se utilizeze spațiul de stocare temporară când numerele sunt combinate. De exemplu, într-o adunare de două numere, unul dintre numere este preluat de la o singură adresă și este plasat în registrul de acumulatori în timp ce unitatea de prelucrare și control primește celălalt număr de la cealaltă adresă de memorie. Apoi cele două numere pot fi procesate de secțiunea aritmetică și logică a procesorului, iar rezultatul este transferat înapoi în registrul de acumulatori. Registrul acumulatorilor este, astfel, un registru temporar de exploatare pentru datele care urmează să fie operate de către unitatea aritmetică și logică și, de asemenea, după operație, pentru păstrarea rezultatelor. Prin urmare, acest registru este implicat în toate transferurile de date asociate executării operațiilor aritmetice și logice.

- ✓ **PC (Program Counter)** – registru utilizat pentru a permite procesorului să țină evidența poziției instrucțiunilor într-un program - acest registru conține adresa locației de memorie ce are următoarea instrucțiune de program. Pe măsură ce fiecare instrucțiune este executată, registrul de numărare a programului este actualizat astfel încât să conțină adresa locației de memorie unde este stocată următoarea instrucțiune care trebuie executată. Contorul de programe este incrementat de fiecare dată, astfel încât unitatea centrală de prelucrare execută instrucțiunile secvențial, cu excepția cazului în care o instrucțiune, cum ar fi un JUMP, un CALL sau un BRANCH, modifică contorul de programe din acea secvență.

- ✓ **Registrul de stare** - conține informațiile privind rezultatul ultimului proces realizat în unitatea aritmetică și logică. Acesta conține biți individuali, fiecare bit având semnificație specială. Biții se numesc steaguri (flag-uri), iar starea celei mai recente operațiuni este indicată de steag cu fiecare parametru care este setat sau resetat pentru a indica o anumită schimbare. De exemplu, steagurile pot fi folosite pentru a indica dacă ultima operație a avut un rezultat negativ, un rezultat zero sau o depășire a limitei : de exemplu suma a două numere binare

precum 101 și 110 a condus la un rezultat 011 care de exemplu, este mai mare decât mărimea cuvântului microprocesorului și efectuează o depășire cu 1 – rezultat pe 4 biți).

- ✓ **SP (Stack Pointer)** – registrul indicator de stivă – conținutul acestui registru indică adresa curentă a stivei. Stiva este un set de locații de memorie care pot fi utilizate pentru stocarea datelor de către programatori. De exemplu, un programator sau o operație poate alege plasarea succesivă a mai multor valori sau locații de memorie în stivă. Implementarea acestui proces presupune existența unui registru de adresare (SP) și a mecanismului de memorare declanșat de instrucțiuni specifice (instrucțiunile PUSH/POP).

- ✓ **Registrul de instrucțiuni (IR) -** după preluarea unei instrucțiuni din memorie, unitatea centrală de prelucrare o stochează în registrul de instrucțiuni. Acesta poate fi apoi decodificat și utilizat pentru a executa o operație sau pentru a semnaliza o întrerupere.

- ✓ **Registrele cu scop general -** Acestea pot servi ca stocare temporară pentru date sau adrese și pot fi utilizate în operații care implică transferuri între diferite alte registre.

b) Unitatea de intrări-ieșiri

Una dintre caracteristicile cele mai importante ale microcontrolerului este numărul de pini de intrare / ieșire utilizați pentru conectarea cu periferice. Pentru ca funcționarea pinilor să se potrivească cu organizarea internă pe 8 biți, aceștia sunt grupați în cinci registre numite A, B, C, D și E.

Din motive practice, mulți pini de intrare / ieșire au două sau trei funcții. Fiecare pin poate fi de intrare sau de ieșire prin configurarea acestuia cu comanda TRIS. Astfel, pentru TRIS=0, pinul portului corespunzător va fi configurat ca ieșire, iar pentru TRIS=1, pinul este configurat ca intrare.

Această regulă este ușor de reținut 0 = Output, 1 = Input.

Similar biților din registrul TRISx care determină care dintre pini vor fi configurați ca intrare și care ca ieșire, biții corespunzători ai registrului ANSEL determină dacă pinii vor acționa ca intrări/ieșiri analogice sau intrări / ieșiri digitale.

Fiecare bit al acestui port are o funcție suplimentară legată de unele unități periferice încorporate.

Idei importante:

- Se va selecta un port prin care microcontrolerul va comunica mediul periferic.
- Dacă se va intenționa utilizarea numai a intrărilor / ieșirilor digitale, se va selecta orice port dorit.
- Dacă se va intenționa utilizarea a unei părți din intrările analogice, se vor selecta porturile adecvate care suportă configurația acestor pini (AN0-AN13);
- Fiecare pin de port poate fi configurat ca intrare sau ieșire. Biții TRISA, TRISB, TRISC, TRISD determină modul în care vor acționa pinii porturilor corespunzătoare - PORTA, PORTB, PORTC, PORTD;
- Dacă se vor utiliza unele dintre intrările analogice, se vor seta biții corespunzători registrelor ANSEL la începutul programului;

- Dacă se vor utiliza comutatoarele și butoanele ca sursă de semnal de intrare, acestea se vor conecta la pinii Port B deoarece au rezistoare de pull-up.

Un alt aspect precizat în Figura I 2 este reprezentat de magistrala de adrese. Magistrala este utilizată, în principal, pentru transferul și recepționarea datelor de la un periferic la altul. Există două tipuri de magistrale: de adrese și de date:

Magistrala de date: este folosită pentru a transfera / recepționa numai datele.
Magistrala de adrese: este folosită pentru a transmite adrese de memorie de la periferice la unitatea centrală de prelucrare.

c) Unitatea de memorie

Memoria face parte din microcontroler și funcția sa principală este de a stoca date. Potrivit [31], cel mai simplu mod de a explica memoria este de a o descrie ca pe un "dulap mare cu o mulțime de sertare." Dacă presupunem că am realizat o diferențiere astfel încât sertarele să nu poată fi confundate, orice conținut al acestora va fi ușor accesibil. Este suficient să cunoaștem locația sertarului și astfel conținutul acestuia va fi cunoscut extrem de ușor.

Două noi concepte sunt precizate: adresarea și localizarea memoriei. Memoria este alcătuită din toate locațiile de memorie, iar adresarea sa se realizează prin selectarea uneia dintre ele. Practic, se realizează selecția locației de memorie dorită după care, trebuie să așteptăm conținutul locației pentru a putea utiliza informația. Potrivit [31], pe lângă citirea dintr-o locație de memorie, memoria trebuie să ofere, de asemenea, scrierea pe ea. Aceasta se face prin furnizarea unei linii suplimentare numite linie de control - dacă R/W = 1, se realizează citirea din memorie, altfel, se realizează scrierea în memorie.

Dacă se va dori efectuarea operațiilor cu locații de memorie, vom avea nevoie de registre. Astfel, registrele sunt locații de memorie al căror rol este acela de a ajuta la efectuarea diferitelor operații matematice sau a oricărei alte operații cu date.

Memoria RAM este o memorie volatilă utilizată pentru stocarea temporară a datelor care poate fi citită sau scrisă de unitatea centrală, iar locațiile sunt împărțite în grupuri. Ea ocupă mult spațiu pe chip, iar costurile pentru implementarea ei sunt mari. De aceea, deseori, un microcontroler include puțin RAM.

Registrele RAM sunt împărțite în două tipuri. Acestea sunt registre cu scop general (GPR) și registre cu scop special (SPR).

- GPR: Registrele cu scop general - Acestea sunt utile, de exemplu, dacă vrem să multiplicăm oricare două numere folosind PIC. Această operație se va realiza cu două registre pentru stocarea numerelor, înmulțirea acestora și stocarea rezultatelor. Deci, registrele cu scop general nu vor avea nicio funcție specială, unitatea de control având acces total la datele din registre.
- SFR: Registre cu funcții speciale - Acestea au funcții specifice și, atunci când se folosește unul din aceste registre, el va acționa în funcție de funcțiile care i-au fost atribuite inițial. De menționat ar fi că registrele cu funcții speciale nu pot fi folosite ca registre normale. De exemplu, nu puteți utiliza registrul STATUS pentru stocarea

datelor, registrele STATUS sunt utilizate pentru a afișa starea programului. Utilizatorul nu poate schimba funcția unui registru SFR deoarece aceasta este dată de producător.

Potrivit [16], memoria ROM este o memorie non-volatilă, fiind "cea mai ieftină și cea mai simplă memorie ce se folosește la stocarea programelor din procesul de fabricație. Unitatea centrală poate citi informațiile, dar nu le poate modifica."

Memoria flash poate fi ștearsă și reprogramată în sistemul în care este folosită, fără a fi necesar un sistem dedicat. Trebuie menționat că această memorie nu permite ștergerea unor locații individuale, ci doar întregul conținut.

d) Modulul Timer

Aplicațiile microcontroler-ului implică o multitudine de funcții de timp pe care utilizatorul le poate accesa prin module de timp. Un microcontroler deține un astfel de modul mai mult sau mai puțin complex. Cele mai simple module timer pun la dispoziția utilizatorului "un set de funcții implementate pe baza unui numărător central și a unor blocuri speciale pentru fiecare funcție în parte" [16].

Timer-ul are în structura sa foarte multe registre, iar funcțiile sale pot genera întreruperi independente.

Potrivit [16], "timer-ul este folosit pentru a măsura timpul și pentru a genera semnale cu perioade și frecvențe dorite și nu sunt doar circuite cu funcții de temporizare, ci dețin și câteva mecanisme care pun la dispoziția utilizatorului funcții specifice."

4. Transmițător și receptor RF

a) ESP8266

ESP8266 este un modul Wi-Fi ce poate răspunde cerințelor de performanță ale utilizatorilor printr-o fiabilitate bună și o utilizare eficientă a energiei electrice. Acesta poate funcționa ca un modul de sine stătător, având și o memorie cache de mare viteză, sau poate fi utilizat în aplicații de tip master-slave. Folosește modul de comunicație Wi-Fi prin interfețe SPI și UART, deținând în structura sa și un amplificator de putere, un receptor cu sensibilitate bună și câteva filtre. Modulul încorporează și un modul de emisie-recepție de 2,4 GHz și un amplificator de putere puternic integrat, cu arhitectură de conversie directă și o antenă de referință de frecvență integrată.

Conform fișei tehnice [18], modulul este des folosit în aplicații de tip electrocasnice, IoT, control wireless industrial, monitoare pentru copii, camere IP și rețele de senzori.

Modul de configurare al pinilor

Numărul pinului	Numele pinului	Utilizare
1	**Ground**	Conectează circuitul la masă
2	**TX**	Pin pentru transmisie
3	**GPIO-2**	Pin general de In/Out
4	**CH_EN**	Pin de enable – activat pe High
5	**GPIO - 0**	Pin general de In/Out
6	**Reset**	Resetează modulul
7	**RX**	Pin pentru recepție
8	**VCC**	Se conectează la +3.3V

Tabelul I. 2: Tabel cu modul de configurare al pinilor pentru transmițător și receptor pentru ESP8266

ESP8266 are un suport complet pentru stivă TCP / UDP. De asemenea, poate fi configurat cu ușurință ca server web. Fiecare modul ESP8266 este preprogramat cu un set de comenzi oferind, astfel, un mod de testare inițial pentru a-l putea conecta direct. Modulul acceptă comenzi prin intermediul unei interfețe seriale simple. Apoi, răspunde înapoi cu rezultatul operațiunii (presupunând că totul funcționează corect). De asemenea, după ce dispozitivul este conectat și este setat să accepte conexiuni, acesta va trimite mesaje nesolicitate ori de câte ori o nouă conexiune sau o nouă solicitare este emisă. Acest modul are o capacitate de procesare și stocare suficient de puternică care îi permite să fie integrat cu senzori cu o dezvoltare minimă și o încărcare minimă în timpul perioadei de rulare.

Gradul ridicat de integrare pe chip permite circuite externe minime, inclusiv modulul frontal, este proiectat pentru a ocupa o suprafață minimă de PCB. ESP8266 acceptă aplicații VoIP și interfețe de coexistență Bluetooth, conține un RF auto-calibrat care îi permite să funcționeze în toate condițiile de operare și nu necesită piese RF externe. Modulele ESP8266 pot funcționa:

- ca stație - îl putem conecta la rețeaua Wi-Fi;
- ca un punct de acces soft (soft-AP) - pentru a stabili propria rețea Wi-Fi și pentru a conecta alte stații la modulul ESP;
- atât ca stație cât și ca mod de punct de acces soft - posibilitatea construirii rețelelor complexe – de exemplu, rețelele plasă.

b) nRF24L01

Modulele nRF24L01 sunt module de tip emisie-recepție – fiecare modul poate să trimită și să primească date, dar întrucât sunt semi-duplex, pot trimite sau primi date simultan. Acesta comunică folosind protocolul SPI și, prin urmare, poate fi interfațat cu ușurință cu orice tip de microcontrolere. Dat fiind faptul că, în intermediul proiectului s-a dorit și testarea bazată pe Arduino, acest lucru este realizabil deoarece există multe biblioteci disponibile pentru acesta.

Conform [25], modul de organizare al pinilor este:

Numărul pinului	Abrevierea pinului	Funcția pe care o îndeplinește
1	**Ground**	Conectează masa pinului
2	**Vcc**	Alimentarea la 3.3V
3	**CE**	Folosit pentru a activa comunicația SPI
4	**CSN**	Folosit pentru a menține active comunicația SPI
5	**SCK**	Oferă semnalul de ceas pentru comunicația SPI
6	**MOSI**	Folosit pentru a recepționa date de la microcontroler
7	**MISO**	Folosit pentru a transmite date spre microcontroler
8	**IRQ**	Pin de întrerupere

Tabelul I. 3: Tabel cu modul de configurare al pinilor pentru transmițător și receptor pentru nRF24L01

De obicei, aceste module au o gamă largă de tensiuni de alimentare, pornind de la 3.3V și ajungând și la 12V. De asemenea, aceste module pot funcționa și ca transmițător și ca receptor în funcție de logica în care este programat. Dacă se alege logica "zero", se folosește partea de recepție a modulului, în timp ce purtătoarea de la transmisie este suprimată complet. Dacă se folosește logica "unu" este activate partea de transmisie. Acesta este, însă, unul dintre cazuri și programatorul poate decide dacă folosește modulul astfel sau îl ajustează ca lucrând în logică inversă.

Datele sunt transmise serial de la emițător la receptor și schimbul acestora este coordonat de microcontroler.

Performanța acestui modul și, implicit, transmisia în cadrul proiectului depinde de mai mulți factori precum: distanța între plăcile de circuit de transmisie și de recepție – deoarece, pentru o distanță mai mare, ar trebui ca puterea emițătorului să crească, durata de viață a bateriei – indiferent dacă aceasta este de 12V, 5V sau 3.3V sau stabilirea puterii optime – deoarece o putere mai mare debitată de către transmițător poate determina interferențe în partea de recepție.

5. LCD – folosit ca mod de recunoaștere de tip gesturi-text

O parte din datele precizate în această parte sunt preluate din Lucrarea de Diplomă unde am abordat același tip de dispozitiv.

LCD-ul reprezintă ecranul cu cristale lichide și este unul dintre cele mai frecvente dispozitive utilizate de producătorii de electronică, având o interfață ușor de folosit și accesibilă multor tipuri de microcontrolere.

Multe produse pe care le vedem în viața noastră de zi cu zi dețin o interfață pe bază de LCD. Acestea sunt folosite pentru a arăta starea produsului sau a furniza o interfață pentru introducerea sau selectarea unui anumit proces. Mașina de spălat, cuptorul cu microunde sau aparatele de aer condiționat sunt câteva exemple de produse care au instalat LCD cu caractere sau grafic.

Cu toate că multe companii multinaționale, cum ar fi Philips, Hitachi, Panasonic, fac propriile lor tipuri de LCD pentru a fi utilizate în produsele lor, toate LCD-urile au aceleași funcții de bază (caractere de afișare caractere speciale caractere, caractere ASCII etc). Programarea lor este, de asemenea, aceeași și toate au aceleași 14 pini (0-13), 16 pini (0 la 15) sau 20 pini. Un LCD notat cu 16x2 înseamnă că are 16 coloane și 2 rânduri. Pe un LCD, un caracter este generat într-o matrice de 5x8 sau 5x7. Unde 5 reprezintă numărul de coloane și 7/8 reprezintă numărul de rânduri. Dimensiunea maximă a matricei este de 5x8 și nu e posibilă afișarea unui caracter mai mare decât aceasta. În mod normal, afișăm un caracter în matricea 5x7 și lăsăm rândul 8 pentru cursor. Dacă folosim al optulea rând al matricei pentru afișarea caracterului, atunci nu va mai fi loc pentru cursor.

RS– Registrul de selecție - Comută între comanda și registrul de date.

Când este selectat RS = 0 registrul de comandă înregistrează datele -> Când trimitem comenzi pentru LCD, aceste comenzi se duc la registrul de comandă și sunt procesate.

Când este selectat RS = 1 registru de date procesează datele -> Când trimitem date către LCD, mergem la registrul de date și le procesăm.

Pinul RW (Pinul de Citire - Scriere)

Pin-ul RW este folosit pentru citirea și scrierea datelor în registrele de date și comandă. Atunci când RW = 1 putem citi datele din LCD, iar când RW = 0 putem scrie date în LCD.

Pinul de EN (Pinul de activare)

Atunci când selectăm registrul RS și setăm RW trebuie să executăm instrucțiunea de date sau comenzi. Pentru trimiterea ultimelor date / comenzi prezente pe liniile de date folosim acest pin de activare. De obicei, EN = 0, dar atunci când vrem să executăm instrucțiunea, facem EN = 1 pentru câteva secunde. După aceasta, se revine la EN=0.

V0 (setează contrastul LCD)

Acest pin e utilizat pentru a seta claritatea afișajului LCD, iar cel mai bun mod este de a utiliza un rezistor variabil, cum ar fi un potențiometru. Astfel, se conectează ieșirea potențiometrului la acest pin și se rotește butonul potențiometrului înainte și înapoi pentru a regla contrastul. Datele pot fi trimise utilizând LCD în modul pe 8 sau 4 biți. Dacă se folosește modul pe 4 biți, două semnale de date (primii patru biți și apoi încă patru biți) sunt trimise pentru a finaliza un transfer complet pe 8 biți. Modul pe 8 biți este cel mai bine folosit atunci când este necesară o viteză mai mare într-o aplicație. În modul pe 4 biți, se utilizează numai primii 4 pini de date (4-7). [16]

6. Modul de înregistrare și redare a semnalului audio – folosit ca mod de recunoaștere de tip gesturi-audio

Modulul de înregistrare vocală se bazează pe ISD1820, care este un dispozitiv de tip înregistrare - redare ce poate conține multiple mesaje diferite. Conform fișei tehnice, acest modul poate oferi înregistrări vocale cu un singur chip, stocare volatilă și capacitate de redare timp de 8 până la 20 de secunde. Acest modul de utilizare este foarte ușor și se poate controla direct prin apăsarea butoanelor sau prin microcontroler. Modulul deține și o rezistență ce controlează frecvența oscilatorului și rata de eșantionare pentru durata maximă a mesajului. Astfel, vor apărea două situații: o înregistrare cu o durată mai scurtă, dar cu eșantionare ridicată sau o durată mai lungă sau cu o rată

de eșantionare mai mică. Conform fișei tehnice, modulul este livrat cu un rezistor de 100K ce este conectat la pinul ROSC. Acesta este prezent în circuit printr-un jumper și stabilește o durată de zece secunde pentru o înregistrare. Îndepărtând jumper-ul respectiv, se poate stabili o altă valoare pentru această rezistență și, implicit, o altă durată predefinită pentru înregistrare. *De asemenea este posibilă înregistrarea mesajului cu o anumită frecvență de eșantionare și redarea acestuia cu o altă frecvență, ceea ce reprezintă o caracteristică cheie.*

Pentru a testa dispozitivul, trebuie ca acesta să fie alimentat și să se conecteze difuzorul între pinii SP- și SP+. Prin apăsarea butonului REC se va putea înregistra un mesaj. Apăsând butoanele de redare de tip PLAYE sau PLAYL, sunetul care tocmai a fost înregistrat va putea fi redat.

Cele două moduri de operare sunt declanșarea adresei și declanșarea directă:

1. În modul de declanșare a adresei atât operațiunile de înregistrare cât și cele de redare sunt manipulate în funcție de adresa de început și adresa de final;
2. În modul de declanșare directă, dispozitivul poate configura memoria până la opt mesaje egale. Cu funcția de înregistrare sau redare selectată în prealabil, fiecare mesaj poate fi accesat la întâmplare prin intermediul butonului său de control al mesajului.

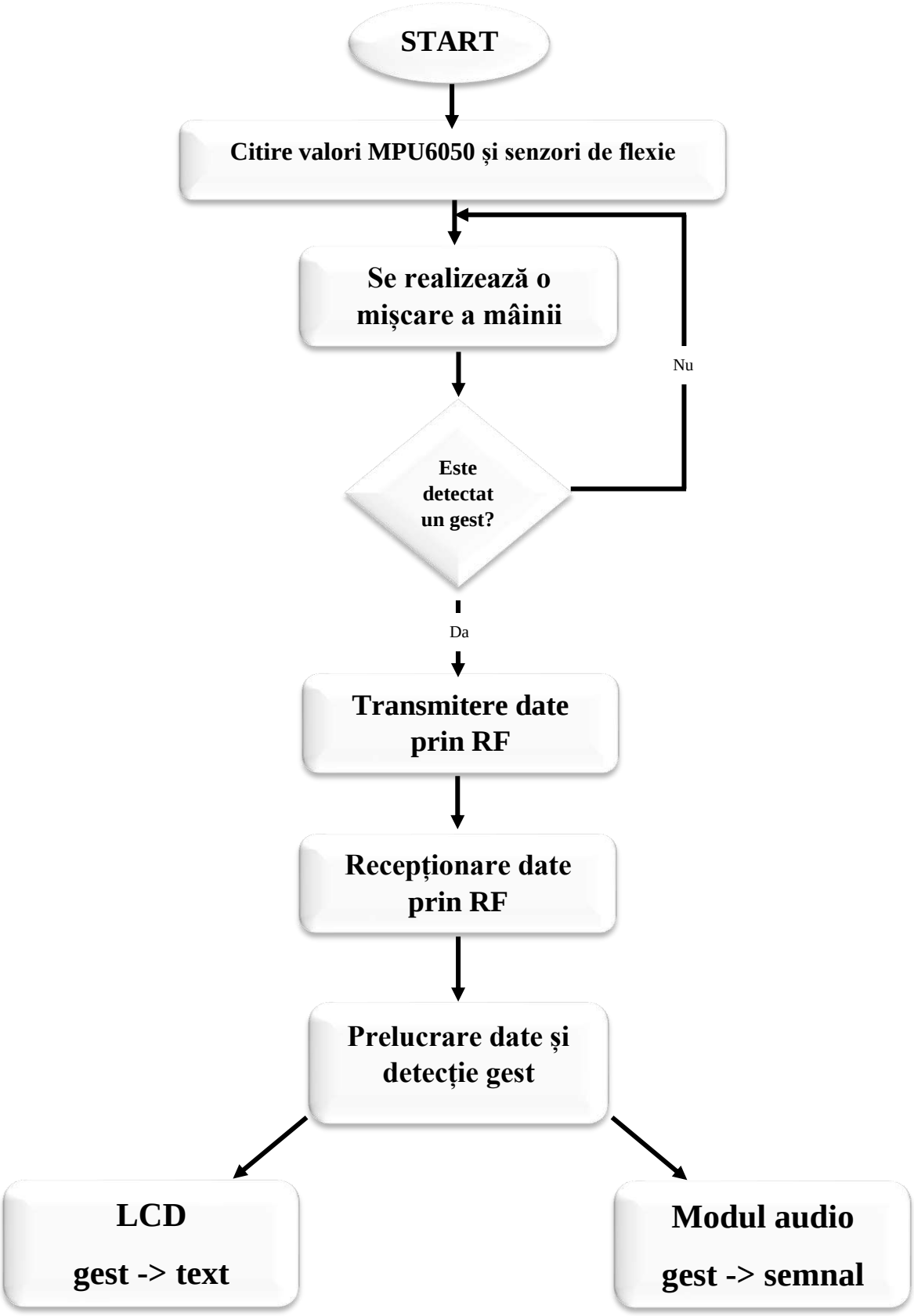
Modul de configurare al pinilor [32]

Nr. pin	Numele pinului	Utilizare
1	**Vcc**	Se conectează la +3.3V
2	**Ground**	Conectează circuitul la masă
3	**REC**	Intrarea REC este un semnal de înregistrare activ pe frontul HIGH. Acesta are prioritate asupra semnalului de redare (PLAYL sau PLAYE).
4	**PLAYE**	PLAYE – Redare - Este un semnal activat pe palier - este detectată o tranziție continuă până când se întâlnește un marker de tip End-of-Message (EOM) sau se ajunge la sfârșitul spațiului de memorie.
5	**PLAYL**	PLAYL – Redare - activat la nivel - când acest pin trece de pe LOW pe HIGH, este inițiat un ciclu de redare
6	**Pini de ieșire spre difuzor**	Pinii SP + și SP- oferă o acțiune directă pentru difuzoarele puternice, cu impedanțe de până la 8Ω
7	**MIC- Microfon**	Intrarea microfonului ce transferă semnalul către preamplificatorul de pe chip.
8	**FT – Alimentare directă**	Acest mod permite microfonului să acceseze direct difuzorul.
9	**P-E**	Redați înregistrarea la nesfârșit.

Tabelul I. 4: Tabel cu modul de configurare al pinilor pentru modulul de înregistrare-redare a semnalului audio

C) Descriere parte software

1. Diagrama software a sistemului



2. Explicații asupra algoritmului

Programul ce se regăsește în anexe reprezintă codul sursă al proiectului ce are mai multe părți și anume: partea de achiziție, de procesare și de transmitere efectivă a comenzilor către modemul RF, precum și partea de recepție a mesajului, afișarea pe LCD și declanșarea sistemului ce conține modemul audio. Toate variabilele sunt declarate la început, iar apoi se face inițializarea comunicației seriale, inițializarea variabilelor și se definesc regiștrii.

Într-o buclă se realizează citirile succesive ale datelor de la senzori, filtrarea și comenzile de transmitere pentru circuitul ce conține partea de transmisie, iar, la recepție, tot într-o singură buclă de procesare se realizează recepția, afișarea pe ecranul LCD și transmiterea semnalului audio.

Pentru a executa o instrucțiune, microcontroler-ul va trebui să fie configurat și programat în mediul MPLAB IDE, MicroC sau Arduino IDE [în partea de testare a circuitului], unde se încarcă diverse biblioteci predefinite, se definesc anumiți parametri și se inițializează anumite registre ale microcontroler-ului. La finalul acestor inițializări, se va realiza programul efectiv ce va conține gesturile predefinite ce vor fi transmise prin intermediul unui mesaj definit de tip șir de caractere.

Fișierele cu terminație de tip .c sau .ino sunt cele ce vor conține programul propriu-zis (main) în limbajul C dedicat.

Pașii esențiali în etapele de programare ale proiectului ar putea fi :

- Se alege limbajul de programare utilizat ulterior – în cazul nostru, limbajul C;
- Se creează un fișier sursă ce se salvează cu extensia .c sau .ino;
- Se încarcă bibliotecile necesare;
- Se inițializează regiștrii sau variabilele necesare;
- Se realizează programul efectiv
- Se rulează programul cu ajutorul compilatorului C și se testează
- Opțional, se realizează depanare – unele microcontrolere nu pot utiliza anumite funcții de afișare [de exemplu, sprintf], astfel că datele de tip int sau float trebuie transformate în șiruri de caractere;
- Pentru microcontrolerul de tip PIC, se realizează fișierul .hex pentru scrierea microcontroler-ului și se realizează fișierul .cof.

3. Exemple de funcții în Arduino IDE

a) Funcție de calibrare a valorilor modulului MPU6050 [27]

```
void calibrare_valori(MPU6050 x)
{
    int valoare;
//valori corespunzătoare accelerometrului
```

offset_x_a=-medie_x_a/8; //*are valoare negativă deoarece, ulterior, se adună fix cu aceeași valoare pentru va rezulta un decalaj = 0*

offset_y_a=-medie_y_a/8; //*comentariu identic cu rândul anterior*

offset_z_a=(16384-medie_z_a)/8; //*se dorește ca valorile de decalaj ale dispozitivului să fie zero. Această condiție se respectă în cazul axelor x și y deoarece dispozitivul este în repaus, dar, în cadrul axei z, dispozitivul va măsura +/- 1g. Astfel, se realizează o compensare a acestei valori scăzând-o din valoarea maximă corespunzătoare fișei tehnice.*

De asemenea, diviziunea cu 8 se realizează corespunzător cu valorile unei precizii mai ridicate a măsurătorilor.

//valori corespunzătoare giroscopului

offset_x_g=-medie_x_g/4; // *are valoare negativă deoarece, ulterior, se adună fix cu aceeași valoare pentru va rezulta un decalaj = 0*

offset_y_g=-medie_y_g/4; //*diviziunea cu 4 se realizează corespunzător cu valorile aferente unei precizii mai ridicate a măsurătorilor.*

offset_z_g=-medie_z_g/4;

```
do{
 valoare=0;
 do{
  x.getMotion6(&val_x_a, &val_y_a, &val_z_a, &val_x_g, &val_y_g, &val_z_g); //funcție prezentă implicit în limbajul de programare Arduino. O altă funcție pe care puteam să o folosim în cadrul acestei lucrări este getMotion9 ce conținea și valorile aferente magnetometrului.

//Se realizează mediile valorile senzorilor comparând valoarea anterioară cu cea curentă -> se poate observa dacă este nevoie de filtrare a valorilor.
  medie_x_a = (val_anterioara_x_a + val_x_a)/2;
  medie_y_a = (val_anterioara_y_a + val_y_a)/2;
  medie_z_a = (val_anterioara_z_a + val_z_a)/2;
  medie_x_g = (val_anterioara_x_g + val_x_g)/2;
  medie_y_g = (val_anterioara_y_g + val_y_g)/2;
  medie_z_g = (val_anterioara_z_g + val_z_g)/2;
  }
```

```
while(++i<=val_buffer); // această condiție se referă la faptul că i nu trebuie să depășească valoarea predefinită pentru buffer.


      if (abs(medie_x_a)<=8) //Dacă valoarea mediei, în modul, este mai mică decât cea implicită
     valoare++; //se incrementează valoarea arbitrară
  else
     offset_x_a=offset_x_a-medie_x_a/8; //Dacă nu se respectă condiția anterioară, se realizează o scădere pentru o recalibrare a decalajului
//Aceleași aspecte sunt valabile și pentru axele y și z
      if (abs(medie_y_a)<=8)
     valoare++;
      else
     offset_y_a=offset_y_a-medie_y_a/8;
      if (abs(16384-medie_z_a)<=8)
     valoare++;
      else
     offset_z_a=offset_z_a+(16384-medie_z_a)/8;


//Abordarea este identică și în cazul valorilor date de giroscop
      if (abs(medie_x_g)<=1) //Eroarea admisa pentru giroscop cu scopul de a obține mai multă precizie - implicit 1
     valoare++;
      else
     offset_x_g=offset_x_g-medie_x_g/(1+1);
      if (abs(medie_y_g)<=1)
     valoare++;
      else
     offset_y_g=offset_y_g-medie_y_g/(1+1);
      if (abs(medie_z_g)<=1)
     valoare++;
```

```
    else
   offset_z_g=offset_z_g-medie_z_g/(1+1);
  }while(valoare!=6); //Este nevoie de 6 valori – 3 corespunzătoare accelerației și 3 pentru giroscopie
}
```

b) Funcție specifică a LCD-ului

```
LiquidCrystal_I2C lcd(I2C_ADDR, EN,RW,RS,D4,D5,D6,D7); //se folosește LCD-ul prin intermediul bibliotecilor și ale funcțiilor de I2C deoarece deține, atașat, și un modul pentru comunicație I2C. Astfel, se definește adresa I2C, pinul de enable, pinul de Read-Write, Pinul de
```

Selecție a registrului și cei patru pini de date.

```
void setup()
 {
  lcd.begin (16,2); //se folosește un afișaj de tip 2 rânduri și 16 coloane
  lcd.setBacklightPin(BACKLIGHT,POSITIVE); //se activează luminiozitatea LCD-ului pentru a putea observa caracterele afișate
  lcd.setBacklight(HIGH);
  lcd.setCursor(0,0);//se setează cursorul pentru afișaj de la prima poziție a rândului poziționat superior
  lcd.print("Recunoastere"); //se setează mesajul ce urmează a fi afișat pe primul rând
  lcd.setCursor(0,1); //se setează cursorul pentru afișaj de la prima poziție a rândului poziționat inferior
  lcd.print("de gesturi");//se setează mesajul ce urmează a fi afișat pe cel de-al doilea rând
 }
```

c) Funcție specifică pentru modulul de emisie-recepție nRF24L01

```
RF24 radio(5, 6); //Definirea pinilor denumiți CE și CSN în cadrul plăcii Arduino Nano

const byte address[6] = "00001"; //Adresa corespunzătoare celor două module de emisie-recepție => va rezulta că doar cele două module ce dețin această adresă pot comunica
```

//la transmisie

```
void setup()
{
   radio.begin(); //Comandă pentru activarea modulului
```

```
   radio.openWritingPipe(address); //Comandă pentru setarea adresei pentru
   comunicație între dispozitive

   radio.stopListening(); //setează modulul ca emițător
}

void loop()
{
   const char s [] = "Proiect de Disertație"; //textul ce urmează a fi
   transmis

   radio.write(&s, sizeof(s)); //se va transmite mesajul către receptor

   delay(1000);
     }
```

// la receptie

```
void setup()
{
    Serial.begin(9600);

   radio.begin(); //Comandă pentru activarea modulului

   radio.openWritingPipe(0, address); //Comandă pentru setarea adresei
   pentru comunicație între dispozitive

   radio.startListening(); //setează modulul ca receptor
}

void loop()
{
   if (radio.available()) //se verifică dacă există date ce trebuie recepționate
   {
     char s[48] = ""; //șirul în cadrul căruia va fi stocat mesajul primit de la emițător
   ce va conține, la o primă analiză 48 caractere

     radio.read(&s, sizeof(s)); //se "citește" mesajul recepționat și se afișează
   prin intermediul Serial Monitor pe ecran

       Serial.println(s);
   }

}
```

I.2 Abordarea bazată pe recunoaștere vizuală prin Android

ANDROID

Android este un sistem de operare bazat pe Linux ce este utilizat pentru dispozitive mobile, fiind dezvoltat de către Open Handset Alliance și preluat de Google. Acesta oferă aplicații dedicate pentru această platformă, dar și open-source, iar licențele pentru codul sursă sunt gratuite. Mediul de dezvoltare include șabloane de cod pentru funcții comune ale aplicației, instrumente de testare ample și un sistem de construire flexibil. [29]

Modul de dezvoltare a unei aplicații Android cuprinde mai mulți pași:

1. Definirea ideii și a cerințelor aplicației
2. Specificarea părților de interfață cu utilizatorul – include modul în care va arăta aplicația, diverse tipuri de layout, diverse funcționalități particulare;
3. Developarea și testarea aplicației: resursele cu extensia .xml – ce includ elemente de interfațare cu utilizatorul, diverse meniuri, butoane, imagini –, componente și teste în Java sau Kotlin, configurații pentru testare și resurse externe

Proiectul Android trebuie să țină cont de dispozitivele în cadrul căruia urmează să fie rulat deoarece trebuie setat, încă de la început, SDK (Software Development Kit) minim. Pentru ca aplicația ce urmează a fi dezvoltată să poată fi rulată pe 97% din dispozitivele existente pe piață, este indicat ca nivelul minim să fie PI 15: Android 4.0.3 (IceCreamSandwich). Fiecare versiune ulterioară adaugă niveluri de complexitate suplimentară. De asemenea, Android Studio deține și o varietate de șabloane pentru a simplifica modul de definire a aplicațiilor – de la șablon gol, până la cele cu diverse tipuri de activități predefinite.

Fereastra de lucru este alcătuită din mai multe zone:

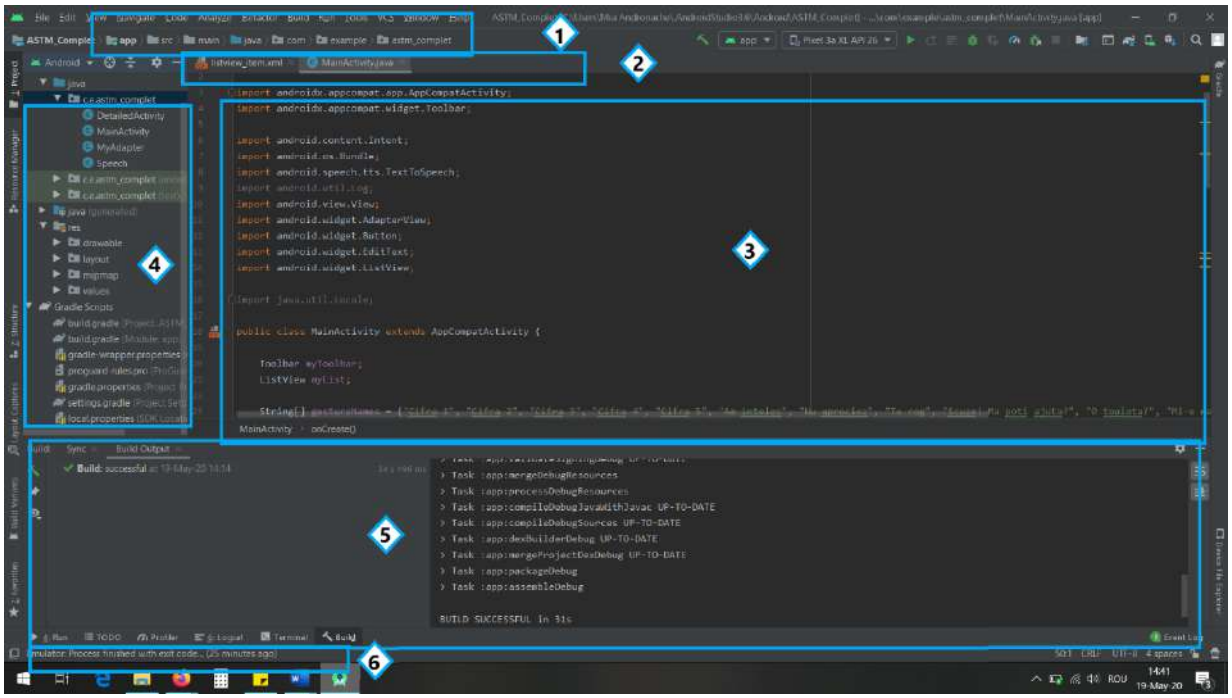


Figura I. 6: Zone de lucru în fereastra Android

1. Bara de instrumente: oferă o gamă largă de acțiuni, inclusiv rularea aplicației Android și lansarea instrumentelor Android.
2. Bara de navigare: permite navigarea prin proiect și prin fișierele deschise pentru editare, oferind o vedere mai compactă a structurii proiectului.
3. Panoul de Editare a fișierului: arată conținutul unui fișier selectat în proiect și este modalitatea prin care este editat codul fișierului.
4. Panoul pentru proiect: arată modul în care este ierarhizat proiectul în curs.
5. Panoul pentru monitorizare: oferă acces la lista TODO (de gestionare a sarcinilor), gestionează execuția aplicațiilor și oferă suport pentru Terminal.
6. Bara de stare: afișează starea proiectului și orice avertisment sau mesaj. Puteți urmări progresul proiectului în bara de stare. Bara de stare afișează un feedback relevant și sensibil la context, cum ar fi informații despre orice proces în curs sau starea depozitului sau a proiectului. Să explorăm acum bara de stare în detaliu:
    - Toggle Margins – se comuta ascunderea și afișarea marginilor. Astfel, atunci când treceți mouse-ul peste acest buton, apare un meniu contextual care vă permite să activați oricare dintre ferestrele instrumentului.
    - Zona de mesaj este utilizată pentru a oferi un feedback și pentru a afișa orice informații despre procesele care rulează simultan.
    - Poziția cursorului Editor afișează locația cursorului în Editor în format linie-coloană. Iar făcând clic pe această zonă se activează o casetă de dialog care vă permite să navigați direct la o anumită linie din codul dvs.
    - Zona formatului text descrie codificarea textului folosit pentru fișierele sursă, iar valoarea implicită este UTF-8, care este în limbaj ASCII, incluzând majoritatea alfabetelor occidentale, inclusiv caracterele pe care le puteți găsi într-un fișier Java standard sau XML.
    - Zona indicatorului de acces la fișier vă permite să comutați între citire/scriere și numai citire. Puteți comuta aceste setări făcând clic pe pictograma indicatorului.
    - Butonul Highlighting Level activează o casetă de dialog cu un glisor care permite setarea nivelului de evidențiere.

Fiecare proiect din Android Studio conține fișierul AndroidManifest.xml ce este un fișier de cod sursă, și alte componente, fișiere sau resurse asociate. În mod implicit, Android Studio organizează fișierele proiectului pe baza tipului de fișier și le afișează în cadrul Proiectului. Acest mod oferă acces rapid la fișierele cheie ale proiectului dumneavoastră.

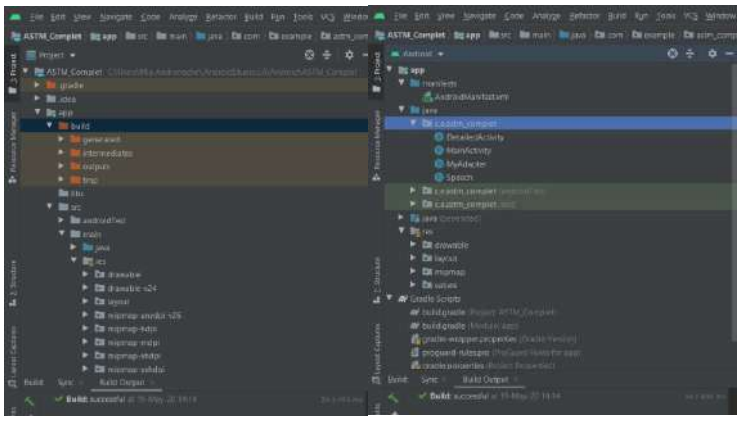


Figura I. 7: Ierarhia proiectului Android

Fișierul Android Manifest.xml este folosit pentru specificarea informațiilor cu privire la aplicația pentru mediul Runtime. Acest fișier este generat automat la alegerea șablonului pentru aplicație.

Directorul ce conține cuvântul cheie java include activități, teste și alte componente din codul sursă Java. Fiecare activitate, serviciu și altă componentă este definită ca o clasă în limbajul obiect orientat. Aplicația trebuie să declare toate componentele sale în acest fișier, care trebuie să fie la rădăcina directorului proiectului aplicației.

Numele primei activități (ecran) pe care o observă utilizatorul, care inițializează, de asemenea, resursele din toată aplicația, este, de obicei, MainActivity. În cadrul exemplului anterior, există mai multe aplicații de acest tip deoarece codul din cadrul acestui proiect cuprinde mai multe fire de execuție.

Dosarul res conține resursele aplicației precum siruri de caractere și imagini. O activitate este de obicei asociată cu un fișier de resurse .xml unde se specifică aspectul vizualizărilor sale. Acest fișier este numit de obicei după activitatea sau funcția sa.

Gradle Scripts cuprinde fișiere de configurare specifice aplicației generate. Șablonul ales creează acest fișier, iar, în cadrul lui, se definește configurația, inclusiv atributele minSdkVersion (care declară versiunea minimă pentru aplicație) și targetSdkVersion (care declară cea mai nouă versiune) pentru care aplicația a fost optimizată.

În partea superioară a fișierului MainActivity.java este o declarație a pachetului care definește aplicația. Acesta este urmat de un bloc de import în care sunt declarate bibliotecile necesare. Un exemplu, în acest caz, ar fi: import android.support.v7.app.AppCompatActivity;

Android Studio deține o interfață grafică ce conține design-ul layout-ului. Această vizualizare oferă un panou Paletă cu elemente de interfață utilizator și o grilă care prezintă aspectul ecranului.

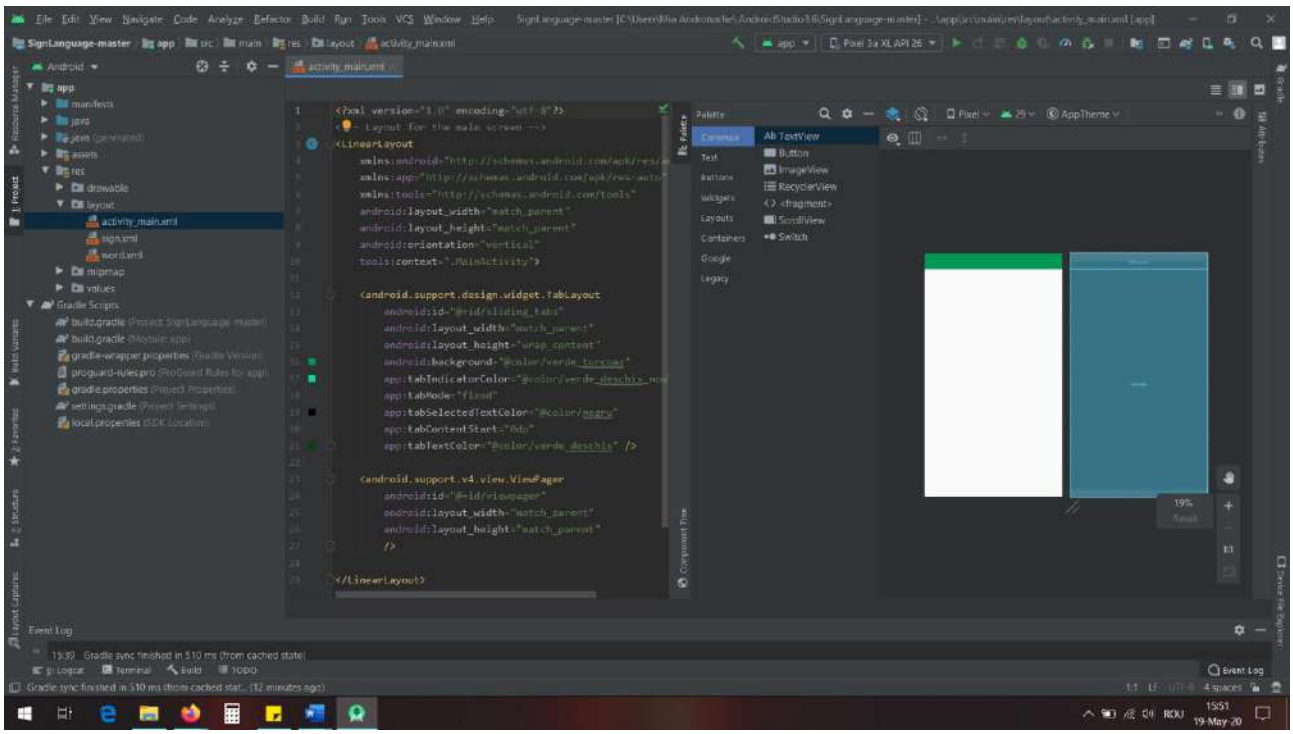


Figura I. 8: Interfață grafică și mod de organizare a layout-ului în proiectul Android

Utilizarea dispozitivelor virtuale:

Android permite crearea unor dispozitive virtuale Android (AVD), prin intermediul cărora se poate emula un dispozitiv pe computer. Există o distincție importantă, dar subtilă, între simulare și emulare. Simularea înseamnă că dispozitivul virtual este doar o fațadă care simulează modul în care se poate comporta un dispozitiv fizic efectiv, dar nu rulează sistemul de operare vizat. Emularea face ca dispozitiv de pe care se rulează Android Studio (PC, laptop) să rezerve un bloc de memorie pentru a reproduce mediul găsit pe dispozitivul pe care îl emulează. Deși emulația oferă un mediu mult mai fidel pe care să-ți testezi aplicațiile decât simularea, declanșarea unui AVD este extrem de consumatoare de resurse, fiind nevoie de o capacitate mare de procesare pentru o rulare corectă a aplicației. Cu toate acestea, se recomandă testarea aplicațiilor pe un dispozitiv fizic, în cazul în care deținești unul.

Clasa View reprezintă blocul de bază de bază pentru toate componentele ce conțin partea de interfață cu utilizatorul deținând componente interactive precum:

- Text (TextView) - câmpuri de introducere și editare a textului;
- Butoane și alte componente interactive
- Listă derulabilă
    - ScrollView - conține tipuri de moștenire în care copiii sunt de tip moduri de vizualizare derulabile
    - RecyclerView - conține o listă de alte vizualizări sau grupuri de vizualizare și permite derularea acestora prin adăugarea și eliminarea dinamicii de pe ecran.
- Imagini (ImageView)

Grupurile de tip Layout view – sunt organizate într-o ierarhie a cărei rădăcină este formată printr-un ViewGroup care conține aspectul întregului ecran.

- LinearLayout: mod de vizualizare poziționat orizontal sau vertical
- RelativeLayout: mod de vizualizare poziționat în raport cu alte vizualizări din grup. Cu alte cuvinte, pozițiile copiilor pot fi descrise unul în raport cu celălalt sau în funcție de grupul părinților.
- ConstraintLayout: un grup de vizualizări ale copiilor care folosesc margini și ghiduri de orientare pentru a controla modul în care vizualizările sunt poziționate în raport cu alte elemente de aspect.
  ConstraintLayout a fost proiectat pentru a facilita glisarea și plasarea imaginilor în editorul de aspect.
- TableLayout: Un grup de vizualizare orientat în rânduri și coloane.
- AbsoluteLayout: Un grup care permite specificarea locațiile exacte în coordonate absolute pe cele două axe. Dispunerile absolute sunt mai puțin flexibile și mai greu de întreținut comparativ cu celelalte.
- FrameLayout: proiectat pentru a bloca o zonă de pe ecran pentru a afișa o singură vizualizare.
- GridLayout: Un grup care plasează zonele într-o grilă dreptunghiulară, care poate fi glisat.

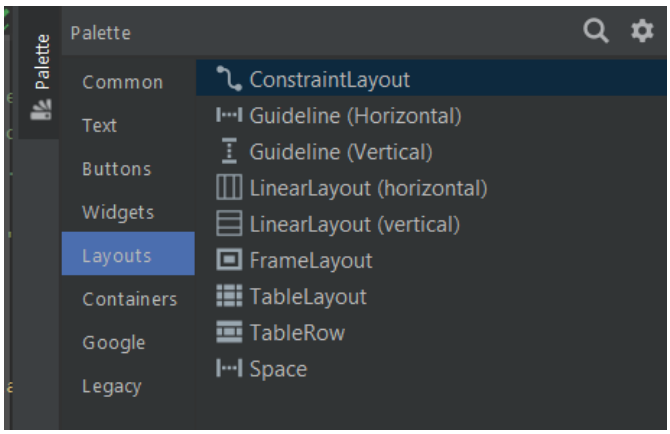


Figura I. 9: Tipuri de layout în Android

CONCEPTE DE MACHINE LEARNING

Învățarea automată folosește modele matematice cu scopul de a răspunde la întrebări specifice folosind date. În viața cotidiană, există multe metode pentru învățarea automată precum detectarea email-urilor spam, recomandări de produse pentru clienți și pentru diverse statistici.

Acest concept se bazează pe rețele neuronale multistrat - modele incredibil de flexibile care pot folosi o varietate uriașă de tehnici matematice diferite. Acestea necesită, însă, o putere de calcul destul de mare pentru a oferi rezultate plauzibile și pentru a mări capacitățile unui proces.

Învățarea automată se bazează pe conceptul de model care dă o mare flexibilitate în a decide modul de utilizare a datelor pentru a avea cel mai bun rezultat. Acesta poate lua toți parametrii de intrare și poate determina automat combinații utile, de ordin superior, ale acestora.

Acest aspect permite un proces de luare a deciziilor mult mai sofisticat, făcând computerele să fie capabile de procese precum conducerea autonomă a unui autoturism, telefoane ce pot determina vocea proprietarului, traducerea automată a unor cuvinte în text, recunoașterea fețelor, etc.. Aceste concepte sunt relativ noi fiind dezvoltate o serie de biblioteci de programare pentru modele profunde în ultimii ani. Însă, o parte din acestea, nu sunt însă pregătite pentru producție deoarece sunt uneori mult prea lente sau nu dețin suficiente moduri de antrenament pentru un rezultat corect.

La o privire mai atentă asupra subiectului, putem observa că este nevoie de puțini pixeli de informații pentru a putea identifica subiectul unei imagini. Însă recunoașterea acestor structuri este destul de dificilă pentru multe funcții existente în literatură. Dacă ne referim și la partea video, unde sunt încadrate mai multe imagini într-o perioadă scurtă de timp, vom observa că o mare parte din inteligența artificială se referă la planificarea sau deliberarea autonomă a sistemelor robotice pentru a naviga printr-un mediu. O înțelegere detaliată a acestor medii este necesară pentru a naviga prin ele – acesta este exemplul cel mai firesc pentru modul în care funcționează vehiculul autonom.

Deseori, în cadrul cercetării efectuate asupra acestui subiect, am observat diferiți algoritmi pentru procesarea și analizarea imaginilor. Aceste tehnici tind să se concentreze pe imagini 2D prin operații efectuate la nivelul pixelilor, cum ar fi îmbunătățirea luminozității imaginii și a contrastului, eliminarea zgomotului sau transformări geometrice, cum ar fi rotirea imaginii. Această caracterizare

implică faptul că procesarea sau analiza imaginii nu necesită presupuneri și nici nu produce interpretări despre conținutul imaginii.

În conceptele bazate pe machine learning, imaginile sunt procesate mai intens. Un exemplu, în acest caz, ar fi reconstruirea unei structuri dintr-o scenă 3D sau dintr-un video corupt. Astfel, viziunea autonomă se bazează pe presupuneri complexe asupra unei scene sau a unei părți din respectiva imagine. De asemenea, dacă se implică senzori de imagine, procesarea valorilor furnizate de aceștia, în timp real, cel mai adesea, este tot un concept bazat pe machine learning. Un alt exemplu, tot în cadrul acestui caz, ar fi condițiile slabe de luminozitate ce pot fi corectate prin intermediul acestei categorii de algoritmi.

În literatură, sunt mai multe tipuri de aplicații a acestor concepte:

- Recunoașterea obiectelor: poate fi recunoscut unul sau mai multe obiecte pre-specificate sau învățate sau mai multe obiecte identic plasate în cadrul unei imagini;
- Identificare: o instanță individuală a unui obiect este recunoscută – cel mai cunoscut exemplu, în acest caz, poate fi considerat recunoașterea feței implementată deja în cadrul ultimelor generații de telefoane mobile sau a aplicațiilor pe care acestea le conțin;
- Detecție: datele prezente în cadrul acestor imagini sunt scanate cu scopul de a testa o anumită condiție pre-specificată. Cele mai cunoscute exemple își au originea în imagistica medicală fiind aplicabilă țesuturilor sau celulelor.

Dacă se consideră o aplicabilitate mai specifică a metodelor de procesare a imaginii, acestea pot fi împărțite în:

- Extragerea unui conținut specific: această metodă se bazează pe livrarea unui set de imagini și a subiectului căutat pentru care algoritmul livrează imagini cu similitudini sau care conțin sau nu subiectul. Cel mai adesea, acești algoritmi se găsesc în aplicații criminalistice;
- Recunoașterea caracterelor unui text – în cadrul aplicațiilor ce codează texte sau a aplicațiilor ce necesită identificarea unui anumit număr de înmatriculare a unui autoturism;
- Estimări și corectări ale poziției unui obiect în raport cu camera video/fotografică – în aplicații de automatizare a unor procese ce necesită diverse manevrări ale utilajelor.

Obiectele și pozele ce prezintă anumite obiecte pot servi ca modele pentru a facilita recunoașterea acestora. Multe sarcini importante de recunoaștere a imaginilor se bazează strict pe context. Astfel, având în vedere un set de imagini de antrenament, modelul descoperă automat pozițiile relevante pentru fiecare tip de alt obiect asemănător și, în plus, creează o bază pentru o detecție ulterioară, după un anumit interval de timp. Operația de identificare/recunoaștere implică o căutare exhaustivă în cadrul imaginilor testate, iar cele mai populare metode sunt, încă, cele bazate pe ferestre glisante.

Este necesar să se implementeze o metodă de detecție universală, dar, momentan, această performanță este destul de greu de atins datorită diversității aplicațiilor ce vor urma să o implementeze. Se fac, însă, progresa mari în cadrul metodelor de estimare, apărând destul de multe aplicații de acest tip ce dețin o bază programabilă comună.

Un pas esențial în recunoașterea obiectelor în cadrul procesării de imagini este “corectarea” imaginii. Acest aspect poate facilita destul de mult recunoașterea modelelor prin eliminările

zgomotului sau îmbunătățirea calității. Astfel, este esențial ca, înainte de procesarea imaginii prin conceptul de machine-learning, să se realizeze o procesare locală a imaginilor de antrenare și/sau test.

Cele mai simple metode pentru îndepărtarea zgomotului, la nivelul imaginii, sunt cele bazate pe filtre mediane sau FTJ.

TENSORFLOW

TensorFlow Lite este soluția dezvoltatorilor TensorFlow pentru dispozitive mobile ce permit interfațare cu latență scăzută. Acesta permite utilizarea API-urilor de tip Android Neural Networks pe platforme de Android și IOS.

În cadrul acestui proiect s-au integrat TensorFlow cu Android Studio IDE și instrumente pentru procesarea imaginilor.

Metoda de învățare automată, ce se va urmări în cadrul acestui proiect, se dezvoltă pe bază de modele. Un model este un mod de reprezentare fixă a datelor din cadrul unui anumit proiect ce este dezvoltat pentru a formata anumite date și este reutilizat pentru operații predefinite. Modelul de calcul pentru TensorFlow este un grafic direcționat, în care nodurile sunt funcții / calcule, iar săgețile sunt numere, matrici sau tensori.

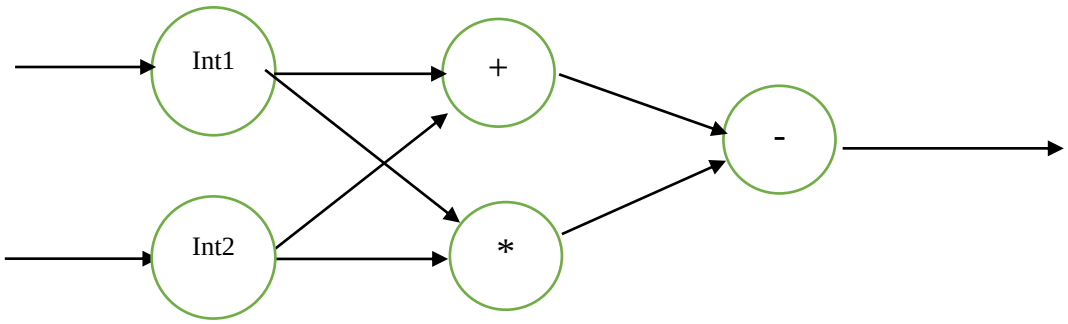


Figura I. 10: Grafic de flux pentru operațiile din TensorFlow

În cadrul imaginilor, TensorFlow necesită zeci sau sute de cadre ale unui obiect pentru a antrena un clasificator de detecție bun. Imaginile de antrenament trebuie să dețină atât obiecte aleatoare, cât și obiectele dorite, mai multe fundaluri diferite și diverse condiții de iluminare (de la condiții de iluminare foarte bună, până la imagini parțial obscure).

Modul de realizare a acestor imagini trebuie să fie caracteristic clasificatorului, imaginile trebuie să fie captate cu un dispozitiv foto sau să fie descărcate de pe internet. Acestea nu trebuie să aibă o dimensiune mare deoarece vor încetini instruirea clasificatorului. Dacă imaginile inițiale sunt mari comparativ cu cerințele proiectului, se poate realiza o redimensionare software a acestora chiar în cadrul programului.

Pentru ca recunoașterea să aibă rezultate bune, este necesar ca antrenarea să cuprindă cel puțin 70-80% din totalitatea imaginilor, iar recunoașterea să fie cu 20-30% dintre acestea. Este important,

de asemenea, ca aceste imagini să fie cât mai diferite, în cadrul ambelor părți, pentru o detecție realizată corect.

Pentru etichetarea imaginilor, se pot folosi diverși algoritmi open-source sau chiar metode integrate în Android Studio. Un exemplu, în acest caz, poate fi urmărit în cadrul următorului exemplu:

```
private class YourAnalyzer implements ImageAnalysis.Analyzer {

    private int degreesToFirebaseRotation(int degrees) {
        switch (degrees) {
            case 0:
                return FirebaseVisionImageMetadata.ROTATION_0;
            case 90:
                return FirebaseVisionImageMetadata.ROTATION_90;
            case 180:
                return FirebaseVisionImageMetadata.ROTATION_180;
            case 270:
                return FirebaseVisionImageMetadata.ROTATION_270;
            default:
                throw new IllegalArgumentException(
                        "Rotation must be 0, 90, 180, or 270.");
        }
    }[23]
```

După etichetarea imaginilor, se vor genera datele de intrare pentru modelul de antrenare. Astfel, datele din formatul .xml sunt utilizate pentru a crea fișiere .csv care conțin toate datele pentru antrenare și testare.

CAPITOLUL II

II.1 Implementare abordarea bazată pe achiziție de date

Schema bloc a circuitelor implementate

1. **Partea de transmisie**

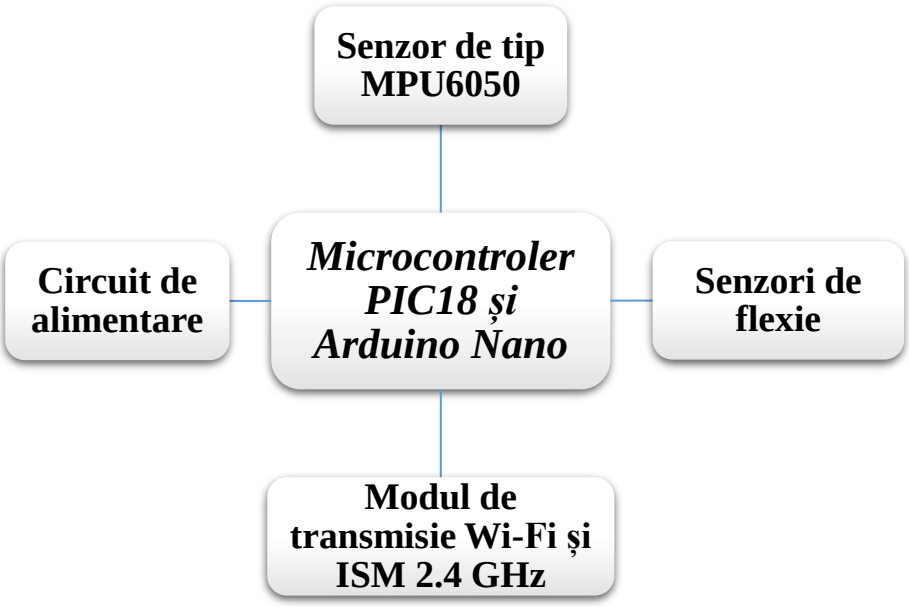


Figura II. 1: Schema bloc a circuitului de transmisie

2. Partea de recepție

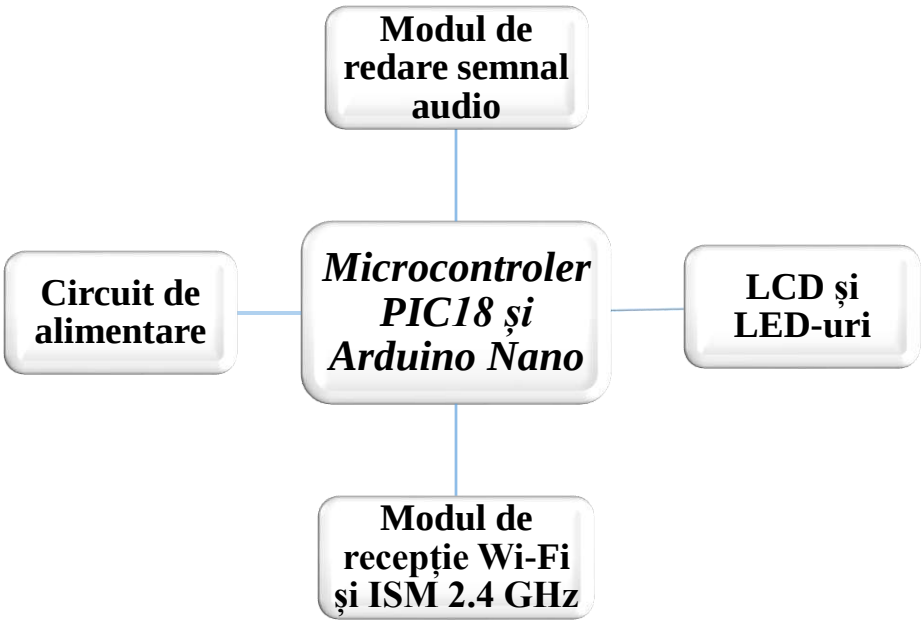


Figura II. 2: Schema bloc a circuitului de recepție

Practic, au fost realizate două circuite individuale – de transmisie și de recepție – ce dețin circuite de alimentare individuale bazate pe mai multe stagii diferite: 12V, 5V și 3.3V și microcontrolere de același tip pentru a putea exemplifica diferențele majore în materie de aplicabilitate.

Țin să menționez că pentru testarea pe microcontroler s-au luat în considerare și alte opțiuni precum Arduino Micro și Raspberry Pi.

În cadrul circuitului de transmisie vom avea ca intrări circuitul de alimentare, senzorul de MPU6050 și senzorii de flexie. Datele colectate prin intermediul acestora – practic, achiziția datelor gestului respectiv - vor fi procesate și transmise prin intermediul modulului de transmisie Wi-Fi și ISM 2.4 GHz.

În cadrul circuitului de recepție vom avea ca intrări circuitul de alimentare și modulul de recepție Wi-Fi și ISM 2.4 GHz. Datele colectate prin modulul de transmisie sunt procesate de microcontroler și sunt transmise spre modulul audio și spre LCD pentru a declanșa recunoașterea vocală și afișarea sub formă de text a gestului.

De asemenea, s-au reliefat și circuitele intermediare precum cel de Reset manual al microcontroler-ului, cel al oscilatorului de cuarț și cel al divizorului de tensiune realizat pentru senzorii de flexie din partea de transmisie.

Aspectele legate de circuit cuprinzând schemele electrice și PCB-urile pot fi regăsite detaliat în Anexa 2.

Descriere componente hardware

Pentru a putea descrie corect și complet componentele hardware, este nevoie de o clasificare a acestora pe blocuri individuale, așa cum sunt precizate în schemele anterioare:

a) Circuitul de alimentare
b) Senzori de flexie – folosiți pentru recunoașterea gesturilor
c) MPU-6050 – Modulul de giroscop și accelerometru cu 3 axe – folosit ca mod de recunoaștere a mișcării mâinii
d) Microcontroler – folosit ca unitate de recunoaștere a gesturilor
e) Transmițător și receptor RF
f) LCD – folosit ca mod de recunoaștere de tip gesturi-text
g) Modul de înregistrare-redare a semnalului audio – folosit ca mod de recunoaștere de tip gesturi-semnal audio

1. Circuitul de alimentare

Datele precizate în această parte sunt preluate din Lucrarea de Diplomă unde am abordat același tip de circuit.

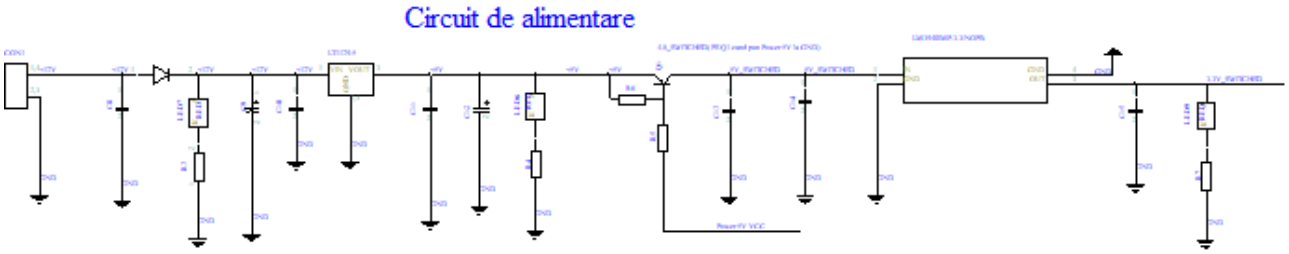


Figura II. 3: Schema circuitului de alimentare cu două regulatoare 12V-5V și 5V-3.3V

Pentru circuitul de alimentare am hotărât să folosesc două regulatoare de tensiune cascadate (unul de la 12V la 5V și unul din 5V în 3,3V), câteva LED-uri de tip SMD pentru semnalizarea tensiunilor, o diodă de tip Schottky și câteva condensatoare.

Probabil, la prima vizualizare a acestui circuit, se va observa numărul destul de mare de condensatoare de decuplare. Un astfel de condensator este utilizat pentru decuplarea unei părți a unui circuit de la o altă parte și, astfel, zgomotul cauzat de alte elemente de circuit este evitat prin condensator, reducând efectul pe care acesta îl are asupra restului circuitului.

Dispozitivele active ale sistemului (tranzistori, circuite integrate, etc) sunt conectate la sursele lor de alimentare prin conductori cu rezistență finită și inductanță. Dacă un curent al unui dispozitiv activ se schimbă, tensiunea de la sursa de alimentare la dispozitiv se va schimba destul de brusc datorită acestor impedanțe parazite. Dacă mai multe dispozitive active dețin o cale comună la sursa de alimentare, modificările curentului unui element pot provoca schimbări de tensiune extrem de mari ce vor afecta funcționarea întregului circuit. Un condensator de decuplare oferă o cale de bypass (evitare) pentru curenții tranzitorii, în loc să curgă prin impedanța comună.

Pentru a reduce impedanțele parazite, condensatoarele mici și mari sunt adesea plasate în paralel, poziționate în mod obișnuit lângă circuitele integrate individuale. Condensatorul stochează o cantitate mică de energie care poate compensa scăderea de tensiune a conductorilor de alimentare la condensator.

Un alt tip de decuplare este oprirea unei porțiuni a unui circuit pentru a nu fi afectată de comutarea care are loc într-o altă porțiune a circuitului. Comutarea în subcircuitul A poate cauza fluctuații în sursa de alimentare sau în alte linii electrice, dar nu se dorește ca subcircuitul B, care nu are nimic de-a face cu acea comutare, să fie afectat. Un condensator de decuplare poate decupla subcircuitele A și B astfel încât B să nu observe niciun efect al comutării.

Un condensator de decuplare a sarcinii tranzitorii este plasat cât mai aproape posibil de dispozitivul care necesită semnalul decuplat. Acest lucru minimizează cantitatea de inductanță de linie și rezistența în serie dintre condensatorul de decuplare și dispozitiv. Cu cât conductorul este mai lung între condensator și dispozitiv, cu atât inductanța este mai mare. Deoarece condensatoarele diferă în funcție de caracteristicile de înaltă frecvență (condensatoarele cu bune proprietăți de înaltă frecvență sunt adesea cele cu capacitate mică), decuplarea implică adesea utilizarea unei combinații de condensatori. De exemplu, în circuitul anterior, s-a realizat o grupare de circa 100 nF formată atât din condensatori electrolitici, cât și din cei cu tantal.

Practic, s-a luat în considerare ideea că orice poate genera curenți tranzitorii. Când acești curenți tranzitorii sunt extrași direct de la sursa de alimentare, tensiunile tranzitorii sunt create ca urmare a impedanței sursei de alimentare, iar acest efect este din ce în ce mai problematic atunci când o componentă trebuie să conducă o sarcină cu rezistență scăzută sau cu capacitate mare deoarece componentele cu rezistență scăzută creează tranziții cu amplitudini mari, iar cele cu capacitate ridicată pot duce la vibrații sau chiar oscilații puternice în linia de alimentare. Rezultatul final poate fi orice, de la performanța circuitului până la defectarea sistemului. Astfel, erau necesare condensatoare de bypass. Din conectorul de alimentare avem 12V, urmând ca tensiunea să fie semnalizată prin LED-ul SMD și să treacă prin dioda de tip Schottky spre regulatoarele în configurație cascadă.

Menționez că am folosit și o configurație cu regulatoare în paralel, dar era mult mai eficientă aceasta (avea randament mai mare).

Primul regulator de tensiune ales este denumit LT1129, iar modelarea acestuia în circuit s-a realizat ținând cont de faptul că acesta necesită și anumite condensatoare de tip reacție pentru a funcționa în parametri nominali.

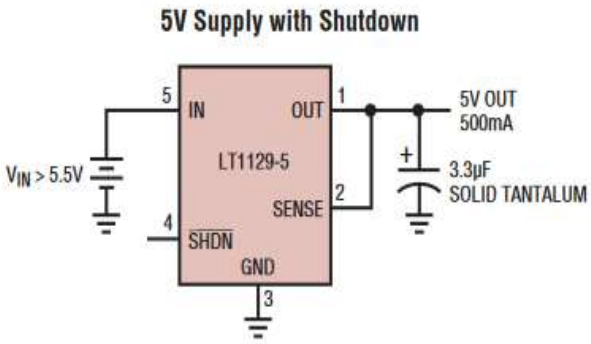


Figura II. 4: Modul de funcționare al primului regulator de tensiune [20]

Putem observa aici faptul că nu am folosit un regulator în comutație, ci un tranzistor bipolar deoarece am realizat calculele consumului de curent și am văzut că disipația termică este sub specificația maximă dată de producătorul regulatorului pe care aș fi vrut să îl folosesc. Un alt aspect important este acela că nu voiam să perturb accelerometrul deoarece este destul de sensibil.

Al doilea regulator de tensiune (de la 5V spre 3,3 V) este denumit LM3940 și prezintă, de asemenea o schemă specifică pentru funcționare nominală.

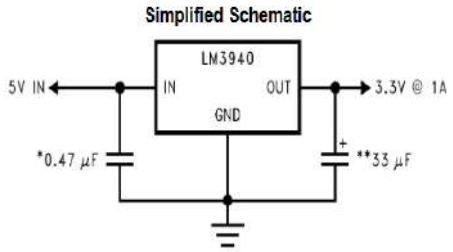


Figura II. 5: Modul de funcționare pentru regulatorul de tensiune LM3940 [22]

Am avut nevoie și de acest tip de regulator de tensiune deoarece modulul MPU6050 avea alimentare la 3,3V și nu avea regulator integrat.

2. Senzori de flexie – folosiți pentru recunoașterea gesturilor

Unitatea de detecție a gesturilor este partea proiectului în care se realizează detecția formei mâinii și procesarea gesturilor. Nucleul acesteia este circuitul de pe mănușă unde există o placă ce primește toate ieșirile de la senzorii de flexie și senzorul de orientare. Senzorii de flexie se bazează pe elemente de carbon rezistive și funcționează ca o rezistență variabilă într-un circuit. Pe măsură ce dispozitivul este îndoit, senzorul produce la ieșire o rezistență corespunzătoare unghiului la care este îndoit. Caracterizând această corelație, putem utiliza senzori de flexie pentru a determina modul în care degetele unei persoane se mișcă în timpul mișcărilor diferite ale mâinii.

Pentru a integra senzorii de flexie în proiectarea sistemului, trebuie caracterizat comportamentul de rezistență al fiecărui senzor pentru o relație între rezistența senzorului și unghiul de îndoire. Astfel, senzorii au fost testați prin legarea unui pinilor la placa Android Nano printr-un divizor rezistiv și la masă. Ulterior, s-au realizat achiziții de date continue prin îndoirea senzorilor în diferite părți, simulând îndoirea unui deget. Se menționează că senzorii au fost îndoiți în ambele direcții posibile -> pe direcția înainte de la 0° la 90° și pe direcția înapoi de la 90° la 0°. Astfel, s-a putut simula atât contracția, cât și extensia degetelor.

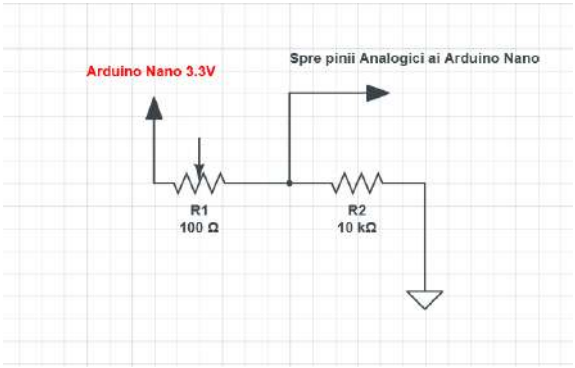


Figura II. 6: Schema electrică a senzorilor de flexie și modul de aranjare în circuit

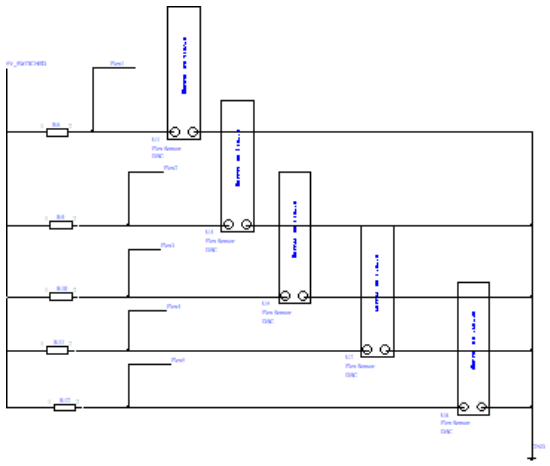


Figura II. 7: Schema electrică a senzorilor de flexie

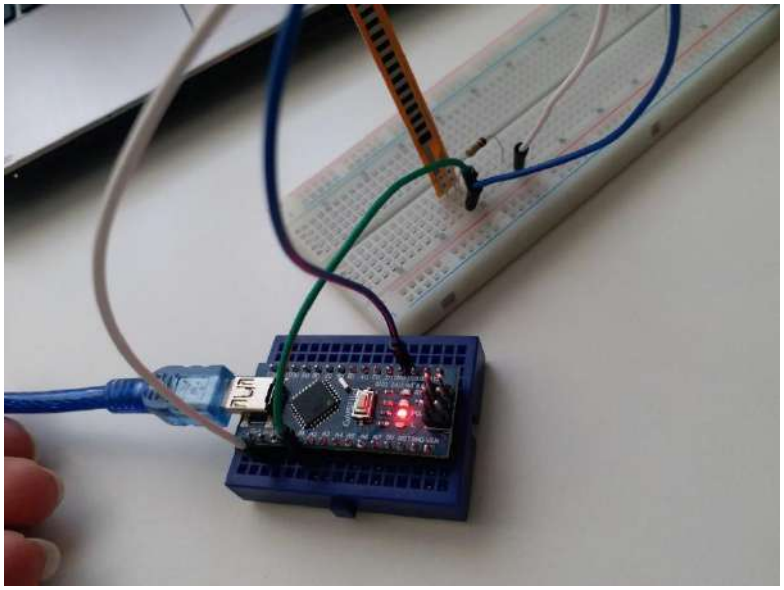


Figura II. 8: Mod de efectuare a testelor senzorilor de flexie

Figura II. 9: Mod de efectuare a testelor senzorilor de flexie

3. MPU-6050

O parte din datele precizate în această parte sunt preluate din Lucrarea de Diplomă unde am abordat același tip de circuit.

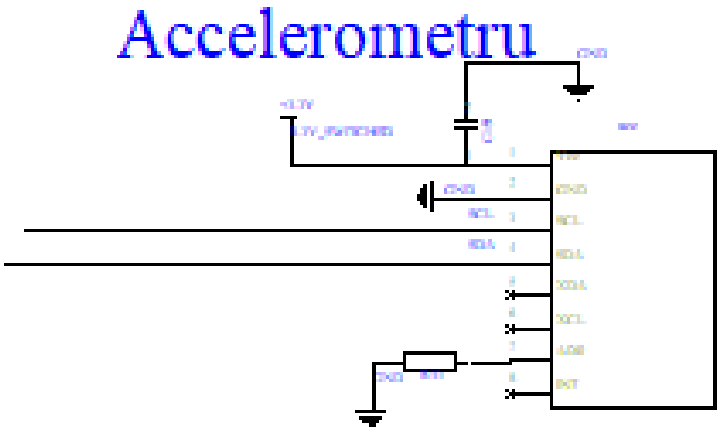


Figura II. 10: Schema electrică a senzorului de MPU6050

Deoarece aveam nevoie de un senzor de tip accelerometru ale cărui date să le prelucrez cu ajutorul microcontrolerului, am ales modulul integrat MPU6050 ce prezintă atât accelerometru, cât și giroscop și un senzor de temperatură. Dat fiind faptul că datele din accelerometru sunt de tip aleator, fiind date digitale, am nevoie de o filtrare. În urma a mai multe încercări, am decis să folosesc un filtru de tip Kalman care să asigure o continuitate a datelor. Am folosit mai multe seturi de rezistențe de pull-up deoarece voiam să verific funcționalitatea modulului la diferite frecvențe. Astfel, în urma testelor, am hotărât să le îndepărtez de pe placă pe cele ce nu lucrau la frecvența dorită deoarece, dacă le lăsam în configurația finală pe toate, aveam o viteză de trei ori mai mică, ei fiind în paralel.

MPU6050 are ca avantaje faptul că oferă atât o măsurare inerțială a vitezei și poziției, cât și ca senzor de înclinare sau orientare sau ca senzor de vibrație sau de impact. Un exemplu de date prezentate de MPU6050 este dat în Anexa 1.

Un alt aspect important care a dus la alegerea sa este și folosirea de către acesta a magistralei de comunicație I2C.

Echipamentul dispune și de funcționalități speciale cum ar fi detecția prezenței sau absenței mișcării, prin compararea accelerației citite cu un prag setat de utilizator, detecția loviturilor simple de tip „impact” precum și căderi în gol. Toate acestea pot fi configurate cu ajutorul pinilor de întrerupere de care dispune senzorul.

Pini acestui dispozitiv folosiți în proiect sunt:

- GND – conexiune la masă;
- VCC – conexiune la tensiunea de alimentare (3.3V);
- SDA – date seriale - conexiune la pin analog microcontroler;
- SCL – ceas de comunicație serială - conexiunea la pin analog microcontroler.

Modalitatea de realizare a conexiunilor modulului cu microcontroler-ul este dată de cele două semnale principale SDA și SCL.

4. Microcontroler

Microcontroler-ul este partea principală a proiectului ce preia datele de intrare (din MPU6050 și senzorii de flexie sau din receptorul de tip Wi-Fi sau ISM 2.4GHz) și le prelucrează pentru a avea, la ieșire, un set de date ce urmează a fi transmise spre partea de ieșire (formată din transmițător sau din modulul audio și LCD).

Astfel, structura exterioară a pinilor e dată de interconectarea între echipamente, iar structura interioară a pinilor arată modul de ordonare a acestora în caracteristicile prezente în fișa tehnică a microcontroler-ului. Alături de structura prezentată a microcontroler-ului, mai avem și structura circuitului de resetare, a cristalului de cuarț, dar și structura conectorului ICSP prin care se încarcă programul.

În ceea ce privește partea de procesare ce trebuie realizată în cadrul unui astfel de proiect, se consideră alegerea unui microcontroler potrivit. Aspectele de care ar trebui să ținem cont când alegem un microcontroler sunt dimensiunea redusă, costul mic, consumul redus de energie și memoria de program.

Date fiind modurile de realizare a circuitului, s-a decis testarea acestuia cu diverse tipuri de microcontroler: Arduino Nano [28], Arduino Micro, Raspberry Pi.

Microcontrolerul de tip PIC18 ce a fost ales are un preț mic și e ușor de utilizat datorită arhitecturii RISC. Deține un număr minim de 33 instrucțiuni cu un grad mare de ortogonalitate (reduce timpul necesar de realizare a unei operații), din care majoritatea se execută într-un singur ciclu de ceas, excepție făcând cele de salt ce necesită două cicluri.

Microcontrolerul are magistrale diferite pentru date și pentru instrucțiuni, iar unitatea centrală are 8 biți și realizează funcții aritmetice și booleene (adunare, înmulțire, operații logice și deplasări la stânga sau la dreapta).

Setul de registre localizat în memoria RAM conține registre cu funcții speciale și registre de uz general.

Sistemul de Reset a fost conectat la un pin MCLR, și, atunci când este apăsat, dă un "0" logic. Acest lucru are ca efect resetarea microcontrolerului, ștergerea memoriei RAM și pornirea din nou a programului. El este deosebit de util atunci când programați microcontrolerul utilizând o tehnică avansată numită Boot-Loader sau când microcontroler-ul rulează o buclă infinită.

Modelul utilizat în cadrul acestui proiect are la bază un microcontroler de la Microchip respectiv un PIC18. Tensiunea de alimentare este de 5V obținută din circuitul de alimentare, frecvența de funcționare a microcontroler-ului este de la 31kHz la 16 MHz.

Protocolul pentru transferul de date poate avea loc dacă magistrala de date nu este ocupată, iar SDA și SCL au ambele valoarea 1. Un START este determinat de frontul SDA din 1 în 0, iar un front SDA din 0 în 1 duce la apariția unei condiții de STOP. În acest interval de START-STOP, datele trebuie să urmeze tranziții succesive din 0 în 1 și din 1 în 0 pe durata unui impuls de tact SCL. În cadrul proiectului, o transmisie începe cu secvența de START și adresa slave-ului urmată de bitul 0 dacă se vrea scrierea unei instrucțiuni pentru slave sau 1 dacă se dorește citirea. Dacă, în continuare, se dorește prelungirea comunicării, se transmite o altă secvență de START, altfel, se transmite secvența de STOP.

În proiect, s-a folosit o interfață I2C Master cu un singur master (microcontroler) și mai multe dispozitive de tip slave(senzorul MPU6050 și senzorii de flexie). Pentru a putea accesa datele de la senzor, se activează chip-ul care începe să înregistreze valori în regiștri. Prin protocolul I2C,

microcontrolerul primește date în timp real și nu mai există interpretări și transformări ca în cazul senzorilor analogici (ce necesită un ADC).

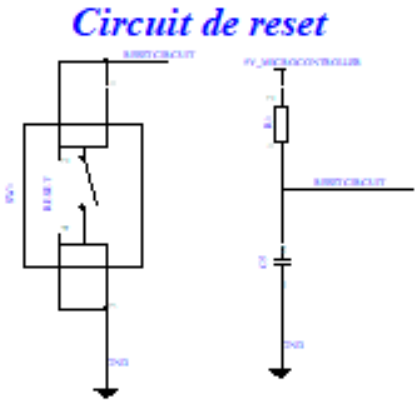


Figura II. 11: Schema electrică a circuitului de resetare

1. pentru placa ce conține partea de recepție

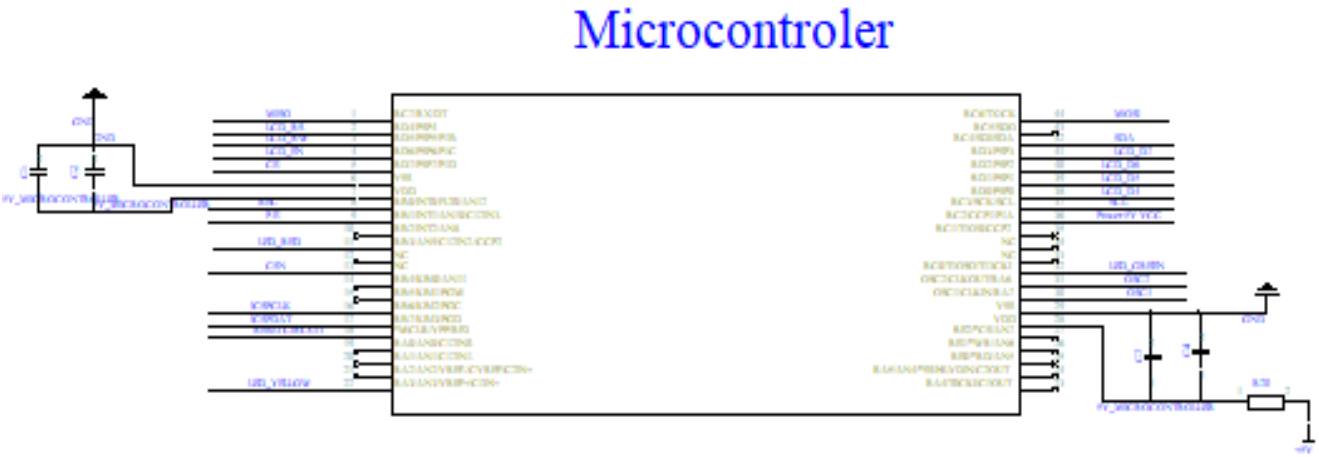


Figura II. 12: Schemă electrică a microcontroler-ului folosit în partea de recepție

2. pentru placa de transmițător

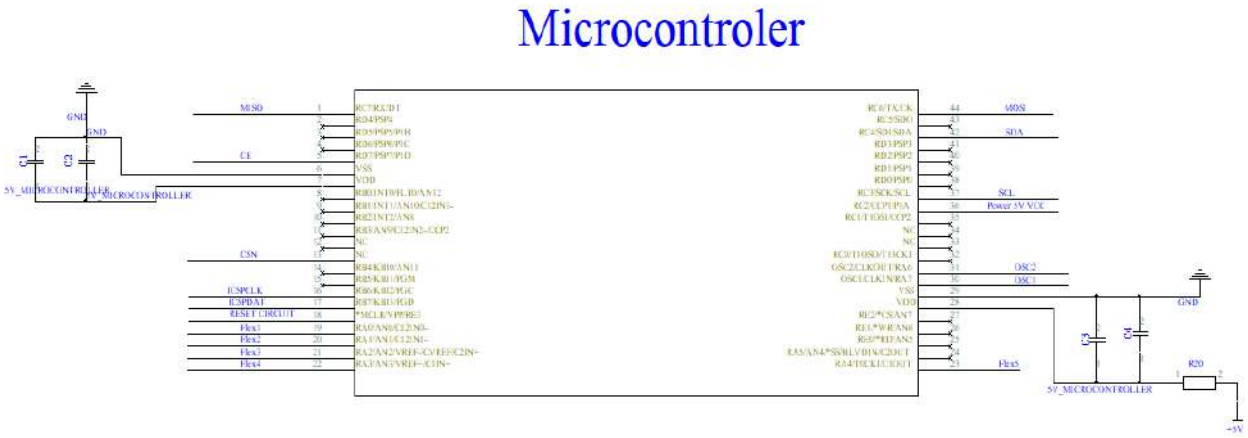


Figura II. 13: Schemă electrică a microcontroler-ului folosit în partea de transmisie

5. Transmițător și receptor RF

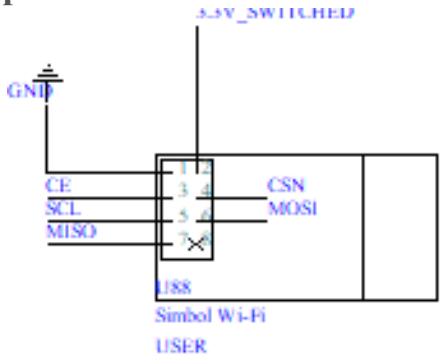


Figura II. 14: Schema electrică a modulului ISM 2.4 GHz

Modulul de transmisie și recepție este un dispozitiv de dimensiuni mici ce trebuie să asigure două părți importante în domeniile sale de aplicabilitate: funcționarea pe o anumită distanță – ce trebuie să fie cât mai mare pentru servicii calitative – și transmiterea unei cantități cât mai mare de informații.

Decizia cu privire la utilizarea acestui modul a fost luată deoarece proiectul nu necesită cerințe de comunicații la distanță mare și nici o putere de transmisie ridicată.

Chiar dacă tensiunea de funcționare este de 3,3 V, majoritatea pinilor sunt toleranți la 5 V și, prin urmare, pot fi interfațați direct cu microcontrolerele de 5 V. În cadrul proiectului de testare cu Arduino Nano, modulul a fost, totuși, interfațat cu pinul corespunzător tensiunii de alimentare de 3.3 V pentru siguranța modulului.

Modulul folosește banda de 2,4 GHz și poate funcționa cu viteze de transfer de la 250 kbps până la 2 Mbps. Dacă este utilizat în spațiu deschis și cu o viteză mai mică de transfer, domeniul său poate ajunge până la cincizeci de metri. De asemenea, acest modul poate comunica cu încă 6 dispozitive de același tip simultan, aspect favorabil în aplicații ce folosesc domeniul IoT.

Din testele efectuate asupra dispozitivului, pinii de activare și de menținere a comunicației SPI pot fi conectați cu orice pin digital al microcontrolerului Arduino Nano. Aceștia sunt folosiți pentru setarea modulului în regim de așteptare sau activ, precum și pentru comutarea între modul de transmisie sau de comandă.

6. LCD

LCD-ul utilizat în cadrul proiectului este de tip 16x2 este un dispozitiv cu 16 pini care are 8 pini de date (DB0-DB7) și 3 pini de control (RS, RW, EN). Restul de 5 pini sunt pentru alimentarea și iluminarea ecranului LCD.

Pinii de control ajută la configurarea ecranului LCD în modul de comandă sau în modul de date sau în modul de citire sau de scriere.

Ecranul LCD 20x2 poate fi utilizat în moduri pe 4 biți sau pe 8 biți, în funcție de cerințele aplicației. Pentru a le folosi, trebuie să trimitem anumite comenzi pe ecranul LCD în modul de comandă și odată ce ecranul LCD este configurat în funcție de nevoile noastre, putem trimite datele necesare în modul de date.

Configurarea pinilor :

- Pinul 3 – VEE / VO - utilizat pentru ajustarea contrastului afișajului. Tensiunea pe acest pin definește contrastul pe afișaj, coboară tensiunea, crește contrastul. Putem conecta un potențiometru pentru reglarea contrastului sau, pentru a obține un contrast maxim, se conectează acest pin la masă.
- Pinul 4 -RS – selectorul de registre ce poate avea valorile:
    - RS = 0: Datele pe pinii D0 - D7 sunt considerate o comandă.
    - RS = 1: Datele de pe pinii D0 până la D7 sunt considerate date de afișat
- Pinul 5 – RW – pinul de scriere / citire ce poate deține valorile:
    - RW = 0: Se vor scrie datele pe ecranul LCD
    - RW = 1: Se vor citi datele de pe ecranul LCD
- Pinul 6 – EN - Pinul de activare - este folosit pentru a bloca datele prezente pe pinii de date D0 până la D7.
- Pinii 7:14 – pini pentru datele D0-D7- sunt utilizați pentru a trimite date / comenzi
- Pinii 15:16 - LED + / A și LED - / K - modulul are un LED cu lumină de fundal.

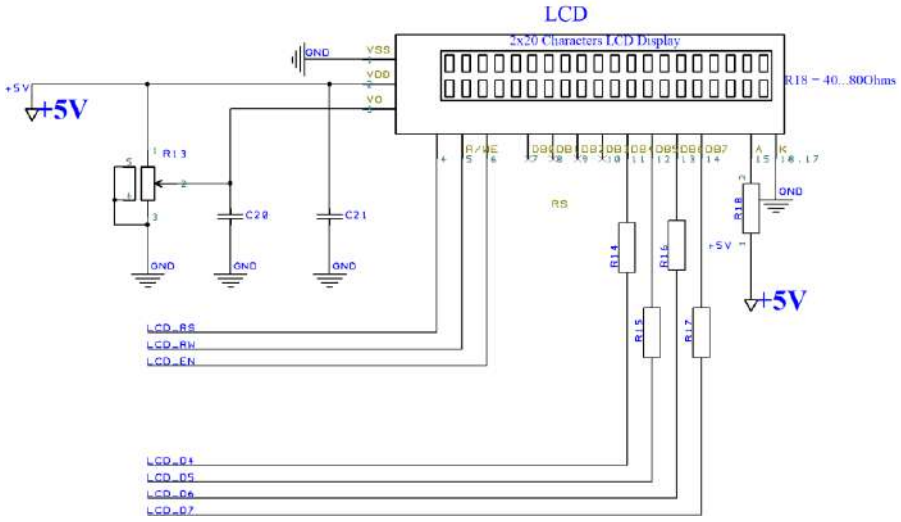


Figura II. 15: Schema în DesignSpark a LCD-ului de tip 16x2

7. Modul de înregistrare-redare a semnalului audio

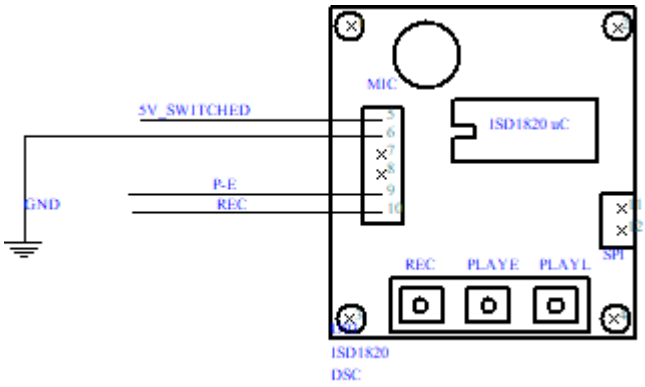


Figura II. 16: Schema electrică a modulului de înregistrare-redare a semnalului audio

Modulul de semnal audio oferă o metodă de înregistrare și redare a semnalului de o calitate bună, pe un singur chip, cu durată de la 8 la 20 secunde. Acesta este format dintr-un oscilator cu control extern, un preamplificator pentru microfon, control automat de câștig și un amplificator pentru difuzor. Înregistrările efectuate sunt stocate în memoria nevolatilă în forma lor naturală, oferind o redare calitativă.

Durata de înregistrare și redare este controlabilă prin valoarea rezistenței externe ce setează rata de eșantionare.

ROSC	Durata	Rata de eșantionare	Lățimea de bandă
80 KΩ	8 s	8.0 KHz	3.4 KHz
100 KΩ	10 s	6.4 KHz	2.6 KHz
120 KΩ	12 s	5.3 KHz	2.3 KHz
160 KΩ	16 s	4.0 KHz	1.7 KHz
200 KΩ	20 s	3.2 KHz	1.3 KHz

Tabelul II. 1: Legătura între rezistențele externe și rata de eșantionare a modulului de înregistrare-redare a semnalului audio[22]

Pinii utilizați în cadrul acestui proiect sunt REC și P-E, cei de alimentare și masă și cei de SP+ și SP-.

1. Pinul de REC este utilizat la înregistrarea mesajelor și este activ în "unu" logic. Dacă se iau în considerare prioritățile asupra semnalelor de redare, semnalul de REC are prioritate mai mare. Astfel, dacă o redare a sunetului este în curs, iar butonul de REC este apăsat, redarea încetează imediat și este pornită starea de înregistrare.
2. Pinul de PLAYE este pinul de redare ce este activat pe front – când este detectată o tranziție începe un ciclu de redare. Aceasta continuă până la finalul mesajului sau până se detectează o întrerupere.
3. Pinul de PLAYL este pinul de redare ce este activat pe palier – când se realizează o trecere din starea logică "zero" în starea logică "unu" este inițiat un ciclu de redare. Acesta are un comportament identic cu pinul prezentat anterior.
4. Pinii SP+ și SP- asigură o acțiune direct pentru difuzoare. Având polaritate opusă, acești pini asigura o îmbunătățire a puterii de până la patru ori și nici nu mai este necesar un condensator de cuplare a difuzoarelor adițional (o conexiune cu un singur capăt necesită condensatoare de cuplaj alternativ între pin și difuzor).

Acest modul este folosit în proiect în forma următoare:

1. Se înregistrează mesajele aferente gesturilor detectate prin butonul REC - se va scrie astfel o parte din memoria dispozitivului
2. Se verifică integritatea acestora prin redare – PLAYL și PLAYE
3. Se verifică dacă spațiul alocat mesajelor este plin - dacă nu este plin, se reia înregistrarea unui nou mesaj, iar dacă este plin se verifică o realocare

Principala problemă a acestui modul este că volumul de redare audio este foarte scăzut. Pentru a îmbunătăți acest aspect, este nevoie de un amplificator înainte de difuzor sau este necesar ca, de la intrarea AGC la pinul de ieșire a LED-ului să fie adăugată o rezistență de 47K ohm. Acest

aspect asigură că circuitul AGC păstrează un nivel mai ridicat în timpul înregistrării și, prin urmare, creșterea volumului de redare.

8. Sistem de LED-uri

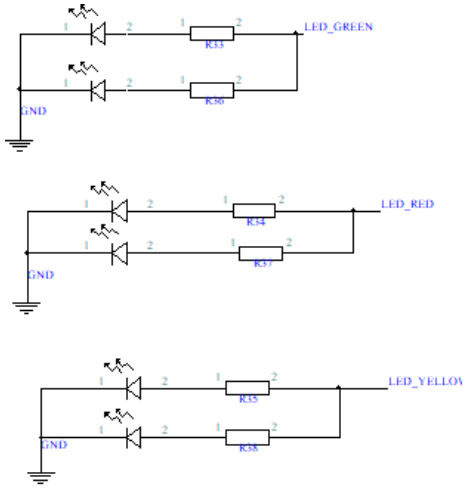


Figura II. 17: Sistemul de LED-uri

Sistemul de LED-uri prezent în proiect este dat de cele 6 LED-uri de diferite culori (verde, galben și roșu) ce formează sistemul de verificare a tensiunii al PCB-ului.

LED-urile prezentate sunt de tip THT și au asigurată legătura cu microcontroler-ul prin pinii acestuia la care s-au adăugat rezistoare de valoarea 270 ohm (Valoarea rezultată din calcule).

Diagramă software

1. Partea de detecție a gesturilor

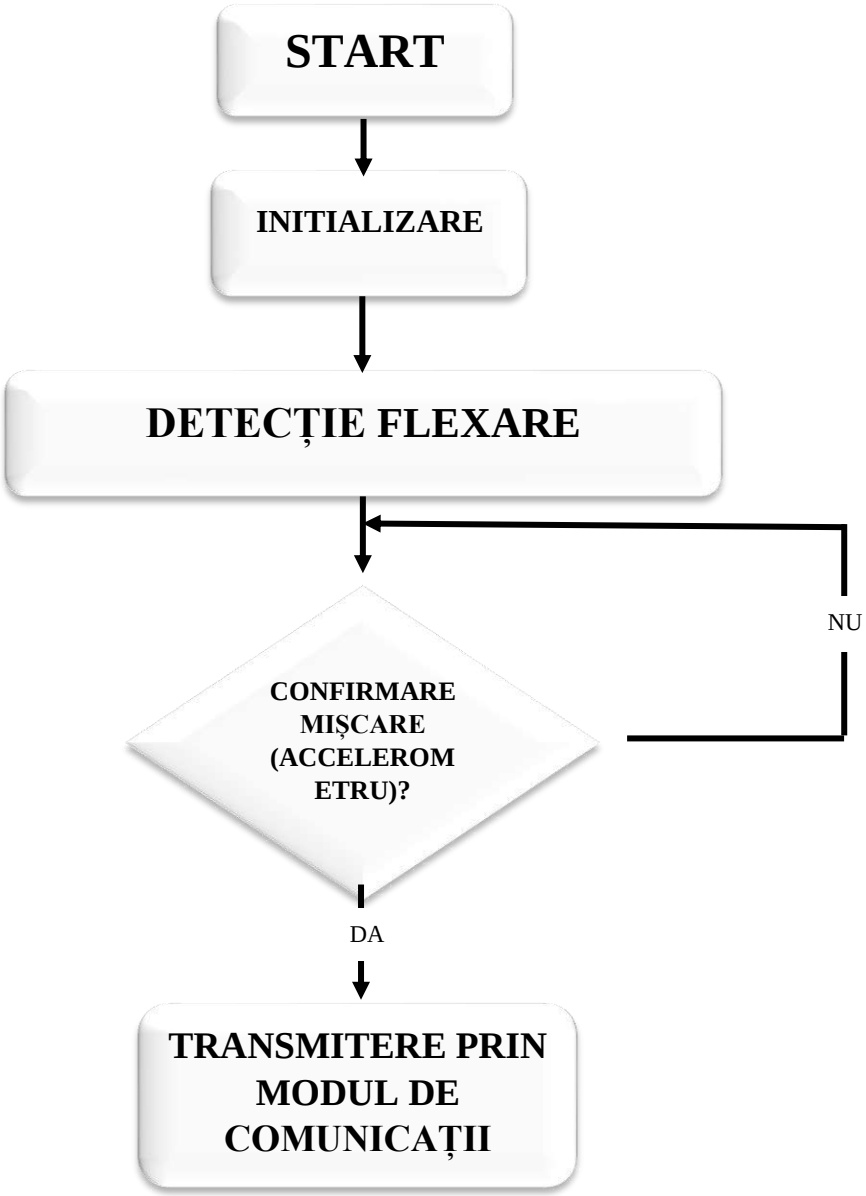


Figura II. 18: Diagramă software pentru placa de transmisie

Senzorii de flexie au fost testați pentru liniaritate și sensibilitate, apoi au fost pe mănușă. Un ADC a fost implementat pentru a introduce valorile analogice ale senzorilor de flexie la microcontroler, astfel încât să poată fi procesate în mod eficient. MPU6050 este folosit pentru a recunoaște cuvintele și/sau literele care au o mișcare a degetelor și o înclinare a mâinii.

Primul pas în realizarea părții software este inițializarea tuturor componentelor sistemului, specificând numele pinilor ce vor fi folosiți, dacă vor fi folosiți ca intrare sau ca ieșire în cadrul circuitului și toate variabilele necesare programului. De asemenea, pentru microcontrolerele de tip PIC, programul trebuie să conțină și inițializarea registrelor sistemului.

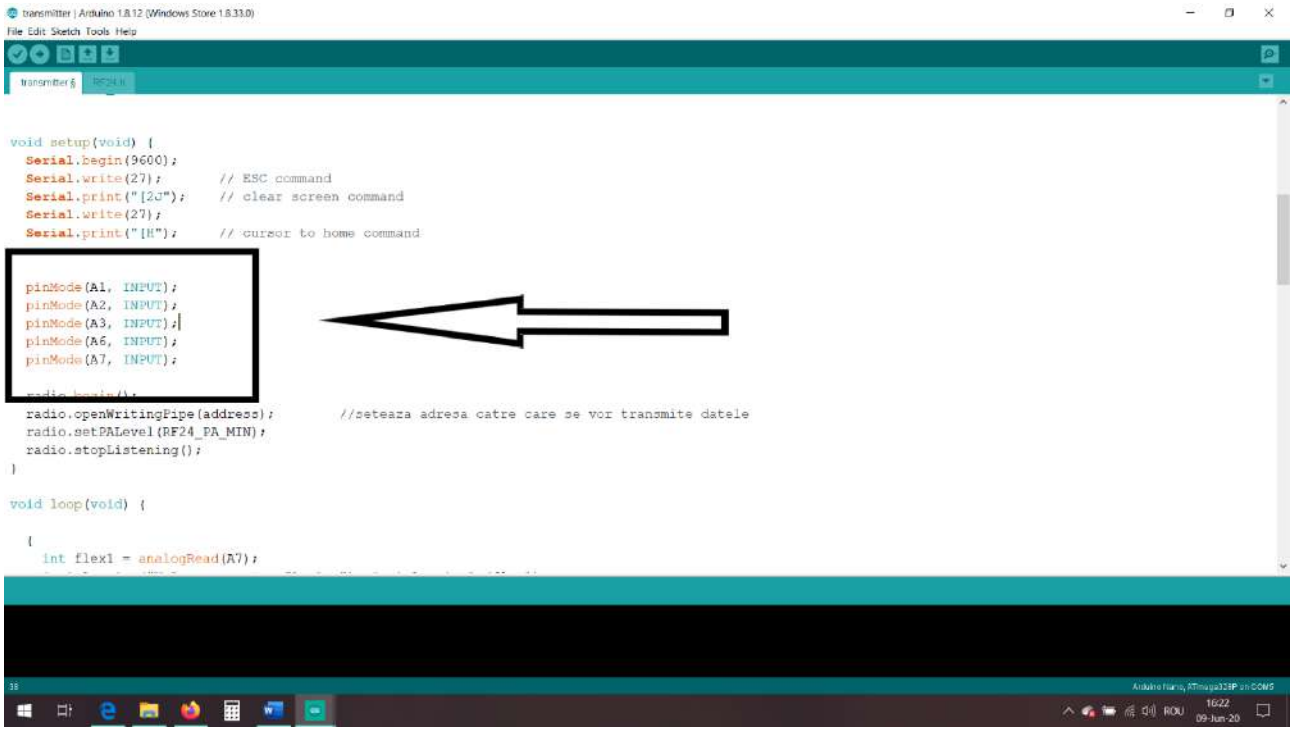


Figura II. 19: Setarea pinilor ca intrări în circuit

În cadrul circuitului de detecție a gesturilor – cel care conține și transmițătorul RF – au fost realizate citiri ale valorilor senzorilor de flexie și al modulului MPU6050 prin implementarea diverselor funcții de achiziție și calibrare. Aceste funcții vor afișa prin Serial Monitor diverse rezistențe și unghiuri de flexare. Următorul pas a fost realizat prin setarea unei valori de prag, în urma căreia se va realiza transmisia.

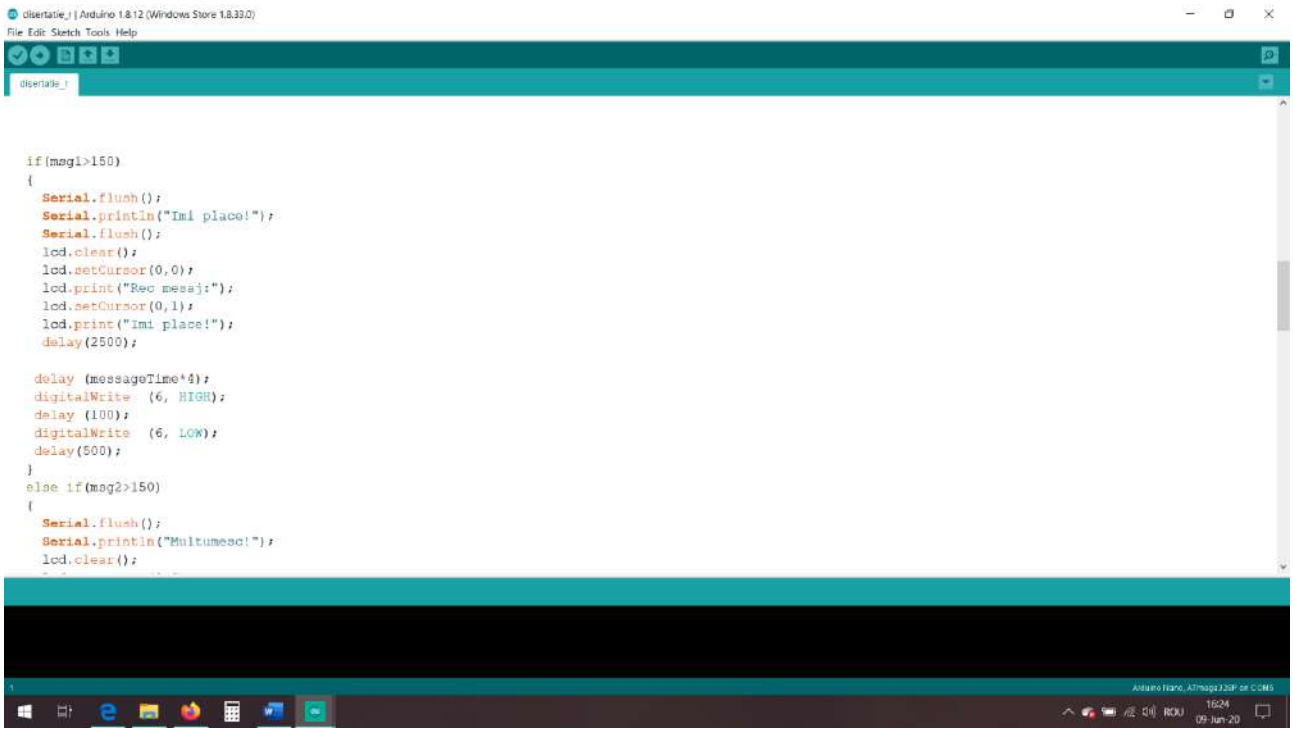


Figura II. 20: Realizarea rutinei pentru depășirea valorilor de prag

În modul de calibrare, se înregistrează valorile minimă și maximă pe care le poate produce fiecare gest. În acest proces, utilizatorul trebuie să îndoaie sau să îndrepte toate degetele sub diverse unghiuri, precum și să mute mâna în orice direcție.

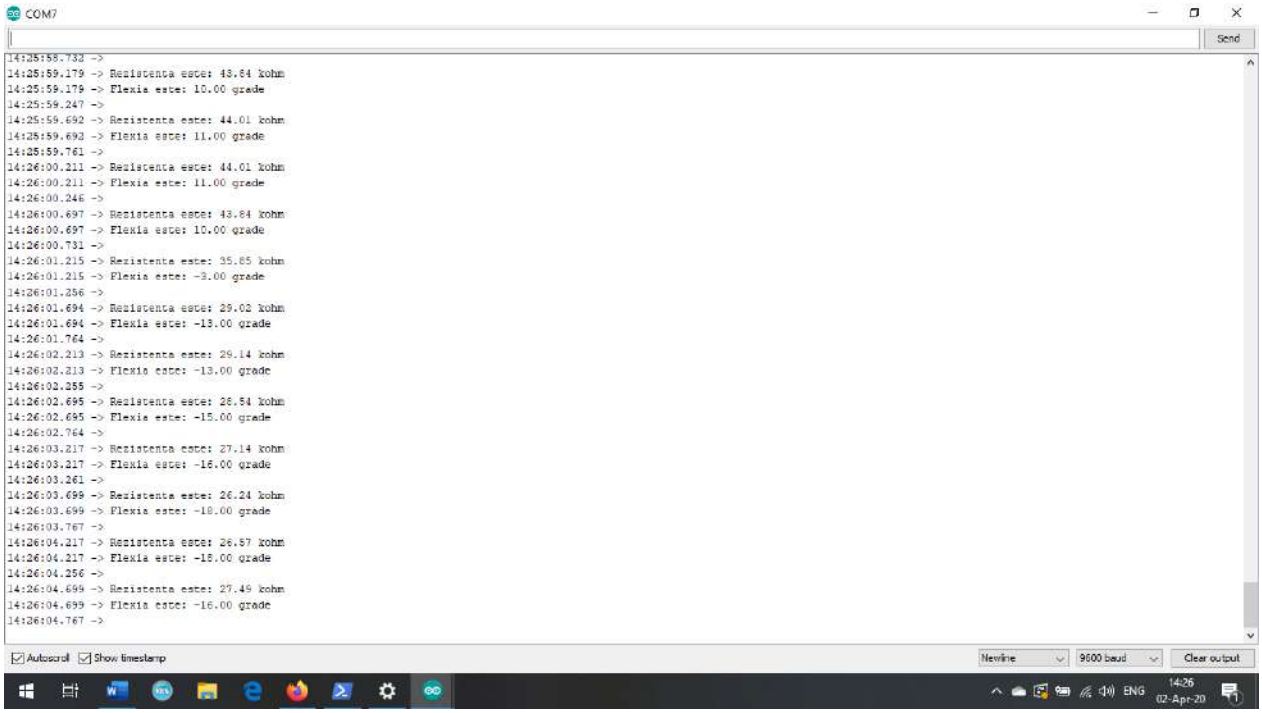


Figura II. 21: Valori regăsite în modul de calibrare a senzorilor de flexie

În paralel, se va realiza achiziția senzorilor din cadrul modulului MPU6050 ce vor returna valori pe trei axe pentru accelerație și valorile giroscopice.

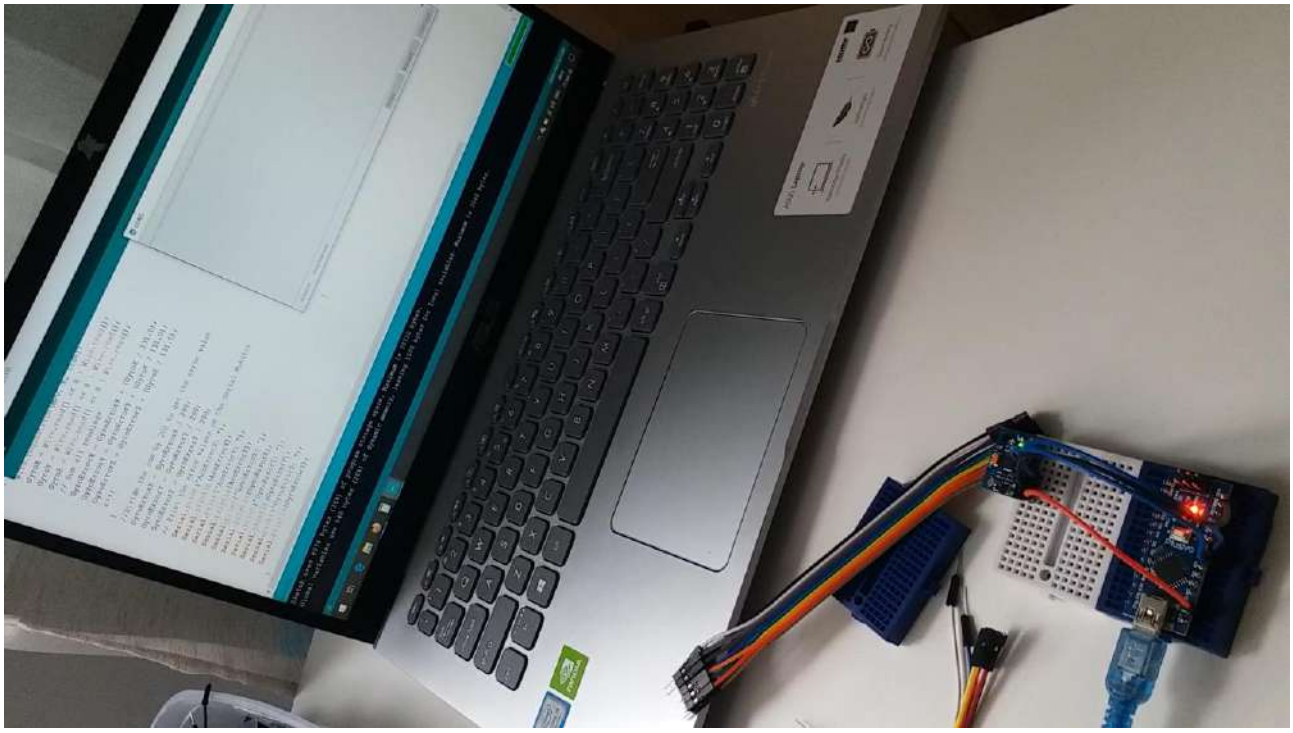


Figura II. 22: Valori regăsite în modul de calibrare a senzorului MPU6050 – demo1

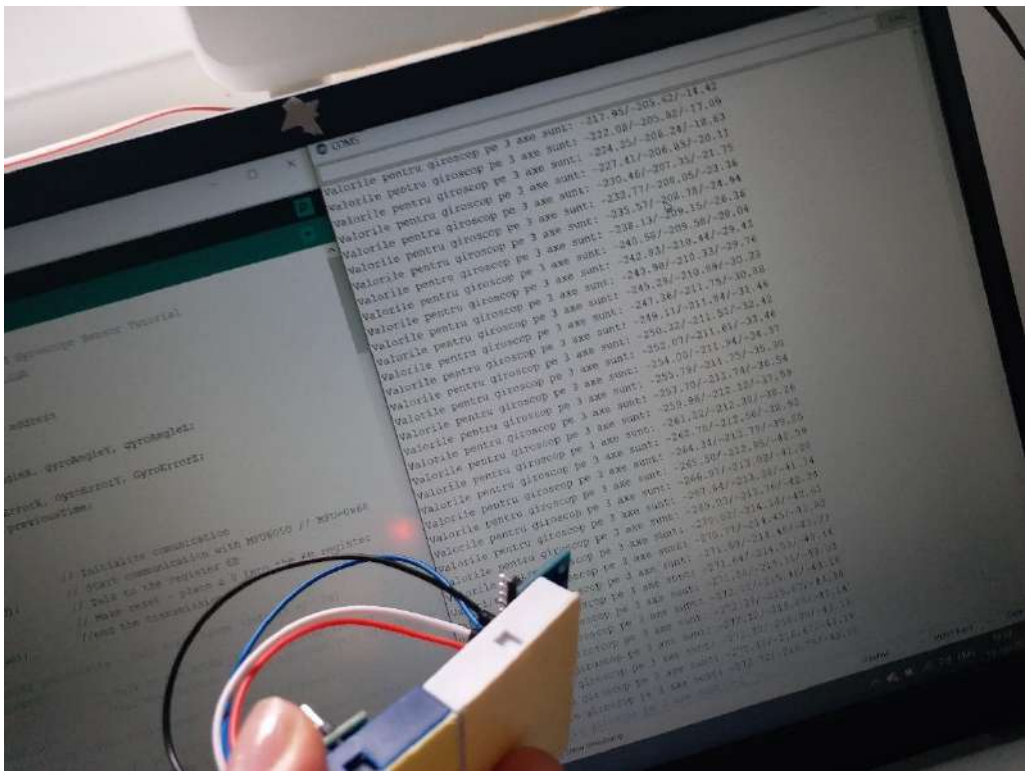


Figura II. 23: Valori regăsite în modul de calibrare a senzorului MPU6050 – demo2

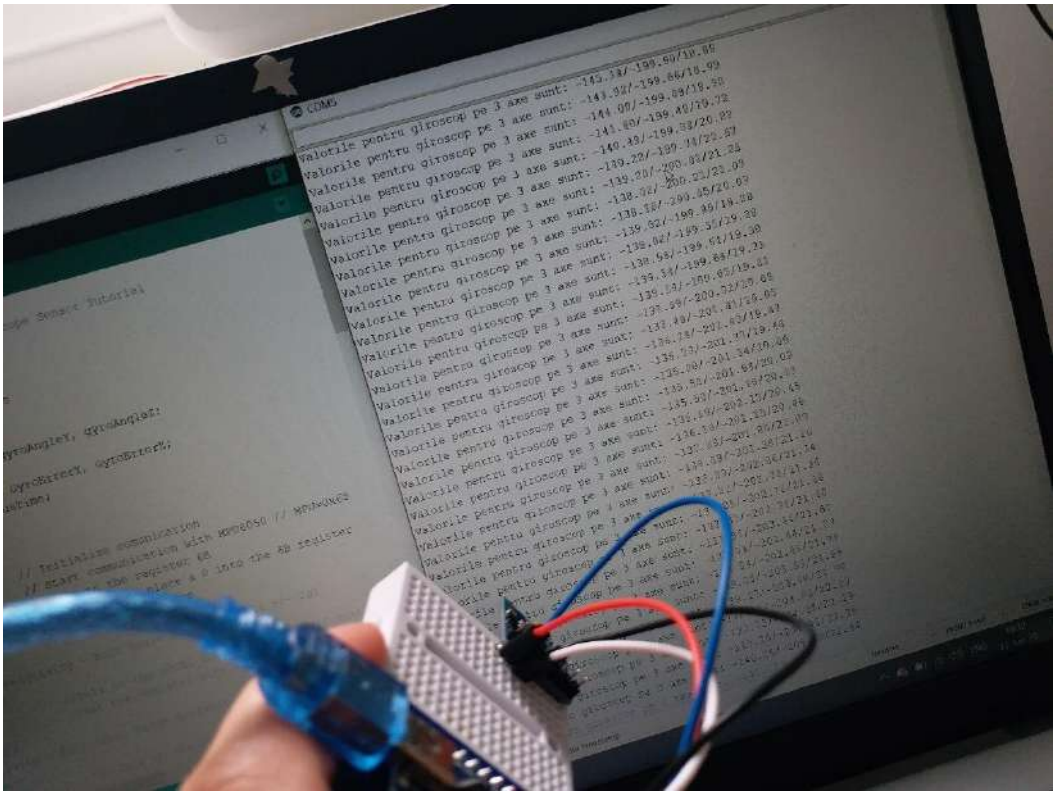


Figura II. 24: Valori regăsite în modul de calibrare a senzorului MPU6050 – demo3

Dacă valoarea testată pentru prag este depășită, se inițiază transmisia prin intermediul modulului nRF24L01 astfel: sunt setați pinii CSN și CE, se setează valoarea adresei de comunicație între cele două module și se inițializează modulul prin radio.openWritingPipe().

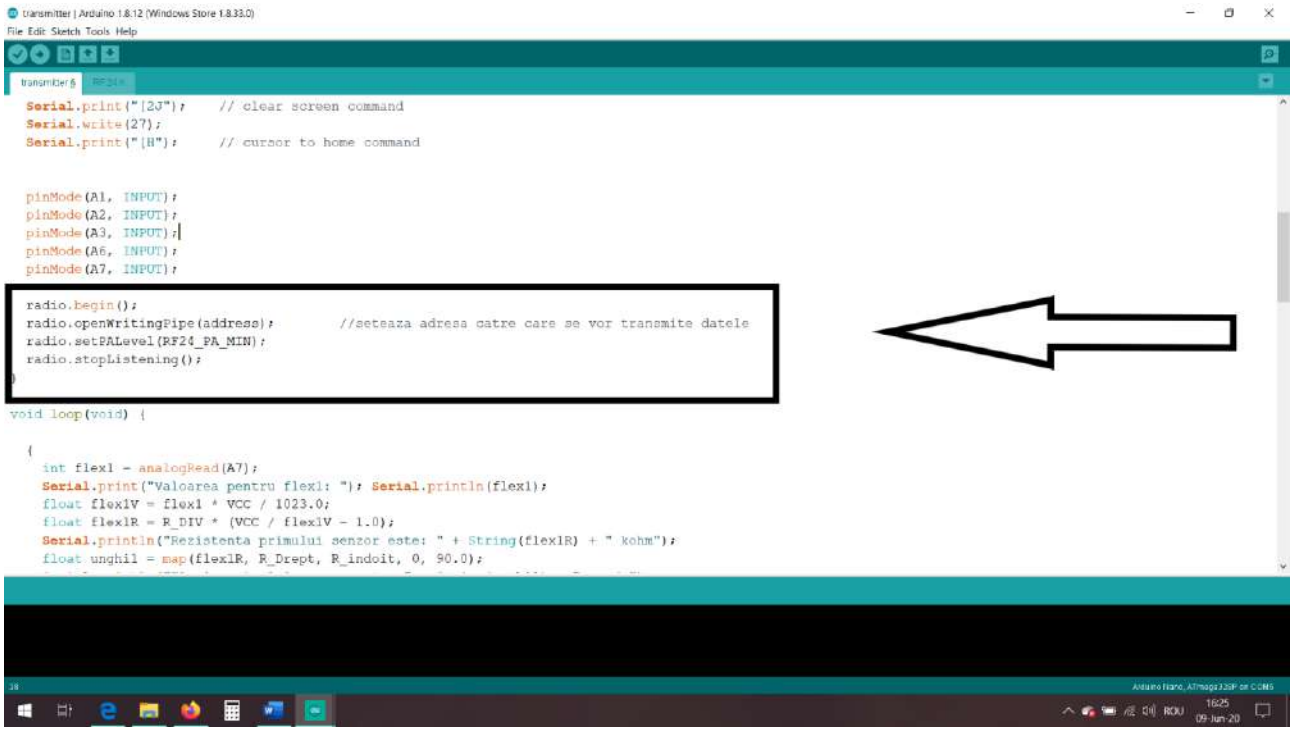


Figura II. 25: Funcții specifice modulului radio în etapa de emisie

Mesajul generat de prag, va fi transmis sub formă de text.

Figura II. 26: Afișaj sub formă de text a mesajului recepționat

2. Partea de translație a gesturilor recepționate

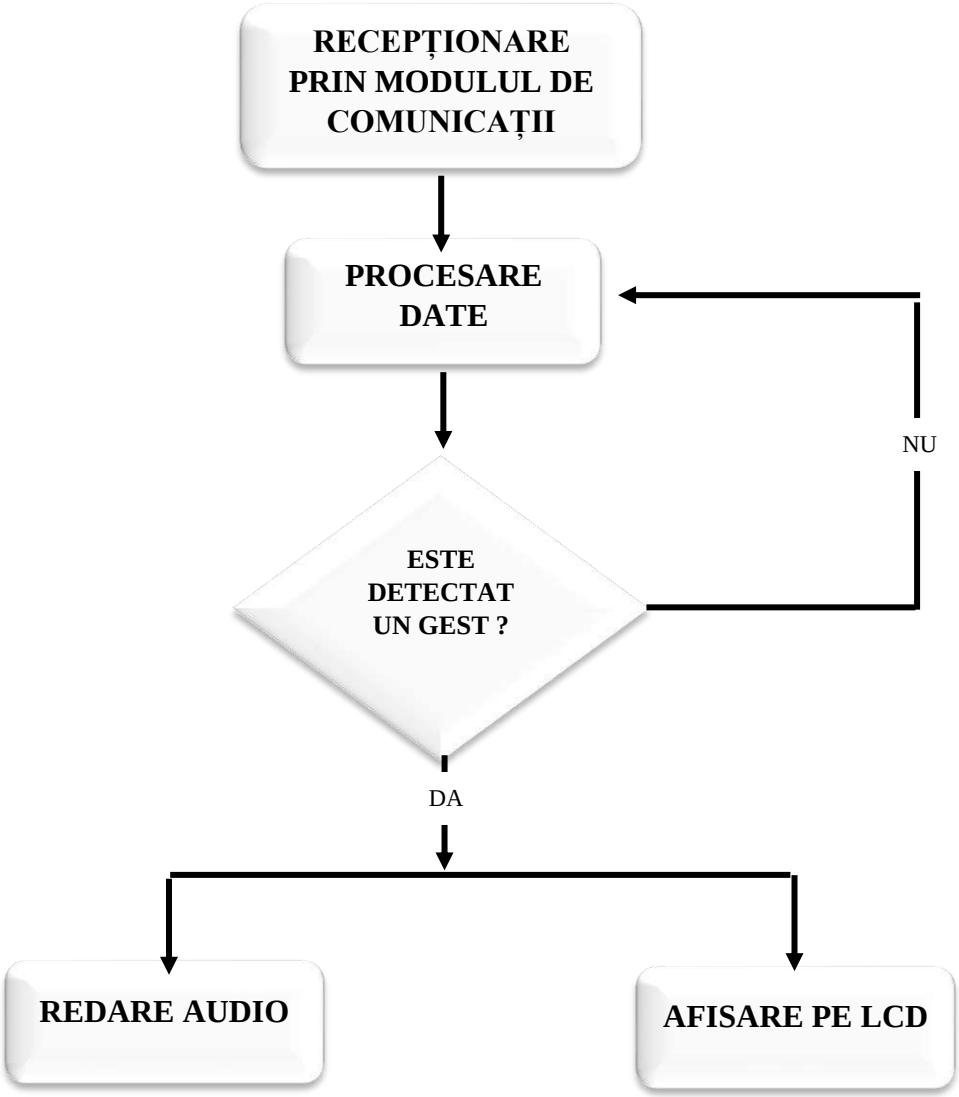


Figura II. 27: Diagramă software pentru placa de recepție

Am utilizat modulul de comunicații RF pentru conectarea între unitatea de detecție a gesturilor și sistemul de bază. În cadrul acestuia, s-a utilizat un alt microcontroler ce avea activate două ieșiri pentru conversie în semnal vocal și în semnal text.

În secțiunea de recepție, se inițializează modulul radio cu adresa dispozitivului de transmisie, și, folosind funcția radio.setReadingPipe(), permitem comunicarea între cele două module. Prin radio.setPALevel() se setează la minim nivelul amplificatorului de putere deoarece cele două module de transmisie-recepție sunt apropiate ca distanță.

Cu ajutorul funcției radio.startListening() se stabilește că modulul va fi folosit ca receptor, iar prin intermediul radio.available() se verifică dacă s-au recepționat date.

Prin intermediul funcției radio.read(&text, sizeof(text)), folosind "&" se indică faptul că se dorește valoarea din adresa indicată de variabila respectivă și, prin sizeof(text) se setează lungimea în bytes pe care o dorim.

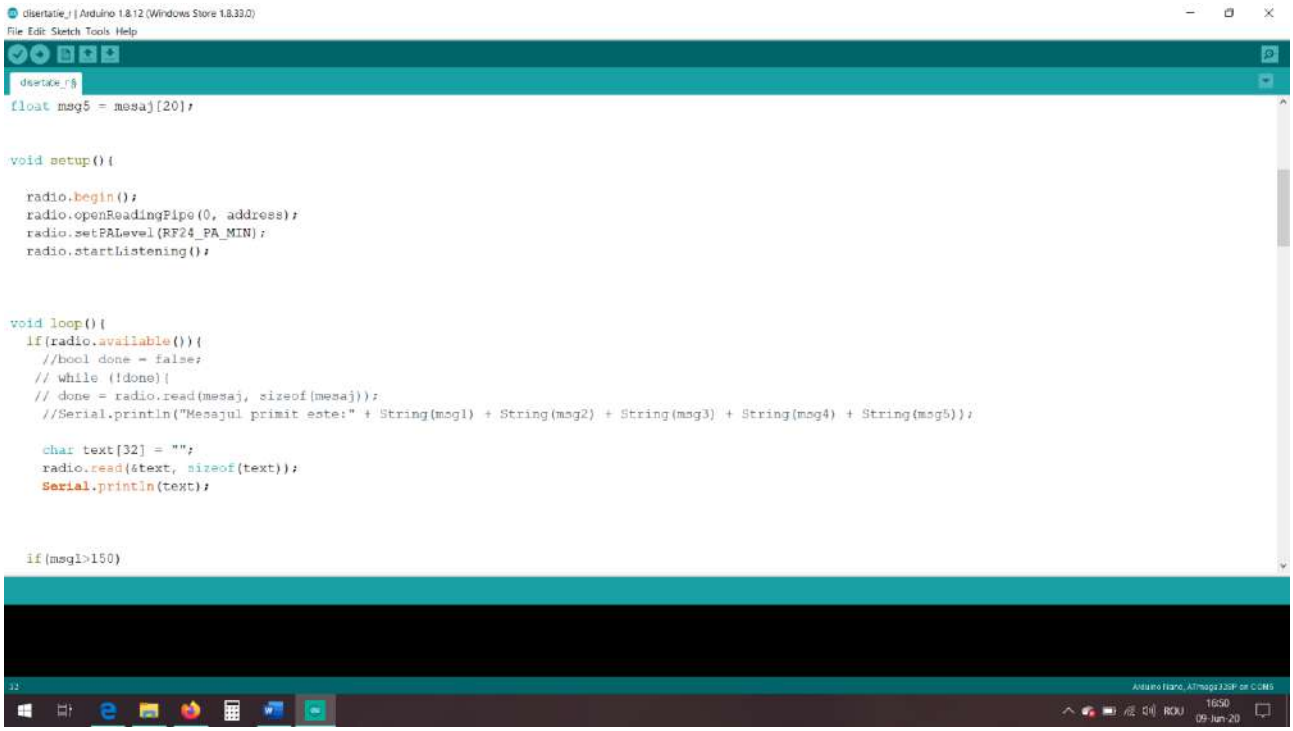


Figura II. 28: Funcții specifice modulului radio în etapa de recepție

După recepționarea mesajului, se inițializează LCD-ul și modulul de redare audio, după care, în funcție de mesajul recepționat, se realizează afișarea acestuia pe ecran și redarea semnalului vocal.

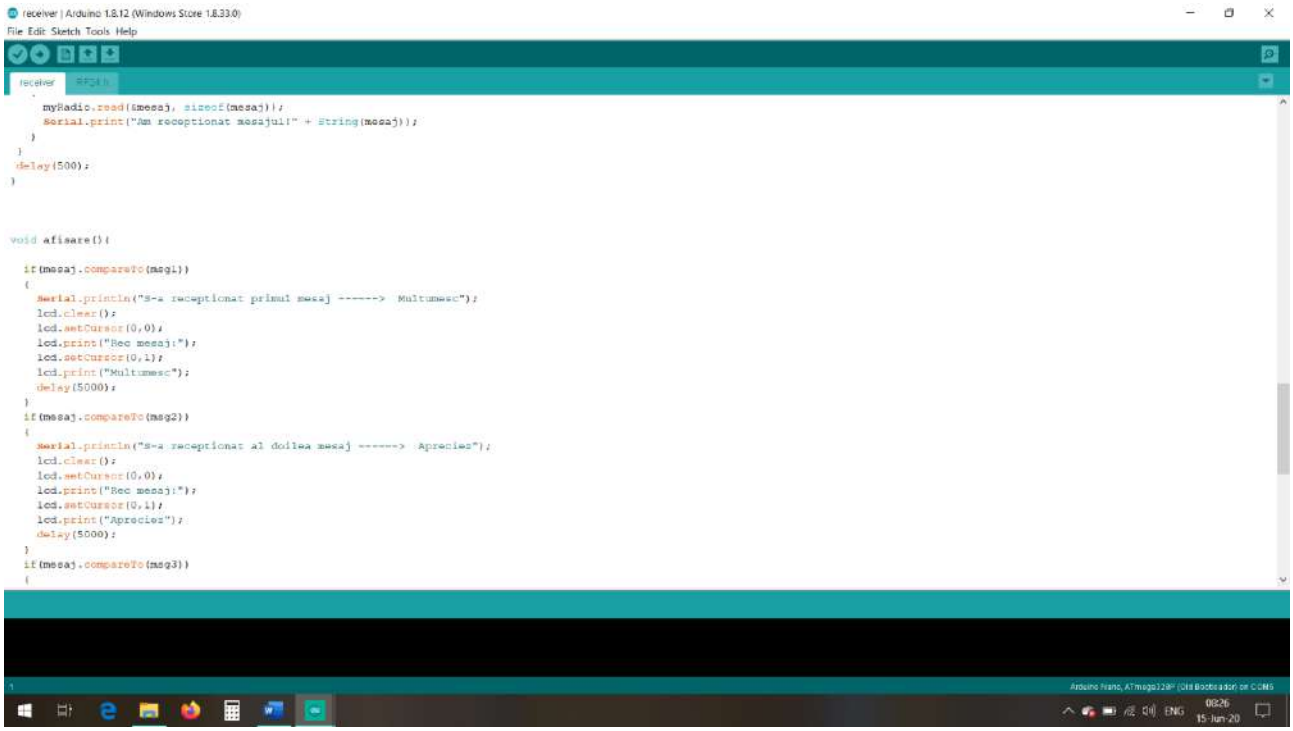


Figura II. 29: Funcții specifice translației gestului – demo1

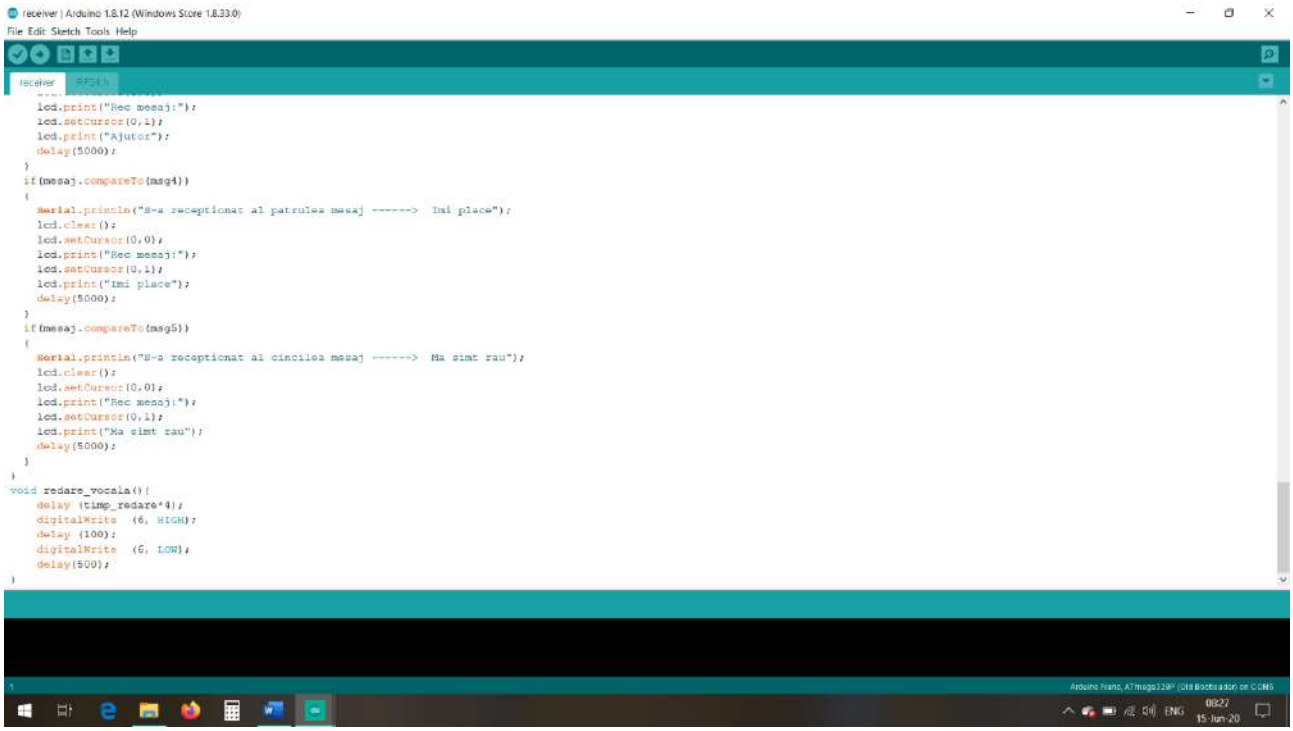


Figura II. 30: Funcții specifice translației gestului – demo2

Sistemul de test final – ce cuprinde atât partea de detecție, cât și pe cea de translație poate fi observat în cadrul următoarei imagini. Se remarcă faptul că circuitul de detecție – ce conține achiziția datelor senzorilor și transmiterea acestora nu este, încă, montat pe mănușă, fiind o fotografie din perioada de testare a circuitelor. Imaginea circuitului final va fi prezentată în capitolul Rezultate.

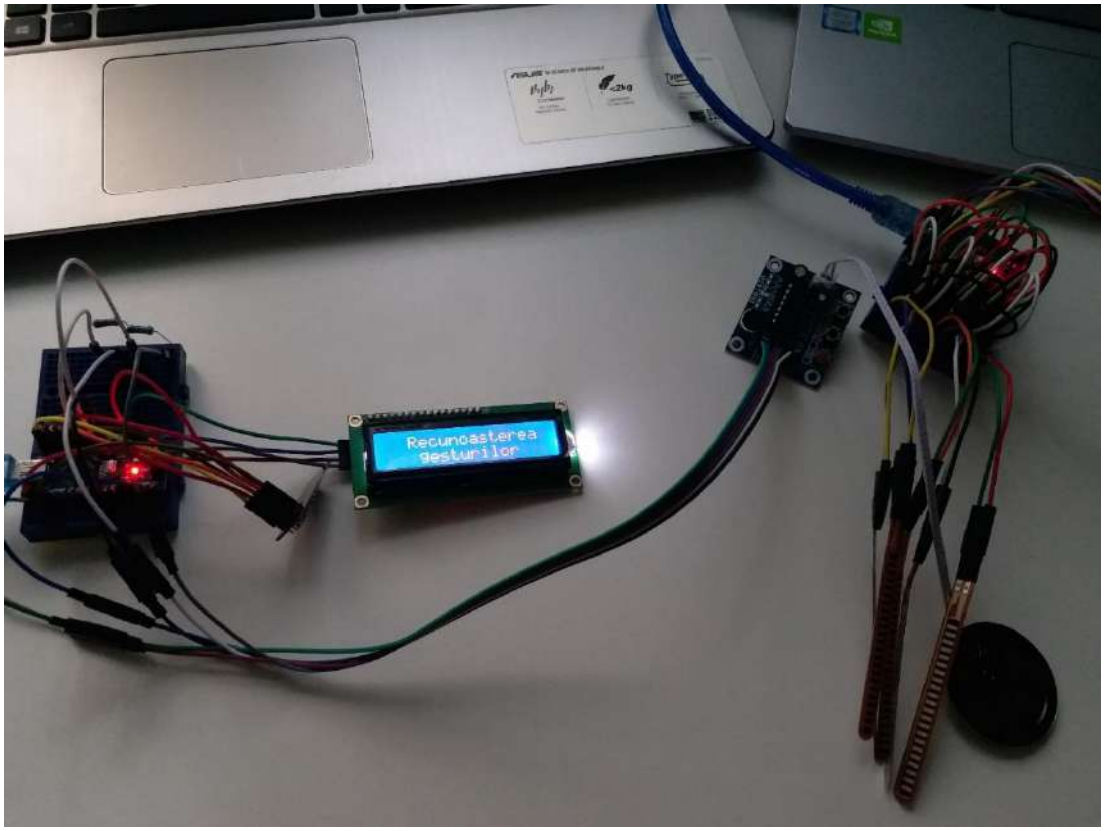


Figura II. 31: Sistemul hardware realizat – partea de emisie și cea de recepție

II.2 Implementare abordarea bazată pe recunoaștere vizuală cu Android

DESCRIEREA PROBLEMATICII

În sistemul de recunoaștere vizuală a gesturilor, se folosește mâna pentru a extrage datele pentru recunoaștere. Astfel, mișcarea mâinii este înregistrată de camera fotografică și se extrag anumite caracteristici ale imaginii pentru achiziția datelor necesare analizei gesturilor.

Implementarea folosește mâna descoperită pentru a concentra informațiile necesare pentru recunoaștere, în cazul în care utilizatorul comunică direct cu sistemul. Pentru a urmări poziția mâinii, se vor folosi algoritmi de machine-learning prezenți în literatură și baza de date cu gesturi predefinite pentru ca regiunea de interes să poată fi determinată. După achiziție, imaginea este procesată și segmentată pentru a fi analizată și pentru a obține caracteristicile unice ale fiecărui semn.

Cu toate acestea, aceste sisteme sunt mai potrivite pentru decodificarea alfabetelor și a numerelor, spre deosebire de perceperea semnelor. Cauza este faptul că un gest în limbajul semnelor depinde și de orientarea mâinii și acest aspect este greu de codificat cu o simplă cameră. Semnele cu poziție comparabilă cu un alt semn pot fi confundate reducând precizia sistemului. Mai mult, procesul de achiziție a imaginii este supus numeroaselor preocupări naturale, de exemplu, poziția camerei, starea fundalului și efectele de iluminare. În plus, iluminarea adecvată este necesară pentru a avea suficientă luminozitate pentru ca gestul să poată fi captat și analizat.

Majoritatea metodelor de recunoaștere a gesturilor conțin de obicei trei etape majore.

Prima etapă este detectarea obiectelor. Scopul acestei etape este de a detecta mâna în imagine. Practic, în acest pas se realizează recunoașterea contururile mâinilor pentru a spori precizia recunoașterii. Problemele comune de imagine conțin luminozitate instabilă, zgomot, rezoluție slabă și contrast. Mediul și dispozitivele mai bune pot îmbunătăți eficient aceste probleme.

Cu toate acestea, aceste aspecte sunt greu de controlat atunci când sistemul de recunoaștere a gesturilor funcționează în mediul real sau devine un produs. Prin urmare, metoda de procesare a imaginilor este o soluție mai bună pentru a rezolva aceste probleme și pentru a construi un sistem de recunoaștere a gesturilor eficient și robust.

A doua etapă este recunoașterea obiectelor. Dacă mâna este recunoscută, următorul pas este identificarea gesturilor. În această etapă, caracteristicile diferențiate și selecția eficientă a clasificatorilor sunt problemele majore.

A treia etapă este analizarea gesturilor secvențiale pentru identificarea instrucțiunilor sau comportamentelor utilizatorilor. Practic, în această etapă, se identifică gestul mâinii și se compară cu gesturi cunoscute din baza de date anterior procesată.

Achiziția datelor

Principalul dispozitiv utilizat ca intrare în cadrul acestui sistem este camera foto. Datele de intrare sunt de forma unei imagini, ce conține un gest care poate fi surprins, cu ușurință, de aparatul

fotografic, dar, pentru o acuratețe mai mare a sistemului și a imaginilor de intrare, se va folosi o cameră cu specificații mai bune.

Sistemele de recunoaștere a limbajului semnelor sunt dezvoltate în două etape, achiziția și clasificarea datelor. Principalul avantaj al utilizării camerei fotografice este că elimină nevoile senzorilor și reduce costurile din construirea sistemului. După cum știm, camera este disponibilă în aproape toate dispozitivele portabile. Dezavantajul folosirii camerei fotografice este că este necesară o preprocesare bună a imaginii pentru obținerea funcției.

Procesarea imaginii

După selectarea imaginii, aceasta este convertită la scară de gri, unde o imagine de 24 de pixeli este convertită în imagine de pixeli de 8 biți care conține doar informații despre intensitate.

Imaginea binară este scăzută cu imaginea după dilatare, ceea ce dă o imagine de ieșire care conține doar degete. Imagine scăzută = Imagine binară - Imagine după operații morfologice.

Procesarea imaginii este realizată pentru a elimina impuritățile din imagine, precum eliminarea zgomotului și, de asemenea, îmbunătățește unele caracteristici importante din imagine. Îmbunătățirea modifică imaginea redimensionare și normalizare. Imaginile nu au dimensiunea standard, deci, pentru a standardiza imaginea s-a ales o formă în dimensiunea 200 × 200. În timpul executării acestor schimbări, se introduce și zgomot. Un aspect utilizat de-a lungul experimentelor este bazat pe filtrul median. De asemenea, de-a lungul experimentelor, unele tehnici au inclus îmbunătățirea contrastului și modificarea caracteristicilor de luminanță și crominanță.

Un alt aspect esențial, este forma obiectelor ce necesită recunoaștere. Aceasta este o caracteristică vizuală importantă și este una dintre caracteristicile primitive pentru descrierea conținutului de imagine. Descriptorii de formă pot fi împărțiți în două categorii principale: metode bazate pe regiune și metode pe contur. Metodele bazate pe regiuni utilizează întreaga zonă a unui obiect pentru descrierea formei, în timp ce metodele bazate pe contur folosesc numai informațiile prezente în conturul unui obiect.

Extracție caracteristică

Gestul este detectat folosind metoda de extracție a caracteristicilor. Practic, se îndepărtează toți pixelii redundanți, în afară de cei ce sunt detectați ca fiind utili. Extragerea caracteristicilor se face prin eliminarea pixelilor albi sub un prag.

Sistemul propus constă în următorii pași de bază:

- Se realizează captura imaginii de pe dispozitiv;
- Se obțin datele importante ale imaginii;
- Se realizează detecția mâinii;
- Se realizează extragerea caracteristicilor gestului;
- Compilarea datelor de antrenament;
- Testarea gestului;
- Recunoașterea gestului;
- Afișarea datelor sub formă de imagine, text sau redarea semnalului audio aferent.

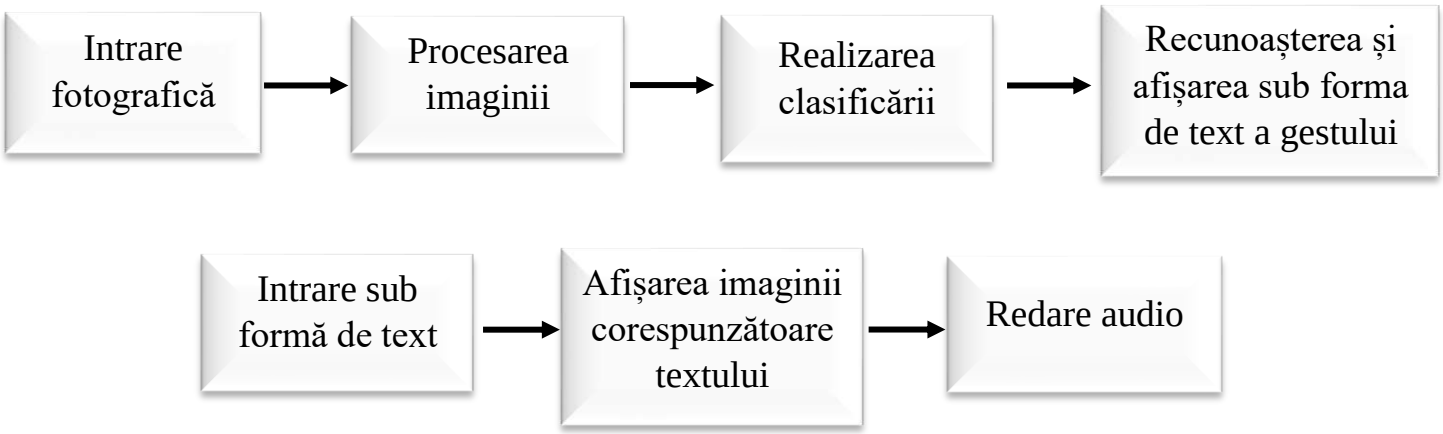


Figura II. 32: Diagrama bloc a sistemului bazat pe recunoaștere vizuală prin aplicație Android

Baza de date joacă un rol important în furnizarea de rezultate corecte și eficiente pentru sistem. După potrivirea rezultatului corect, ieșirea (sub formă de text, imagine sau redare audio) este afișată pe ecran. Pentru crearea bazei de date, s-au folosit anumite imagini descărcate de pe internet – se menționează că, inițial, acestea au fost capturate cu o cameră fotografică, doar că apăreau erori de detecție, cele de pe internet, având fundal alb, fiind mult mai clare. Astfel, s-au folosit treisprezece imagini pentru detecție și una pentru a anunța că datele introduse nu pot fi corelate cu baza preexistentă. Imaginile au un contrast și o iluminare bună pentru a reduce zgomotul și pentru ușurință în cazul segmentării.

Alături de aceste imagini, s-au introdus și mai multe variante pentru șirurile de caractere de detecție, iar pentru redarea semnalului audio s-au folosit biblioteci open-source.

Se menționează că recunoașterea gesturilor din cadrul acestui proiect nu se bazează pe recunoașterea întregului limbaj al semnelor, ci pe gesturi ce se asociază cu un text predefinit.

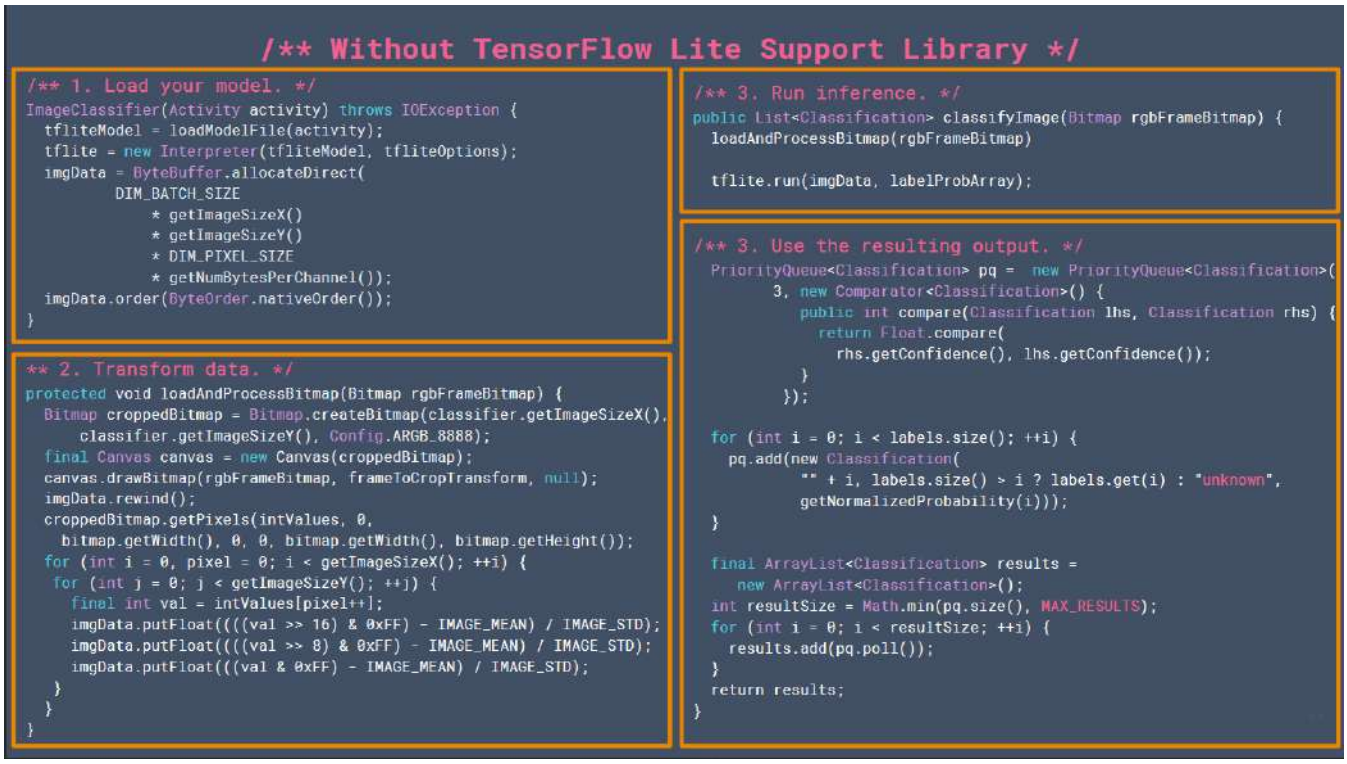
Modelul propus constă din două tehnici – pentru o comunicație bidirecțională, în care una dintre acestea funcționează pe recunoaștere statică a cuvintelor prin alăturarea de imagini, iar cealaltă parte constă în recunoașterea efectivă a gesturilor date în imagini.

Toate tehnicile menționate anterior sunt modelate cu structuri de tip machine learning. Acestea au o structură mult diferită de ceea ce se aplică în programarea tradițională.

- În programarea tradițională, există un set de date și un set de reguli pentru care se obțin niște răspunsuri –> un bun exemplu ar fi o simplă operație de adunare a doi termeni
- În programarea bazată pe tehnici de învățare automată, avem un set de date și un set de răspunsuri pe baza cărora se creează un set de reguli –> un exemplu ar putea fi viteza de deplasare:
    - Dacă viteza unei persoane este de sub 5 km/h spunem că aceasta merge;
    - Dacă viteza unei persoane este între [5, 9] km/h spunem că aceasta aleargă;
    - Dacă viteza unei persoane este între [9, 12] km/h spunem că aceasta merge pe bicicletă;

Practic, aplicațiile ce dețin structuri de învățare automată se bazează pe date de intrare ce sunt de tip imagini, text sau video ce sunt introduse într-un cod de aplicație ce realizează anumite operații asupra acestor date, pe baza unor modele (baze de date de antrenament), pentru a lua anumite decizii la ieșire. În cazul aplicației dezvoltate, partea de antrenare, clasificare și recunoaștere de gesturi este realizată prin intermediul TensorFlow Lite.

Un exemplu de cod în Java pentru TensorFlow este:



```
/** Without TensorFlow Lite Support Library */

/** 1. Load your model. */
ImageClassifier(Activity activity) throws IOException {
  tfliteModel = loadModelFile(activity);
  tflite = new Interpreter(tfliteModel, tfliteOptions);
  imgData = ByteBuffer.allocateDirect(
          DIM_BATCH_SIZE
              * getImageSizeX()
              * getImageSizeY()
              * DIM_PIXEL_SIZE
              * getNumBytesPerChannel());
  imgData.order(ByteOrder.nativeOrder());
}

/** 2. Transform data. */
protected void loadAndProcessBitmap(Bitmap rgbFrameBitmap) {
  Bitmap croppedBitmap = Bitmap.createBitmap(classifier.getImageSizeX(),
      classifier.getImageSizeY(), Config.ARGB_8888);
  final Canvas canvas = new Canvas(croppedBitmap);
  canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
  imgData.rewind();
  croppedBitmap.getPixels(intValues, 0,
    bitmap.getWidth(), 0, 0, bitmap.getWidth(), bitmap.getHeight());
  for (int i = 0, pixel = 0; i < getImageSizeX(); ++i) {
   for (int j = 0; j < getImageSizeY(); ++j) {
     final int val = intValues[pixel++];
     imgData.putFloat((((val >> 16) & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
     imgData.putFloat((((val >> 8) & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
     imgData.putFloat(((val & 0xFF) - IMAGE_MEAN) / IMAGE_STD);
   }
  }
}

/** 3. Run inference. */
public List<Classification> classifyImage(Bitmap rgbFrameBitmap) {
  loadAndProcessBitmap(rgbFrameBitmap)

  tflite.run(imgData, labelProbArray);

/** 3. Use the resulting output. */
  PriorityQueue<Classification> pq =  new PriorityQueue<Classification>(
        3, new Comparator<Classification>() {
          public int compare(Classification lhs, Classification rhs) {
            return Float.compare(
              rhs.getConfidence(), lhs.getConfidence());
          }
        });

  for (int i = 0; i < labels.size(); ++i) {
    pq.add(new Classification(
            "" + i, labels.size() > i ? labels.get(i) : "unknown",
            getNormalizedProbability(i)));
  }

  final ArrayList<Classification> results =
     new ArrayList<Classification>();
  int resultSize = Math.min(pq.size(), MAX_RESULTS);
  for (int i = 0; i < resultSize; ++i) {
    results.add(pq.poll());
  }
  return results;
}
```

Figura II. 33: Exemplu cod pentru interpretare a unei activități în TensorFlow Lite [22]

FIREBASE

Firebase este un serviciu cloud ce deține baze de date NoSQL predefinite și biblioteci pentru a le accesa din aplicații de tip interfață web sau IOS sau Android.

Potrivit [26] bazele de date de tip NoSQL sunt baze de date nerelaționale care stochează datele în cadrul unei singure structuri și care au apărut din nevoia de stocare eficientă a informațiilor. Astfel, acest tip de baze de date sunt foarte eficiente din punct de vedere al dezvoltării software deoarece mai multe aplicații diferite pot accesa aceste resurse în același timp. De asemenea, oferind, suplimentar, procesare în Cloud, aplicabilitatea acestora a crescut și mai mult deoarece aproape oricine – de la programatori amatori, la programatori profesioniști – au reușit să le folosească pentru aplicații și date. Dorindu-se extinderea acestora în cadrul companiilor din diverse regiuni geografice, bazele de date nerelaționale au răspuns capabilităților oferind un răspuns pozitiv la cerințele de scalabilitate.

Prin intermediul unui serviciu web, Firebase oferă posibilitatea de a programa aplicații ce conțin procesări de date în timp real prin intermediul unei interfețe de utilizator destul de intuitivă.

Astfel, prin intermediul acestui serviciu, se pot procesa baze de date preexistente sau se pot încărca baze de date proprii, oferind și posibilitatea de a publica bazele de date proprii spre a fi disponibile și altor programatori. Se menționează că, în cadrul dezvoltării proiectului, s-a utilizat o bază de date proprie ce a fost, ulterior, publicată și poate fi regăsită sub denumirea precizată în cadrul programului din Anexe.

Pentru a utiliza serviciul Firebase, este nevoie ca acesta să fie adăugat în aplicație și să fie utilizat un cont activ. Se menționează că acest serviciu este accesibil în urma unui abonament preplătit. Astfel, odată creată aplicația și create legăturile, adăugarea Firebase se face prin intermediul unui fișier JSON și definirea unor extensii și a unor reguli de accesare în cadrul aplicației.

Funcțiile principale ale Firebase sunt legate de autentificare, stocare și management al bazelor de date, ML, analiza datelor, configurare la distanță, testare comptabilități ale datelor, etc.

Se menționează că, în cadrul anexelor este redat modul de utilizare al Firebase în cadrul programului, atât prin intermediul unor imagini din momentul antrenării rețelei, cât și prin intermediul aplicabilității la nivel de cod.

EXEMPLE DE COD ANDROID FOLOSIT ȘI EXPLICAȚII

1. Diferența dintre OnClickListener() și OnClick():

Pe scurt și în termeni generali, OnClickListener() este o funcție care așteaptă ca cineva să facă un click pentru a genera o acțiune, pe când OnClick() determină acțiunea care se petrece atunci când cineva realizează un click.

Mai în detaliu, OnClickListener() este o interfață ce trebuie implementată în interiorul unei clase pentru ca variabilele și metodele să poată obține funcționalități suplimentare pentru a gestiona anumite evenimente sau anumite funcții. Astfel, utilitatea acestei funcții apare în momentul în care se dorește schimbarea comportamentului unei variabile de tip buton în cadrul aplicației sau se dorește ca acesta să apeleze altă metodă decât cea predefinită. De asemenea, acesta se poate păstra în program, dacă se dorește doar dezactivarea temporară prin setarea funcției cu parametri nuli. Ca urmare a acestui fapt, va apărea o alertă de tip warning, dar restul programului va funcționa în parametri.

OnClick() face legătura funcțiilor implementate în intermediul unei clase cu fișierele .xml ce conțin definiția unui mod de vizualizare(View). Astfel, aceasta trebuie să aibă argumente din clasa View pentru a funcționa. Când definiți o variabilă pentru așteptarea unui eveniment folosind atributul onClick, vizualizarea caută o metodă cu acest nume doar în activitatea definită special pentru ea. Acest aspect nu este important în momentul în care aplicația utilizează doar un MainActivity, însă devine extrem de dificil în momentul în care se vor implementa Fragmente, așa cum este și cazul acestui proiect.

Din fragmentul denumit “Dezvoltare.java” din cadrul acestui proiect, s-au preluat următoarele părți de cod:

```
//Se creează o variabilă de tip buton ce va fi utilizată pentru a exemplifica diferența dintre funcțiile mai sus menționate
//De asemenea, se menționează că această parte de cod este implementată efectiv în cadrul proiectului

buton_dezvoltare = (Button)
rootView.findViewById(R.id.buton_dezvoltare);
//Se poate observa că variabila definite în fișierul .java are aceeași denumire cu cea definite în fișierul .xml

buton_dezvoltare.setOnClickListener ( new View.OnClickListener()
//Se poate observa că este apelată interfața OnClickListener() deoarece se așteaptă ca variabila - definită anterior - buton_dezvoltare  să aibă atașat un eveniment.

{
  @Override //adnotare folosită pentru suprascrierea unei metode

  public void onClick(View view) //se apelează funcția onClick ce are ca argument modul de vizualizare deoarece face legătura cu argumente din fișierele .xml, ce sunt moduri de acțiune vizuală
        {
//Se definește o altă variabilă de tip imagine – declarată final deoarece nu își va schimba valoarea – denumită imagine dezvoltare, ce are corespondent, cu același nume, în fișierul .xml
              final ImageView imagine_dezvoltare = (ImageView)
rootView.findViewById(R.id.imagine_dezvoltare);

//variabila definită anterior va avea implementată și o funcție de tip resursă deoarece trebuie încărcată din fișierele de tip drawable din cadrul proiectului

       imagine_dezvoltare.setImageResource(R.drawable.dezvoltare);

//numele "dezvoltare" este denumirea imaginii în cadrul fișierelor drawable
        }
}

);
//Practic, toată această funcție are scopul de a afișa o imagine când este acționat un anumit buton
//Se menționează că, atât imaginea, cât și butonul sunt predefinite în cadrul fișierelor .xml
```

Fișierele .xml aferente butonului și a imaginii:

```
<Button
    android:id="@+id/buton_dezvoltare" //setarea denumirii sub care va fi utilizată această variabilă, în cadrul programului .java

    android:layout_width="174dp" //parametru de dimensiune a vizualizării variabilei în cadrul proiectului
//Unitatea dp provine de la density-independent pixels
```

```
    android:layout_height="wrap_content" ///parametru de dimensiune a vizualizării variabilei în cadrul proiectului

    android:layout_marginLeft="120dp"  ///parametru ce setează shiftarea la stanga a variabilei în modul de vizualizare

    android:background="@color/verde_turcoaz"  //parametru ce setează culoarea fundalului variabilei de tip buton

    android:gravity="center_horizontal"  ///parametru de centrare a variabilei în cadrul modului de vizualizare

    android:drawablePadding="5dp"//parametru de încadrare a variabilei de tip buton
    android:paddingLeft="10dp"  //parametru de încadrare a variabilei de tip buton
    android:paddingRight="10dp"  //parametru de încadrare a variabilei de tip buton

    android:text="Adauga"  //parametru ce conține textul ce va fi scris în cadrul variabilei de tip buton
    android:layout_centerInParent="true"/> //Specifică modul în care este poziționată o vizualizare într-un RelativeLayout.
```

```
<ImageView
    android:id="@+id/imagine_dezvoltare"  //setarea denumirii sub care va fi utilizată această variabilă, în cadrul programului .java

    android:layout_width="252dp"  //parametru de dimensiune a vizualizării variabilei în cadrul proiectului
//Unitatea dp provine de la density-independent pixels

    android:layout_height="328dp"//parametru de dimensiune a vizualizării variabilei în cadrul proiectului

    android:layout_marginLeft="85dp"  //parametru ce setează shiftarea la stanga a variabilei în modul de vizualizare

    android:layout_marginTop="5dp"//parametru ce setează shiftarea în jos a variabilei în modul de vizualizare

    android:scaleType="fitCenter"  //opțiune pentru scalarea limitelor unei imagini
/>
```

2. Intent:

Un Intent este un obiect pe care îl puteți utiliza pentru a solicita o acțiune de la o altă componentă a aplicației. Acestea se utilizează pentru începerea unei activități sau pornirea unui serviciu – printr-o comandă de startActivity(). Sistemul Android se ocupă de rezoluția tuturor intențiilor deoarece modul lor de definire este destul de diferit astfel că intențiile pot fi, uneori, extren

de vagi. Astfel, în unele cazuri se poate solicita a fi lansate doar activitățile care să corespundă anumitor criterii.

```
Intent checkIntent = new Intent(); //se crează o variabilă de tip Intent denumită checkInternet
checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
//acestei variabile îi este asociată o acțiune de pornire a activităților din motorul TextToSpeech al platformei pentru a verifica instalarea și disponibilitatea corectă a fișierelor de resurse de sistem

startActivityForResult(checkIntent, CHECK_CODE); // se lansează activitatea pentru care ați dori un rezultat când s-a terminat.
//In cazul acesta, se verifică dacă activitatea detine codul prestabilit.
```

Obiectele de tip Intent se utilizează și în cadrul fișierului AndroidManifest.xml prin intermediul obiectelor implicite. Astfel, se începe prin compararea conținutului intenției cu filtrele de intenție declarate în fișierul manifest al altor aplicații de pe dispozitiv. Dacă intenția se potrivește cu un filtru de intenție, sistemul pornește acea componentă și îi livrează obiectul Intent. Dacă nu, se realizează intențiile predefinite în MainActivity sau în Fragmente.

```
<intent-filter>

    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />

</intent-filter>
```

3. Funcții ce utilizează Firebase

```
private void setLabel(Uri x) //definirea funcției ce utilizează ca parametru un parametru de clasă Uri – ce este utilizată, în cea mai mare parte, pentru parametric ce folosesc resurse din Internet sau din Intranet
{
//se defineste variabila var_local ce va fi utilizată pentru a seta calea din cadrul căreia se vor încărca etichetele imaginii
    var_local = new FirebaseAutoMLLocalModel.Builder()
            .setAssetFilePath("model/manifest.json").build();


    try
       {
//se setează un prag “de încredere” a valorilor
        FirebaseVisionOnDeviceAutoMLImageLabelerOptions y =
      new FirebaseVisionOnDeviceAutoMLImageLabelerOptions
.Builder(localModel).setConfidenceThreshold(0.0f).build();

//se utilizează variabila l pentru încărcarea etichetelor
       l =
FirebaseVision.getInstance().getOnDeviceAutoMLImageLabeler(y);
```

```
//se utilizează constrângeri pentru disponibilitatea versiunilor de SDK
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT)
       {
//se încarcă imaginea alături de parametrul aferent etichetei
            image = FirebaseVisionImage.fromFilePath(
requireNonNull ( Gest.this.getActivity () ), x);
        }

//se realizează procesarea imaginii și alăturarea etichetei
        processImageLabeler(labeler, image);
    }

//se tratează excepțiile prin afișarea stivei
      catch (FirebaseMLException | IOException e)
       {
        e.getStackTrace ();
       }
}
```

```
private void parteRemote() // definirea funcției pentru utilizarea sursei din Internet
{
   progressDialog.show(); //funcție ce indică progresul funcționării unei aplicații

//prin intermediul variabilei r se va încărca calea aferentă bazei de date de antrenare
    r= new
FirebaseAutoMLRemoteModel.Builder("Gesture_dataset_Mia_20205241536
").build();

// prin intermediul variabilei x se va încărca modelul în cadrul programului
// această funcție necesită conexiune la internet
    x = new
FirebaseModelDownloadConditions.Builder().requireWifi().build();

// se va realiza identificarea și descărcarea modelului
    FirebaseModelManager.getInstance().download(r, x)
.addOnCompleteListener(new OnCompleteListener<Void>()
       {
// se verifică compatibilitatea pentru rulare
       @RequiresApi(api = Build.VERSION_CODES.KITKAT)

// se rescrie metoda onComplete pentru decuparea imaginii
       @Override
           public void onComplete(@NonNull Task<Void> task)
             {
// se realizează decuparea imaginii
                    CropImage.activity().start( requireNonNull (
Gest.this.getActivity () ) );
                }
          });
}
```

Diverse implementări și rezultate intermediare

În cazul aplicației Android, de-a lungul realizării, au existat mai multe părți intermediare. Câteva dintre acestea, vor fi expuse în continuare.

În prima parte a dezvoltării, am realizat o aplicație de tip listă cu fotografie și text. Această parte a fost incipientă pentru adaptare cu mediul Android IDE. Practic, s-a creat o clasă ListView și s-au folosit obiecte de tip TextView și ImageView pentru fiecare rând. Aceste widget-uri sunt potrivite pentru selectarea unui set mic de opțiuni și pentru o vizualizare rapidă, fiind extrem de intuitive și simple.

Printre clasele utilizate sunt este și DataAdapter, folosită pentru a furniza o colecție de date unui widget. Aceste date pot proveni dintr-o varietate de surse, cum ar fi matrici, imagini, siruri de caractere sau baze de date mari.

Ciclu unui ListView este de tipul:

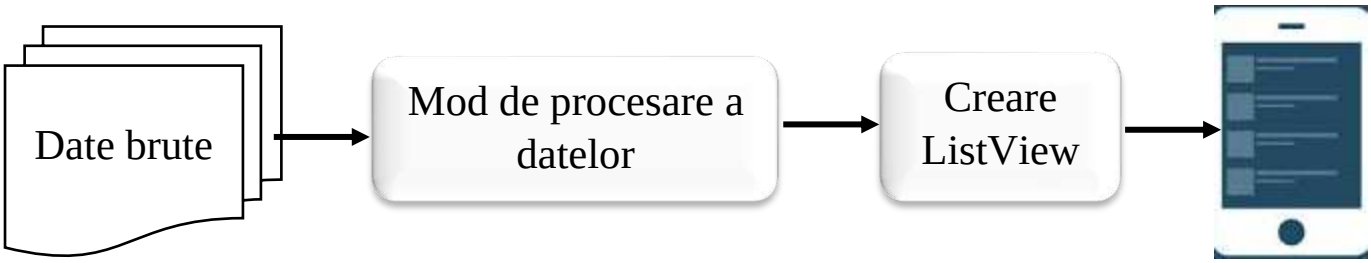


Figura II. 34: Ciclul de formare al unei aplicații de tip ListView

Câteva exemple din cadrul codului aplicației sunt redate și explicate în următoarele paragrafe:

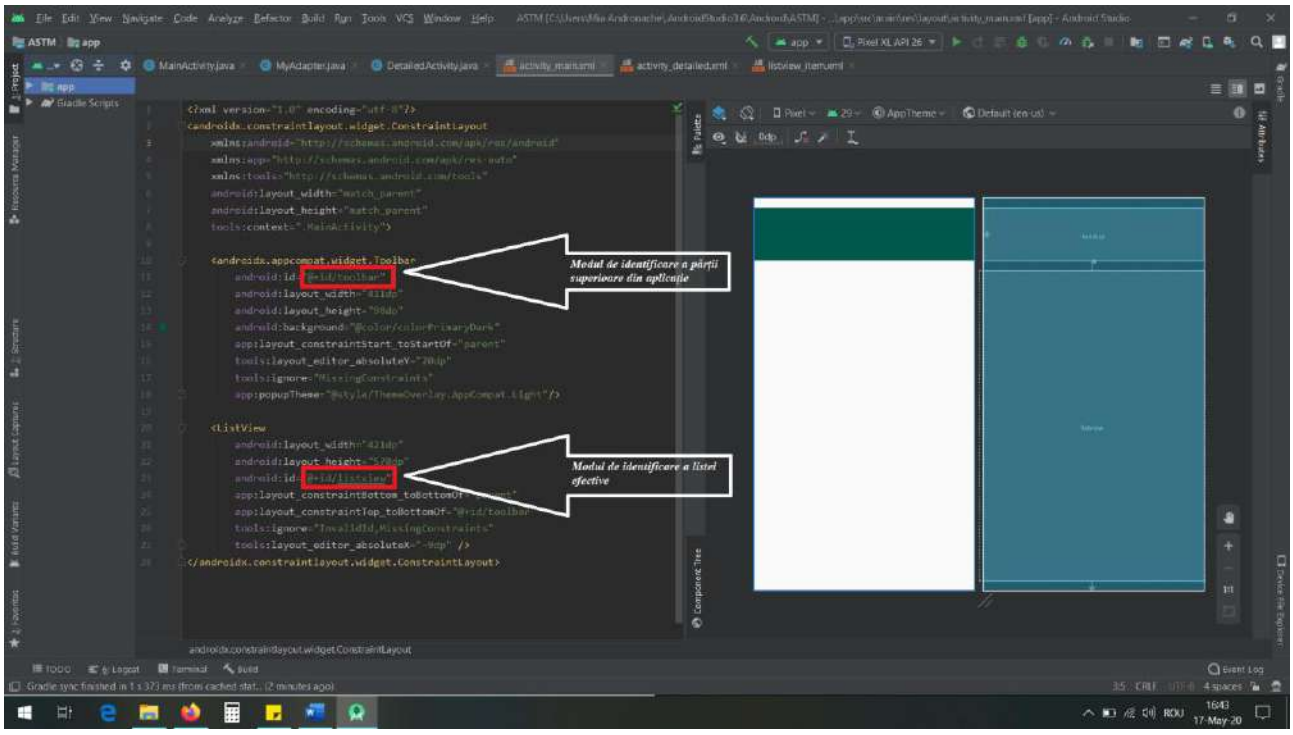


Figura II. 35: Detaliere cod activity_main.xml

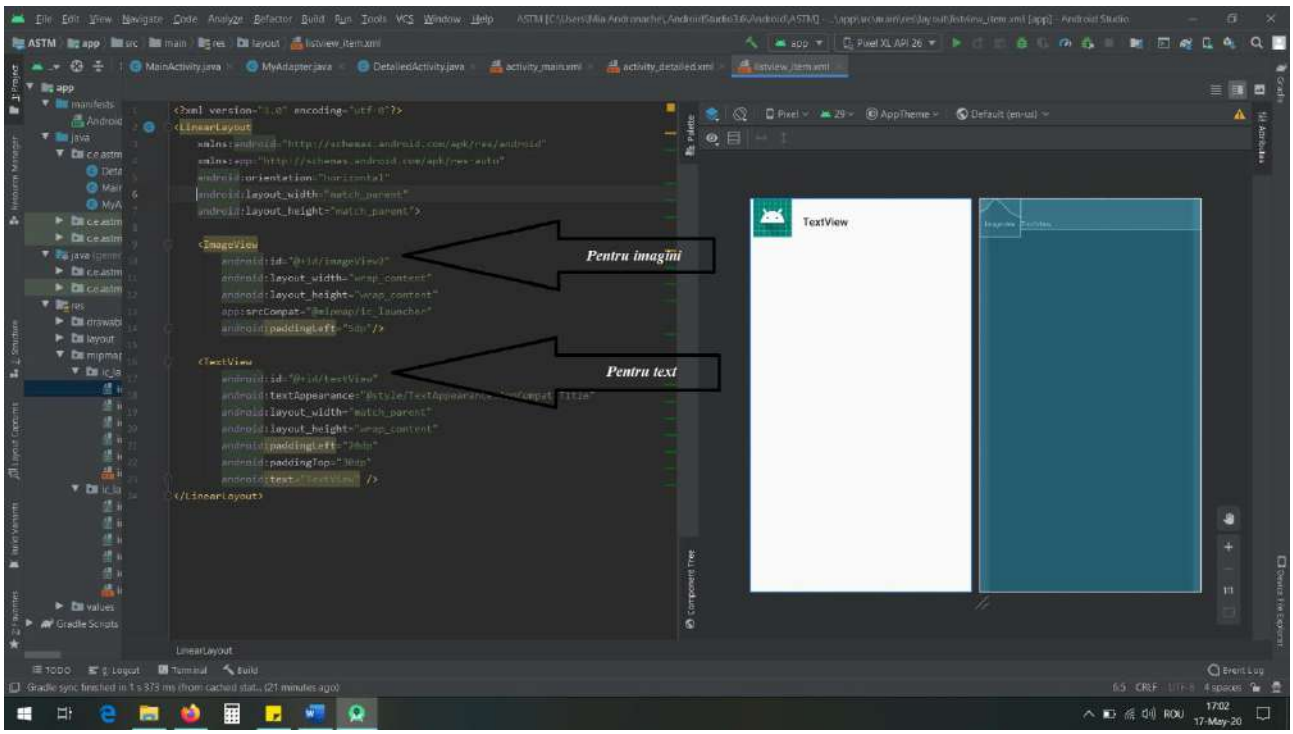


Figura II. 36: Detaliere cod listview_item.xml

În cadrul imaginii următoare poate fi vizualizată o parte din codul pentru MainActivity unde sunt prezente definirea vectorului de șiruri de caractere (o matrice bidimensională) și a vectorului ce conține imaginile aferente.

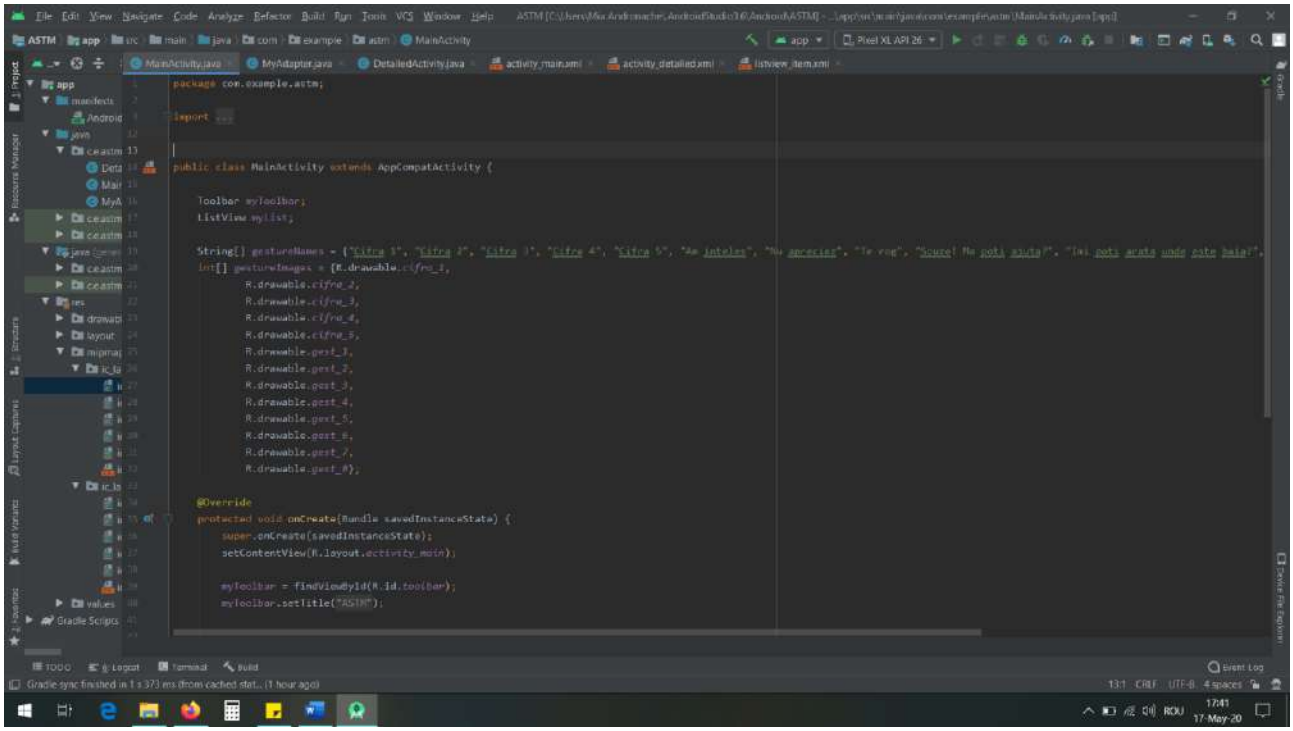


Figura II. 37: Detaliere cod MainActivity.java

Exemplu cu rularea efectivă a aplicației – mai multe exemple pot fi vizualizate în cadrul Anexei 3.

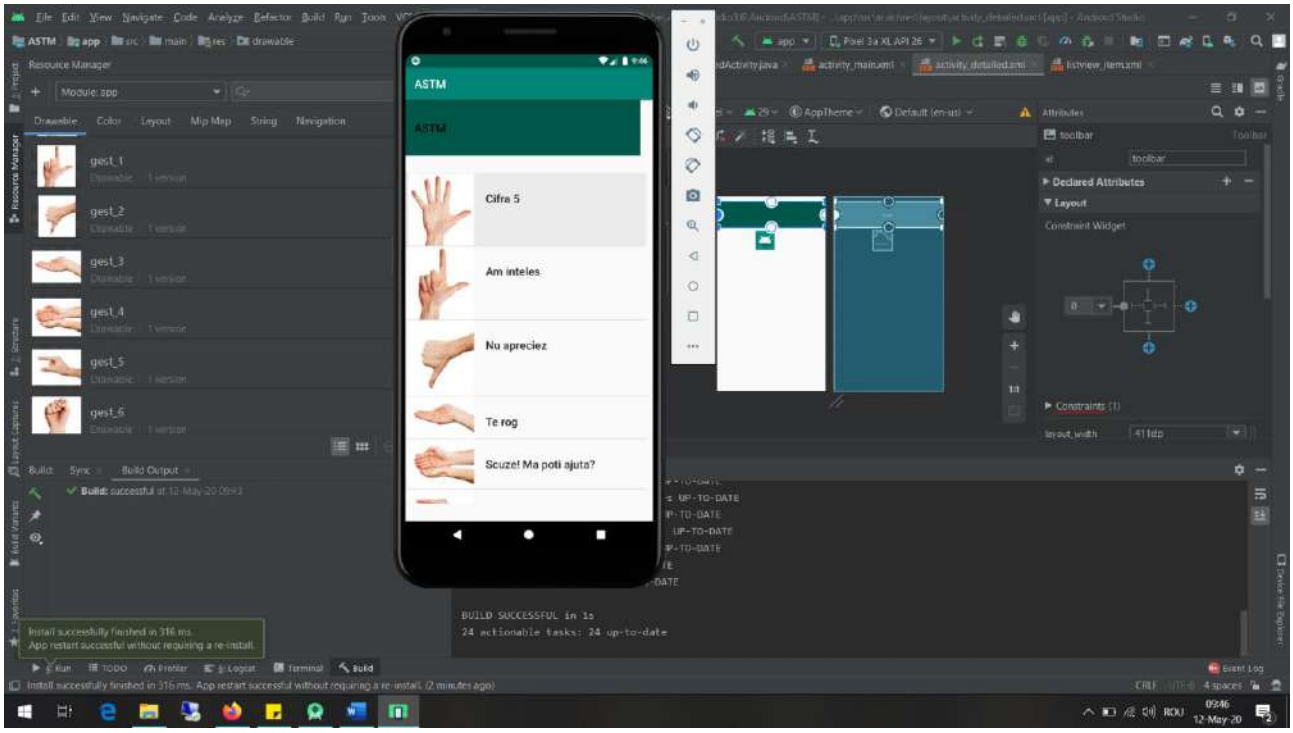


Figura II. 38: Aplicația de tip ListView

Următoarea etapă din cadrul implementării a fost realizarea unei aplicații de tip introducere text – redare audio, în cadrul căreia am introdus anumite cuvinte ce au fost redate cu ajutorul difuzorului de la telefon.

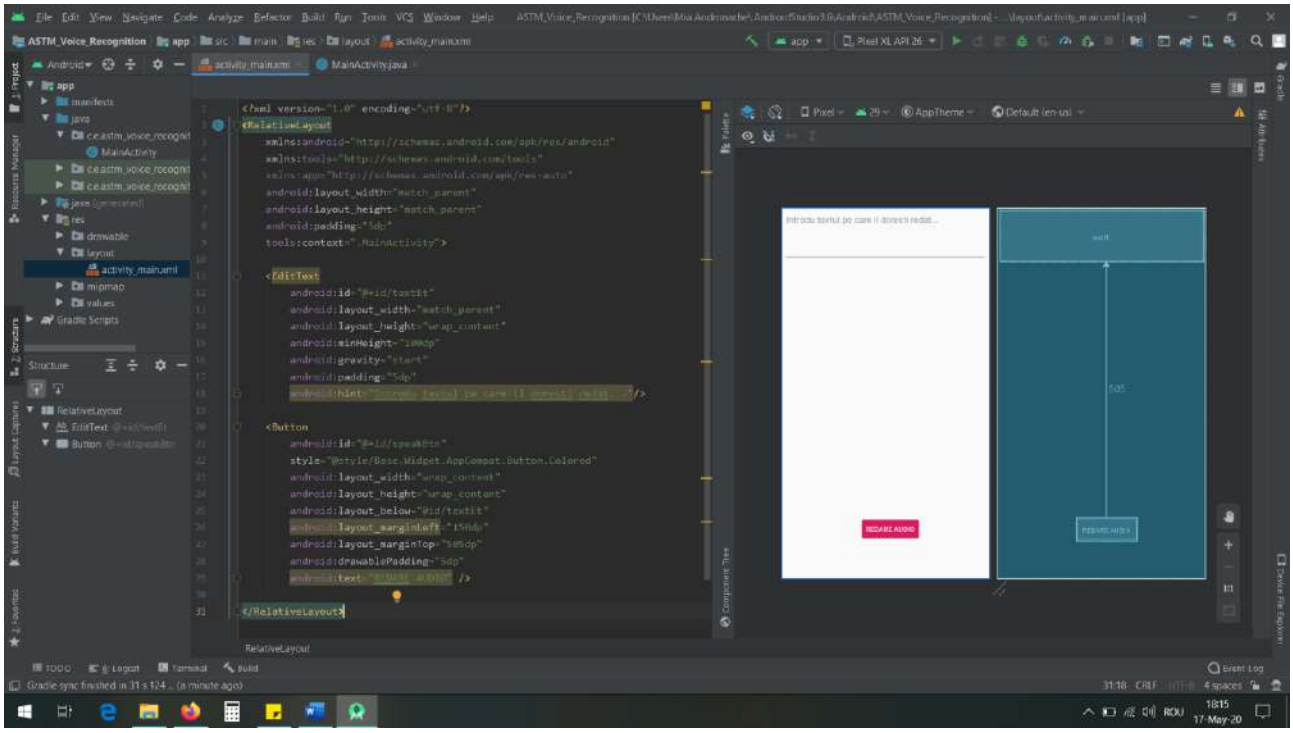


Figura II. 39: Detaliere cod activity_main.xml pentru a doua aplicație

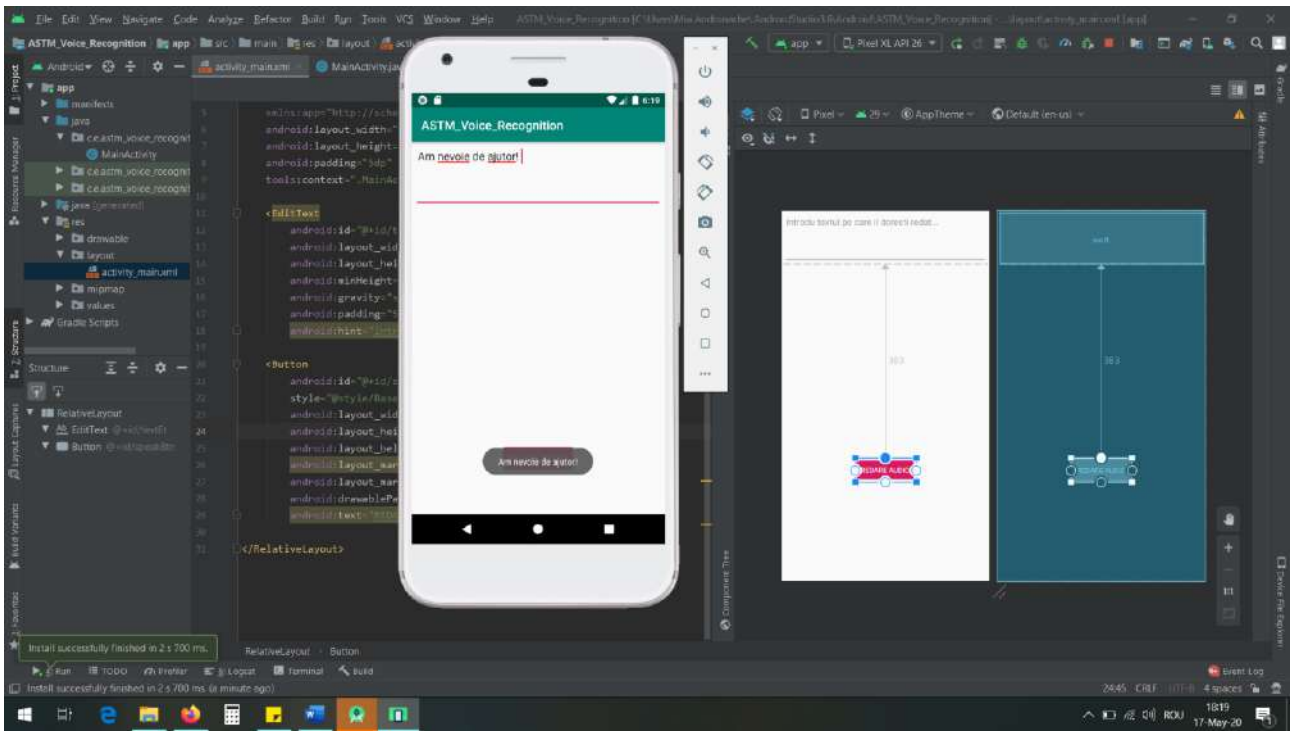


Figura II. 40: Aplicația de tip sintetizare vocală

În codul prezent în imaginea de mai jos, se poate observa că limba de redare este franceză. Se menționează că nu există o bibliotecă pentru limba română, iar cuvintele pronunțate în limba franceză și limba italiană seamănă destul de mult cu cele în limba română, având fond lexical comun.

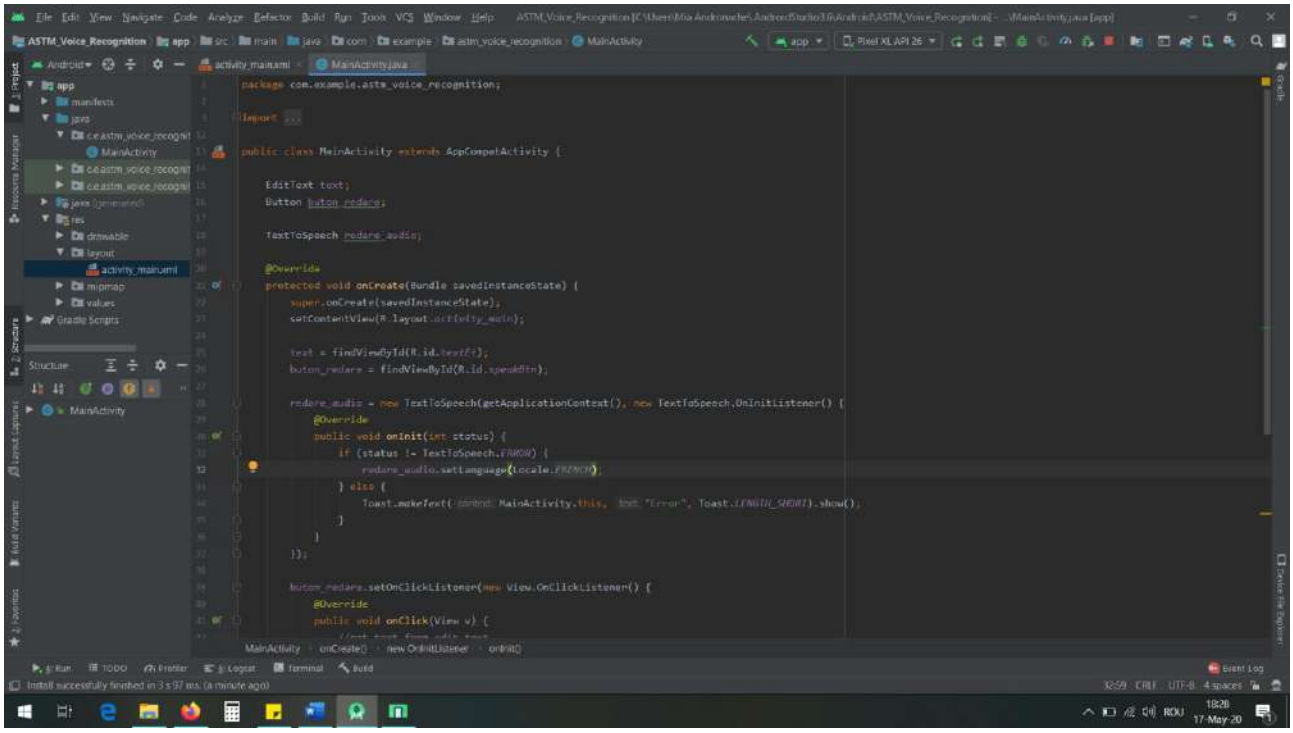


Figura II. 41: Detaliere cod MainActivity.java pentru aplicația de tip sintetizare vocală

Următorul pas a fost integrarea celor două aplicații într-una singură.

În imaginea de mai jos se poate observa structura de bază a aplicației ce conține imaginea gestului, textul ce urmează a fi afișat și butonul pentru redare audio.

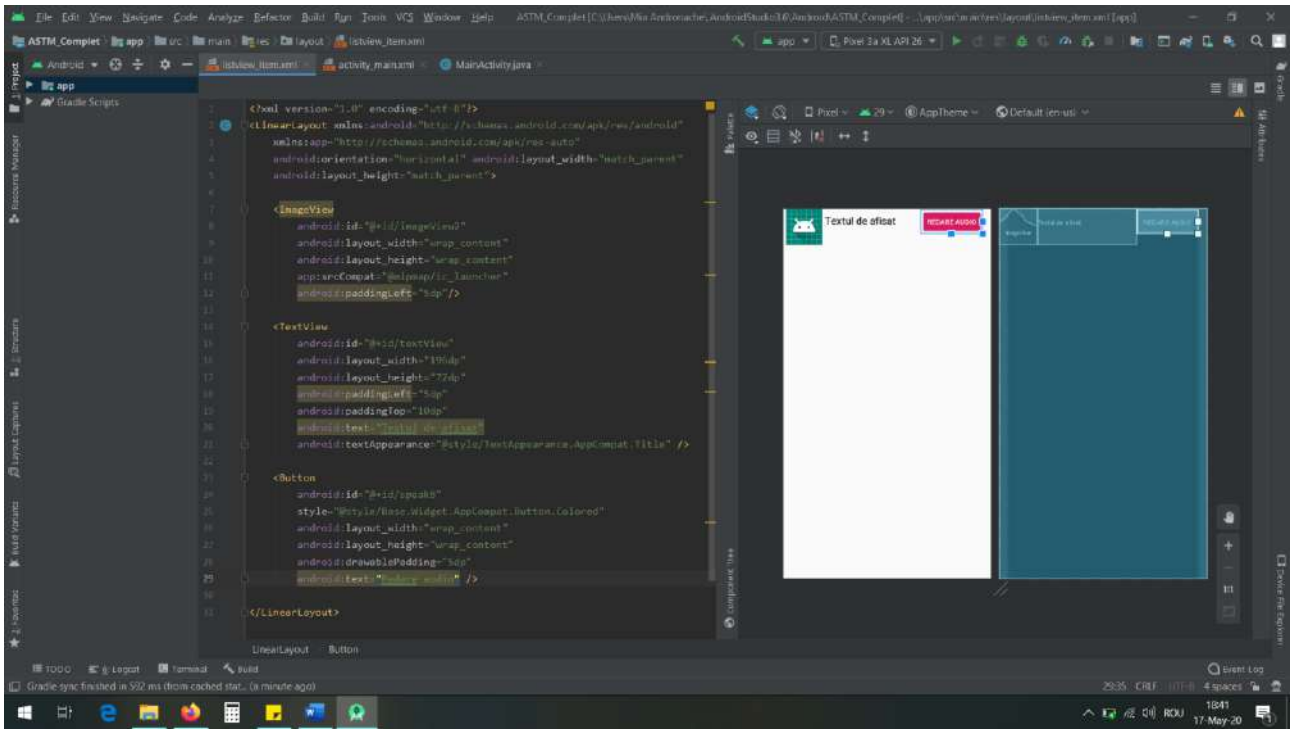


Figura II. 42: Detaliere cod activity_main.xml pentru aplicația completă

Aplicația ce integrează părțile anterioare.

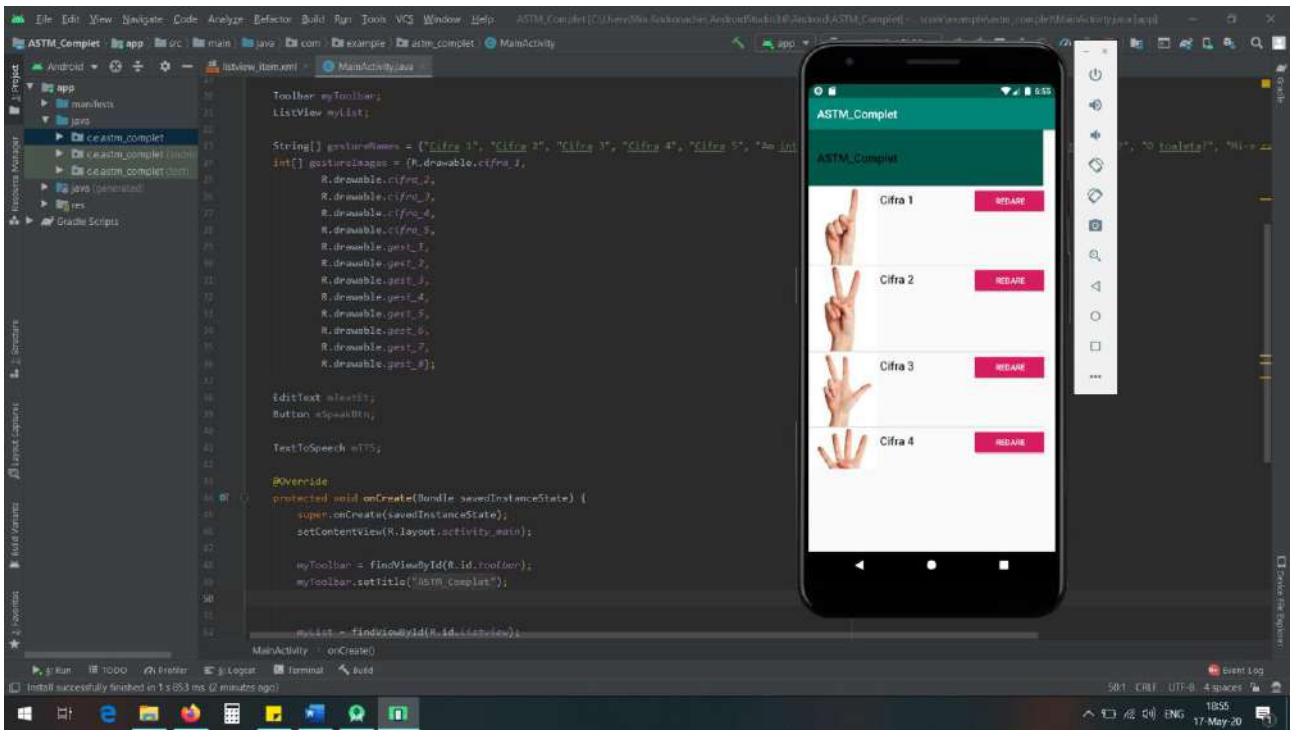


Figura II. 43: Detaliere aplicația completă

CAPITOLUL III

REZULTATE

În cadrul acestui sistem au fost prezentate două moduri diferite de realizare a aceleiași teme. Sistemul a cuprins atât o parte realizată hardware, cât și o parte software.

În cadrul părții hardware [circuit programat prin intermediul mediului de programare Arduino], s-au utilizat diverse componente precum microcontrolere în module integrate, senzori, afișaj electronic sau module de redare audio. Acestea au condus la realizarea a două circuite separate ce comunicau prin intermediul modemului de comunicație emisie-recepție. Trebuie, însă, menționat faptul că toate aceste module sau componente nu au fost utilizate pentru condiții extreme precum condiții de temperatură ridicată, condiții meteo agresive – precipitații abundente sau vânt puternic – sau condiții de umiditate crescută.

Ca și dificultăți întâmpinate în cadrul realizării se pot menționa:

1. Ieșirea analogică extrem de instabilă a senzorului de flexie;
2. Deconectările succesive ale firelor de legătura din breadboard;
3. Necesitatea recalibrării senzorilor pe parcursul funcționării;
4. Necesitatea continuă de adăugare a componentelor adiționale pentru ca modulele să funcționeze la parametrii optimi –> modem emisie-recepție sau modul de redare semnal audio.

În cadrul părții software [aplicație în Android], s-a realizat o aplicație ce integrează trei tab-uri diferite corespunzătoare a trei acțiuni diferite.

Astfel că, în cadrul primei activități, se poate observa o translație din gestul captat cu o cameră fotografică în detecția acestuia pentru ca persoanele ce prezintă dificultăți de vorbire să poată arăta pe un ecran ceea ce și-au dorit să precizeze. Această activitate a fost realizată astfel încât să precizeze și pragul de detecție pentru a putea reliefa modul în care este antrenată rețeaua și cum se realizează

detecția. Se menționează că, în cadrul acestei activități urmează a se aplica și un buton de redare audio a sunetului gestului predefinit.

În cadrul celei de-a doua activități, s-a implementat translația inversă dintre un cuvânt scris de utilizatorul ce nu cunoaște limbajul semnelor și redarea în imagini sau vocală pentru utilizatorul cu dizabilități.

Ulterior, în cadrul celei de-a treia activități, este realizată o precizare cu privire la dezvoltarea ulterioară a proiectului. Astfel, se dorește ca utilizatorul să își poată defini un alt gest decât cele existente în cadrul aplicației. Acest aspect este o provocare deoarece, pentru fiecare gest nou, trebuie apelată baza de date implicită, iar rețeaua trebuie reantrenată.

Ca și dificultăți întâmpinate în cadrul realizării se pot menționa:

1. Captarea imaginii în cadrul primei activități a fost a provocare deoarece, uneori, aplicația se întrerupea;
2. Redarea audio nu este disponibilă în limba română, astfel că sunetele sunt puțin distorsionate;
3. Mișcările camerei fotografice în timpul realizării fotografiei poate duce la probleme mari de claritate;
4. Condițiile de fundal și luminozitate influențează recunoașterea gesturilor astfel că, uneori, precizia gestului recepționat este mai mică;
5. Antrenarea rețelei cu gesturi puține duce la confuzii în detecția corectă;
6. Există extrem de puține modele predefinite pentru gesturile mâinii – majoritatea sunt pentru detecția feței sau a întregului corp

În cadrul următoarelor imagini vor fi reliefate rezultatele obținute:

1. Rezultate cu privire la partea hardware:

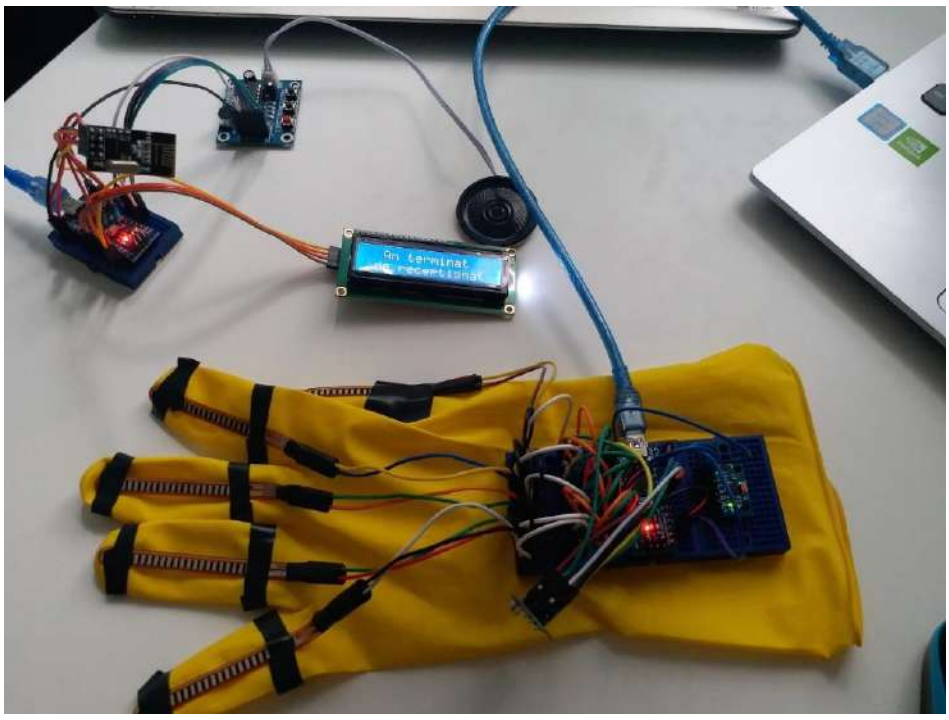


Figura III. 1: Sistemul hardware complet

În cadrul următoarelor imagini se pot observa achizițiile de date ale senzorilor de flexie și ale modulului MPU6050 în momentul în care unul dintre degete este flexat și mesajul declanșat de către fiecare gest:

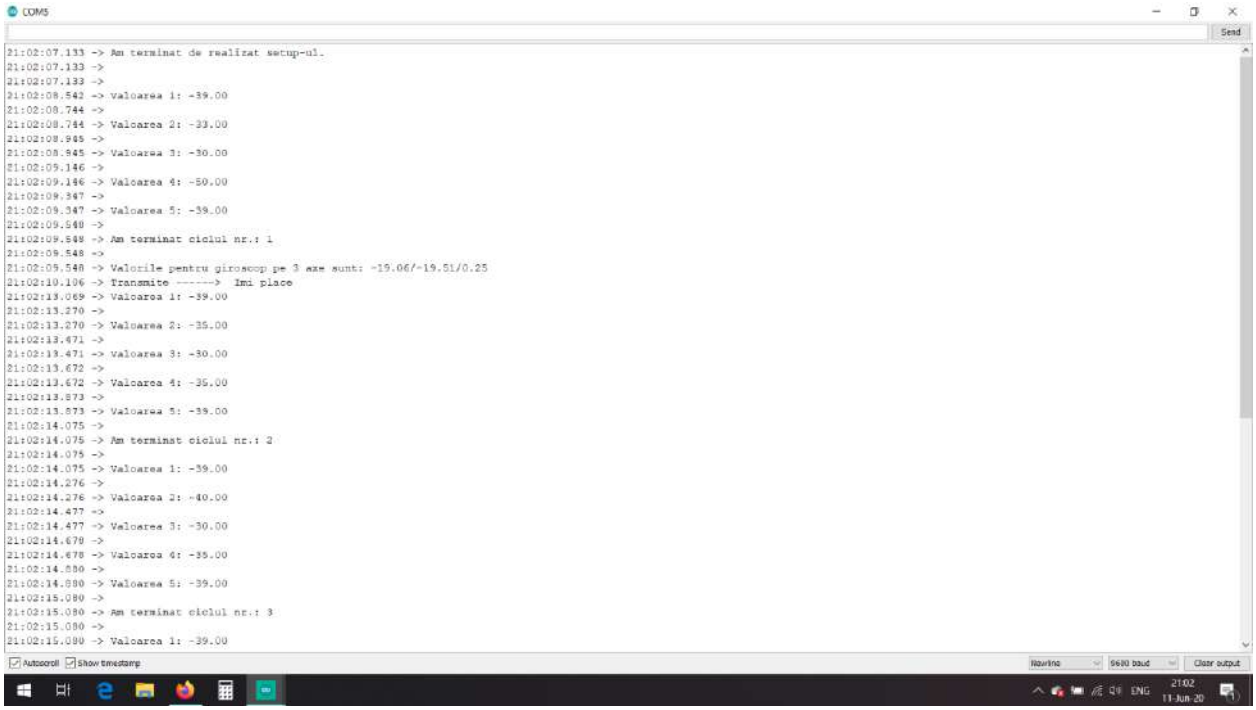


Figura III. 2: Citirea senzorilor de flexie și afișarea mesajelor transmisie – demo1

Figura III. 3: Citirea senzorilor de flexie și afișarea mesajelor transmisie – demo2

În cadrul imaginii de mai jos se poate observa modul în care sunt recepționate mesajele în cadrul Serial Monitor:

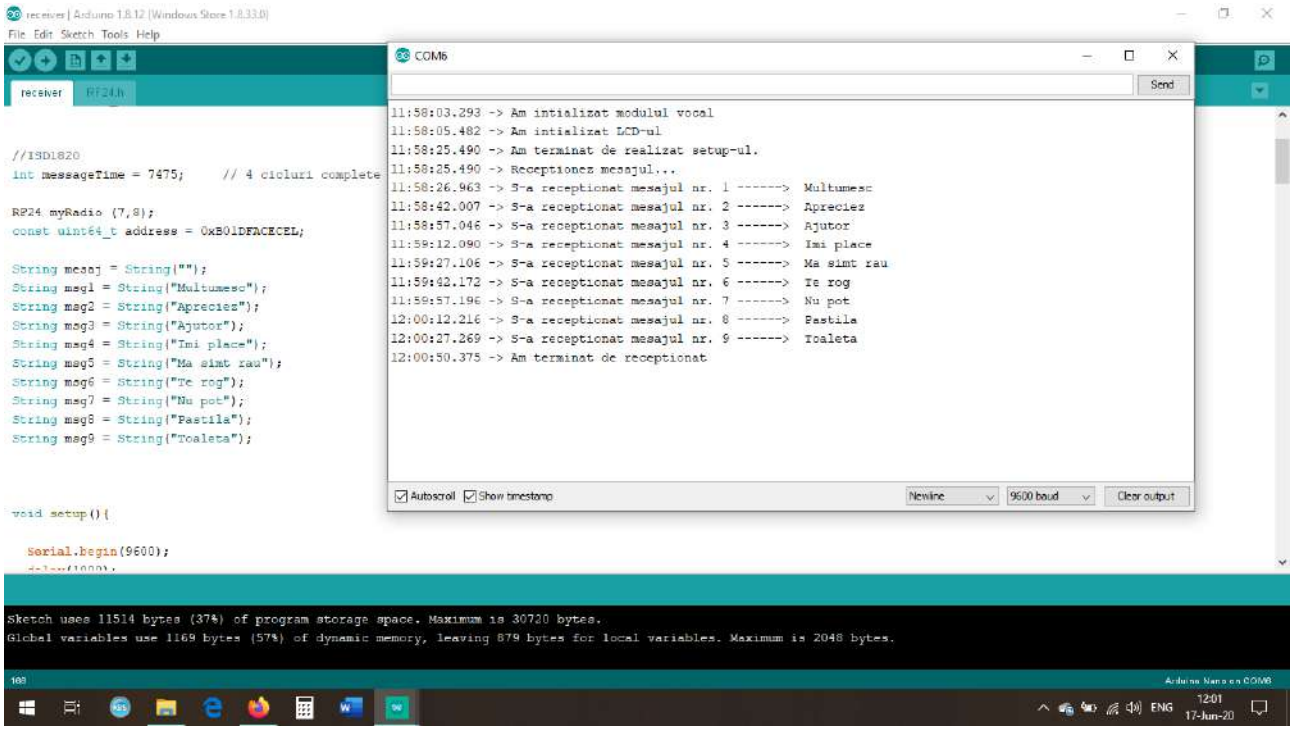


Figura III. 4: Modul în care sunt recepționate mesajele

Comprimând toate datele, va rezulta un tabel de forma:

Senzor de flexie 1	Senzor de flexie 2	Senzor de flexie 3	Senzor de flexie 4	Senzor de flexie 5	Semnificație	Afișaj LCD
Peste -40 grade	0	0	0	0	Mulțumesc!	
0	Peste -40 grade	0	0	0	Apreciez!	

0	0	Peste - 40 grade	0	0	Ajutor!	
0	0	0	Peste - 40 grade	0	Îmi place!	
0	0	0	0	Peste - 40 grade	Mă simt rău!	
Peste - 40 grade	Peste - 40 grade	0	0	0	Te rog!	
0	Peste - 40 grade	Peste - 40 grade	0	0	Nu pot!	
0	0	Peste - 40 grade	Peste - 40 grade	0	Am nevoie de pastilă!	
0	0	0	Peste - 40 grade	Peste - 40 grade	Am nevoie la baie!	

Tabelul III. 1: Tabel cu valorile obținute prin intermediul circuitului

2. Rezultate cu privire la partea software

În cadrul părții realizate prin intermediul aplicației Android, se pot observa translațiile gesturilor.

În cazul imaginii de mai jos se poate observa prima activitate:

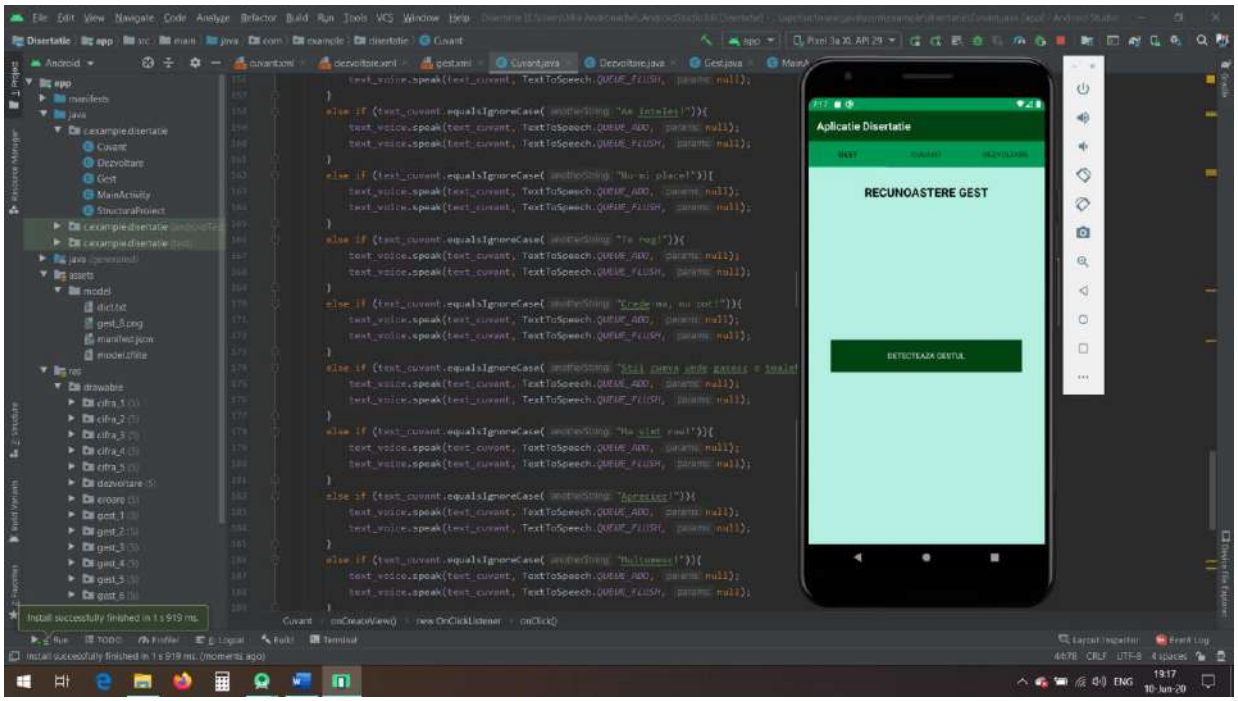


Figura III. 5: Rezultate cu privire la partea software – activitate1

După apăsarea butonului, se deschide camera fotografică, realizându-se imaginea:

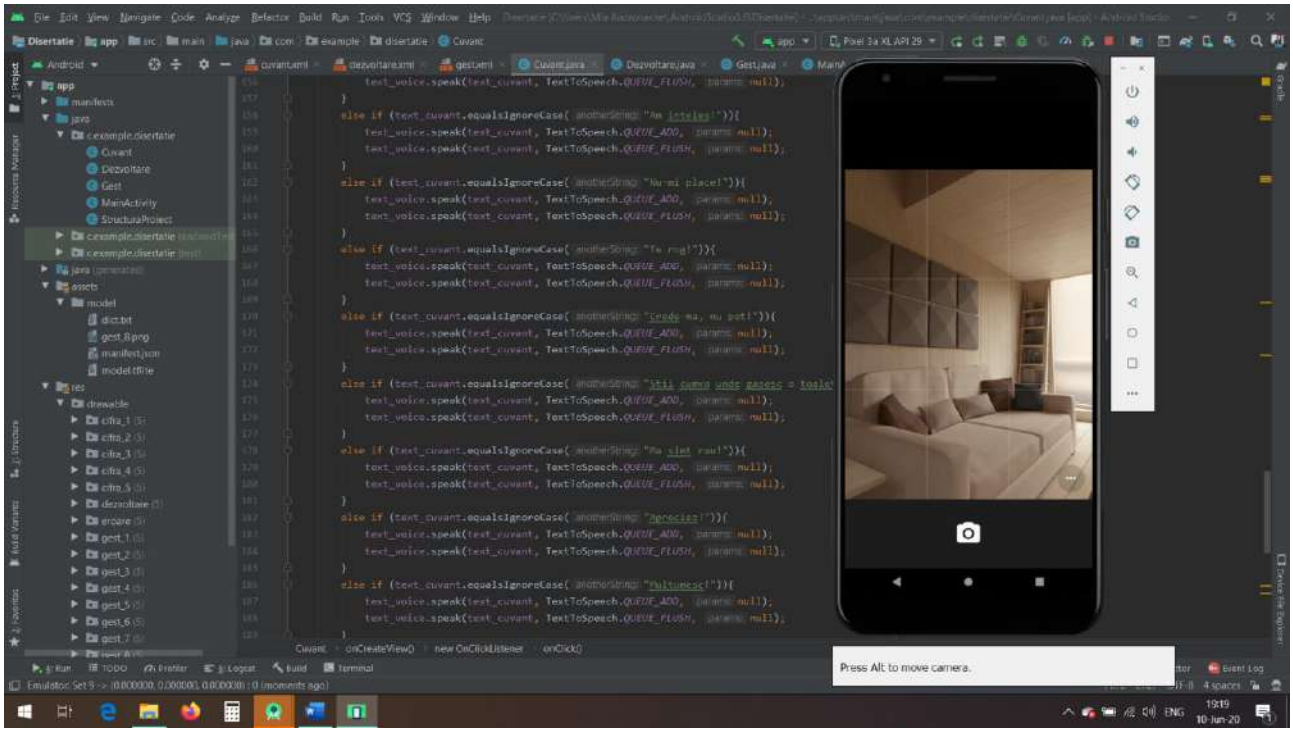


Figura III. 6: Rezultate cu privire la partea software – realizarea fotografiei

Se verifică dacă imaginea este cea dorită:

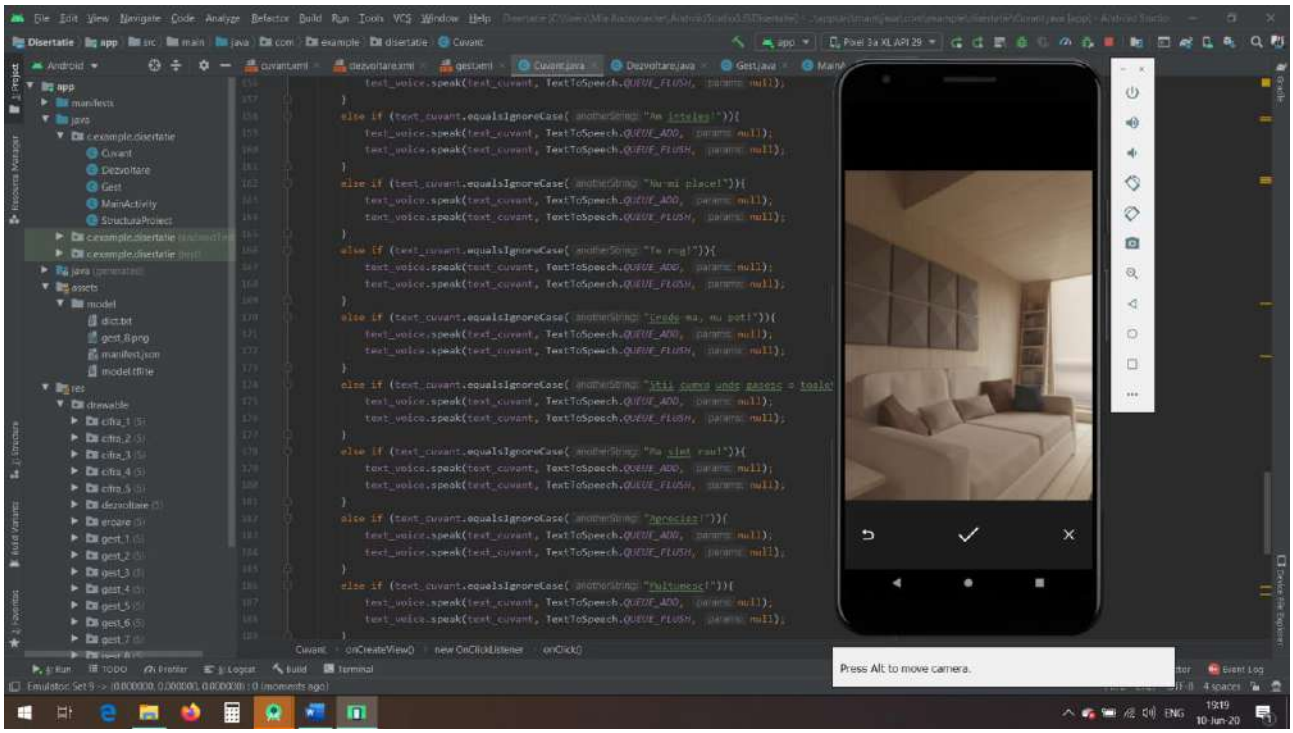


Figura III. 7: Rezultate cu privire la partea software – activitate1 – captarea fotografiei

Se realizează detecția:

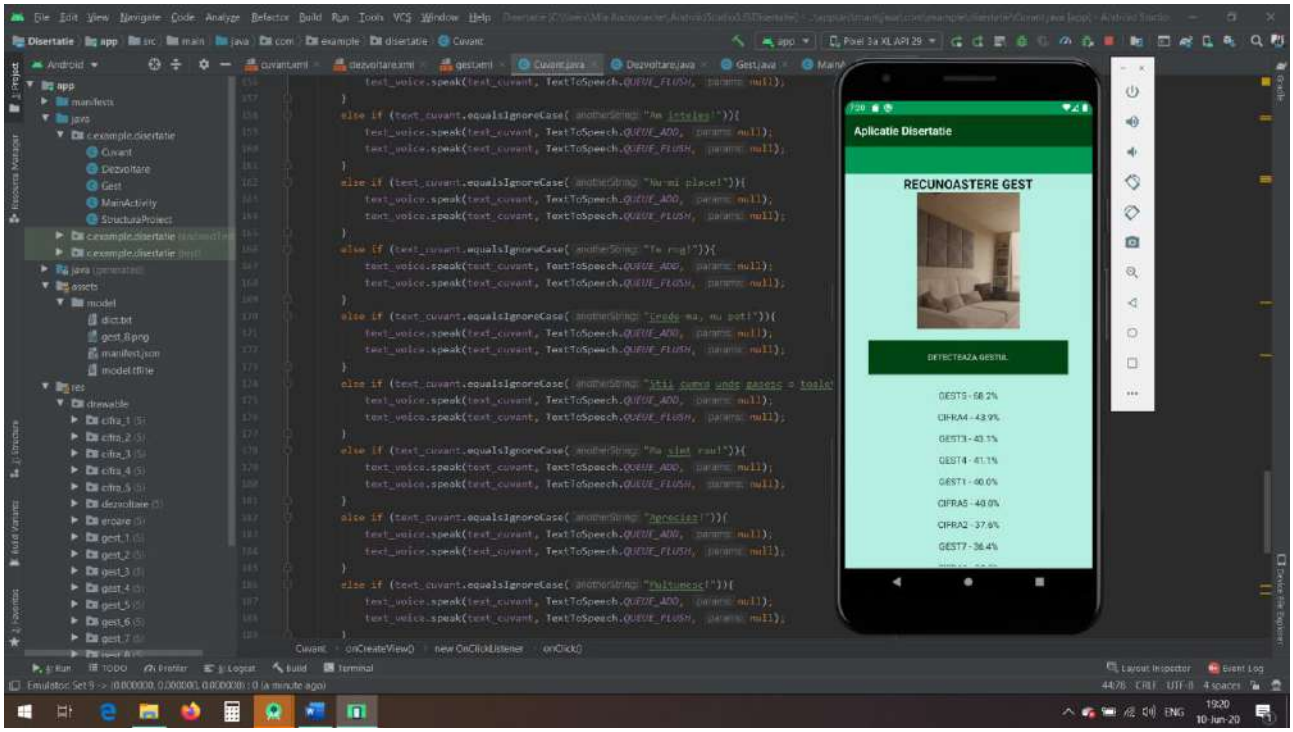


Figura III. 8: Rezultate cu privire la partea software – activitate1 – afișarea recunoașterii și a coeficientului de încredere

În cadrul celei de-a doua activități, se poate observa translația inversă:

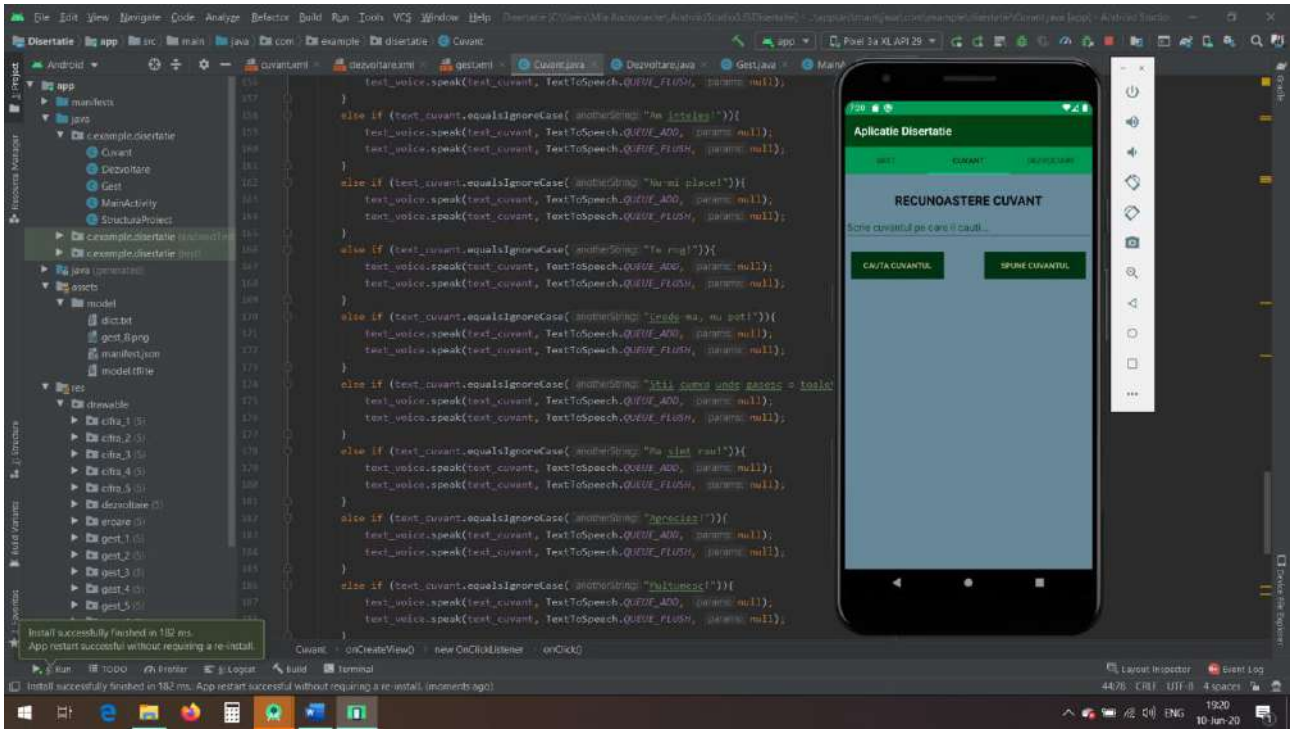


Figura III. 9: Rezultate cu privire la partea software – activitate2

Un exemplu în cazul translației inverse:

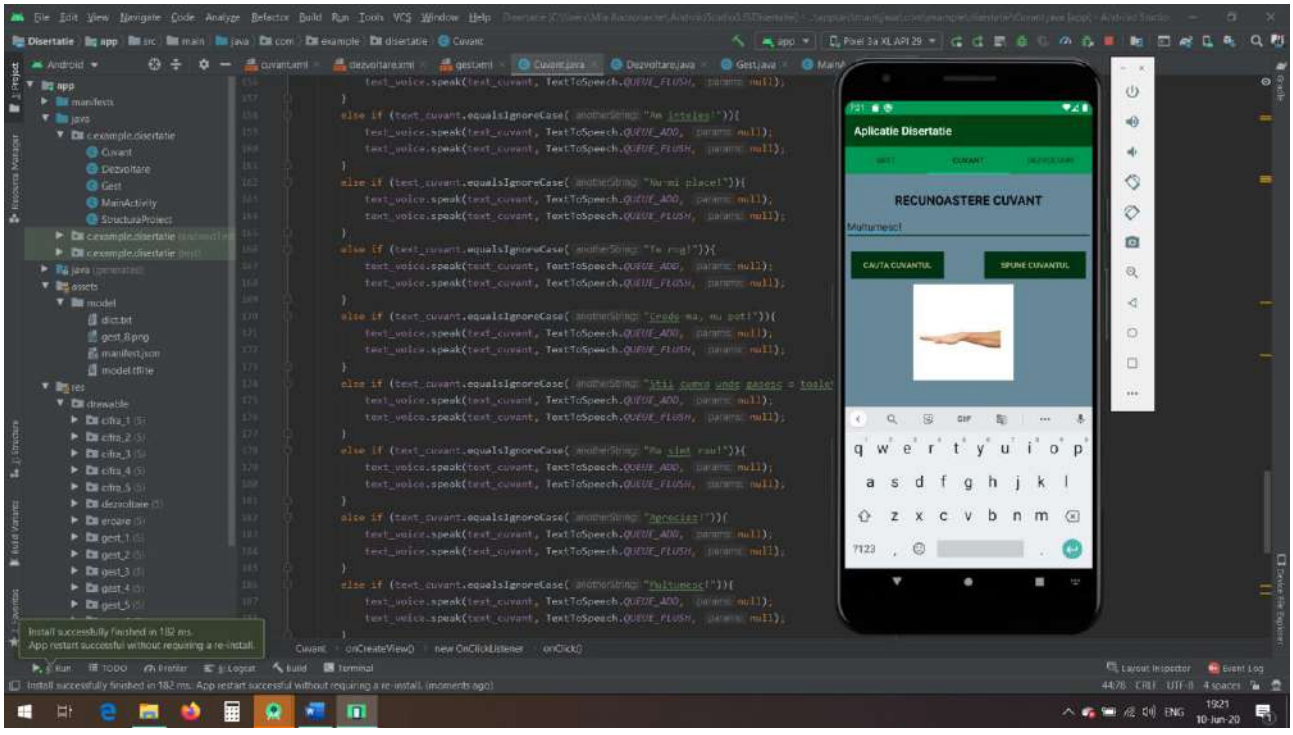


Figura III. 10: Rezultate cu privire la partea software – activitate2 – detecția unui cuvânt prin fotografie și semnal audio – demo1

Alt exemplu:

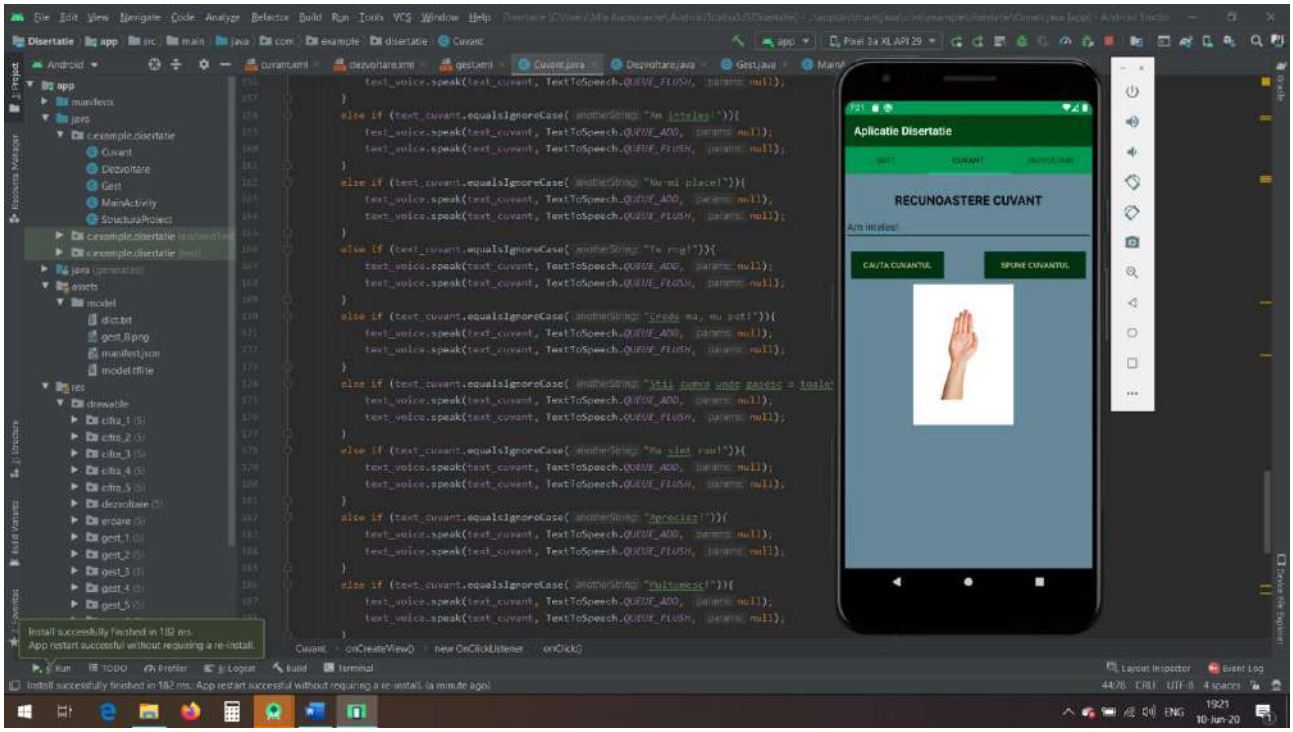


Figura III. 11: Rezultate cu privire la partea software – activitate2 – detecția unui cuvânt prin fotografie și semnal audio – demo2

A treia activitate:

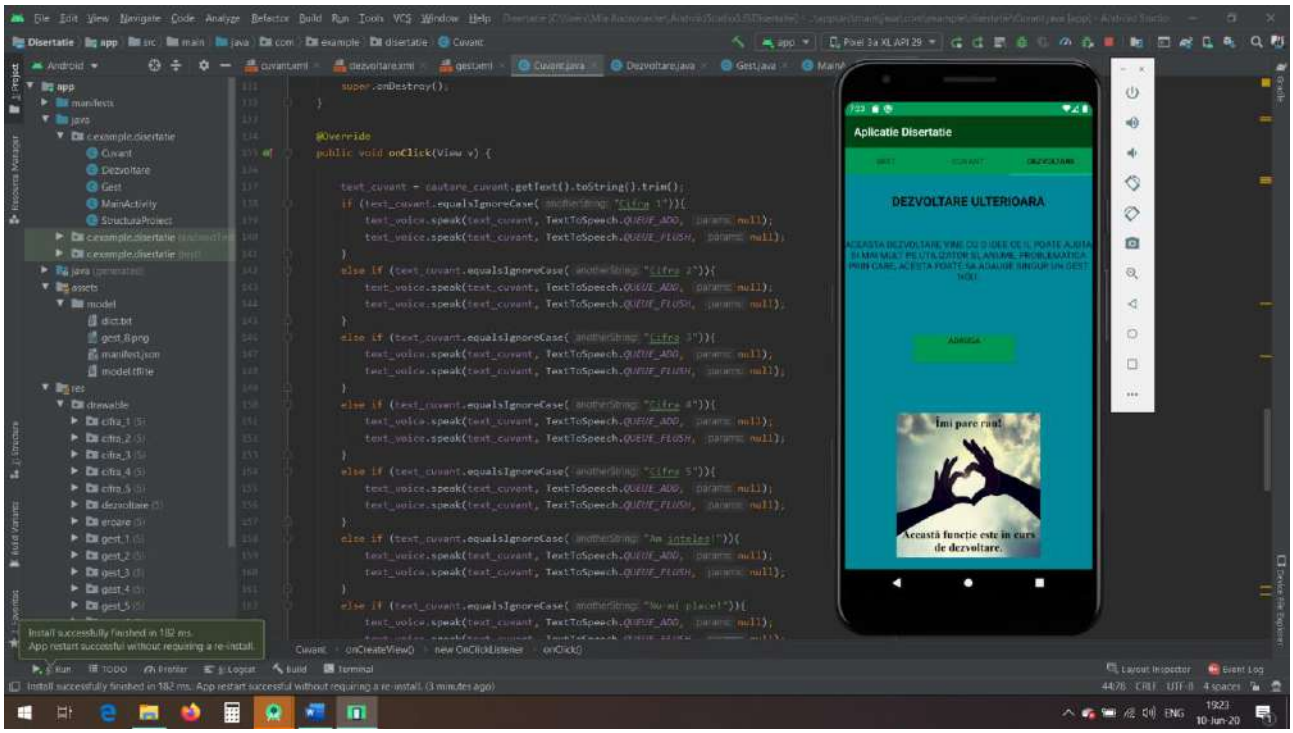


Figura III. 12: Rezultate cu privire la partea software – activitate3 – dezvoltare ulterioară

3. Comparație

Din pricina faptului că proiectul implementat deține două direcții diferite, se poate realiza și o comparație a celor două metode precizând avantaje și dezavantaje:

1. Sistemul hardware [în Arduino] recunoaște mai puține gesturi predefinite decât cel software [în Android];
2. Sistemul în Android are rapiditate mai mare de recunoaștere a gesturilor față de cel în Arduino;
3. Dezavantajul ambelor modele constă în gesturile asemănătoare ce, uneori, nu sunt identificate corect;
4. Sistemul hardware este incomod datorită greutății componentelor hardware și a necesității de a fi purtat împreună cu utilizatorul;
5. Ambele sisteme presupun ca utilizatorul primar – cel cu dizabilități, să cunoască toată baza de date pentru a ști ce gest trebuie să utilizeze;
6. Uneori, datorită orientării mâinii, sistemele nu pot realiza o detecție corectă;
7. În cazul sistemului în Android, la condiții de iluminare slabă, detecția este puternic afectată;
8. În cazul sistemului hardware, eficacitatea sistemului depinde de capabilitățile componentelor – dacă una dintre componente este defectă, sistemul poate să nu mai funcționeze sau să aibă erori mari;
9. Sistemul hardware impune un cost suplimentar pentru achiziția componentelor;
10. Ambele sisteme produc rezultate stabile și pot ajuta la stabilirea unei comunicații;
11. Dacă, în cazul sistemului hardware, senzorii și modulul MPU6050 ar fi implementate în interiorul mănușilor, ar scădea impactul problemelor datorate de condițiile meteo, crescând, astfel, eficacitatea;

În concluzie, ambele sisteme prezintă atât avantaje, cât și dezavantaje, iar alegerea cea mai potrivită trebuie să fie realizată de către utilizatorul primar în funcție de nevoile sale, acest aspect fiind unul subiectiv.

CONCLUZII

Limbajul semnelor este unul dintre instrumentele utile pentru a ușura comunicarea între persoanele ce prezintă dizabilități de auz sau vorbire și societatea normală. Deși limbajul semnelor poate fi implementat pentru a comunica, ambele persoane trebuie să îl cunoască, fapt ce nu este posibil întotdeauna. Prin urmare, proiectul implementat încearcă să scape de astfel de bariere prin două direcții diferite de utilizare: hardware – prin intermediul unui circuit ce conține senzori de flexie și prin intermediul unei aplicații Android – pentru a ușura detecția senzorilor fără o unitate hardware atașată (se presupune că orice persoană poate deține un telefon).

Prin această lucrare, a fost proiectat un circuit aplicabil transpunerii gesturilor aferente limbajului semnelor în cuvinte de tip text sau semnale vocale. Astfel, s-au utilizat senzori de flexie și un modul de tip MPU6050. Acesta din urmă, a fost folosit pentru a îmbunătăți precizia prin mișcările date de încheietura mâinii. Etapele de procesare au fost efectuate cu ajutorul unor microcontrolere transpuse pe ambele plăci implicate, iar semnalele de ieșire ce au fost modelate și schimbate între module wireless, au fost procesate pentru a fi de tip semnal audio și text.

Sistemul poate fi extins pentru un număr mult mai mare de gesturi și moduri de implementare diferite sau poate fi aplicabil în domeniul animalelor (în special partea hardware). Momentan, eficiența poate fi sporită prin creșterea numărului de gesturi și cuvinte pe care le poate folosi utilizatorul, dar se pot exploata diverse tehnici de programare noi, care să aibă același scop, dat fiind faptul că se folosesc și algoritmi de ML.

Acest proiect mi-a oferit un mod de expunere în domeniul proiectării sistemelor bazate pe microcontroler și în domeniul aplicațiilor Android.

Avantajele proiectului sunt:

1. Întârzieri relativ mici, timpul de răspuns fiind foarte bun mai ales în cazul abordării software ce conține aplicația Android.

2. Sistemul ce conține aplicația Android poate fi adaptat pentru a permite intervenția utilizatorului în adăugarea unui gest nou.
3. Aplicația Android poate fi protejată prin introducerea unui tab de autentificare pe bază de amprentă a utilizatorului.
4. Costul este relativ scăzut – prețul fiind dat doar de componentele hardware.
5. Folosește sisteme wireless – subiect adaptabil cu noile schimbări în acest domeniu
6. Sistemul hardware este ușor de manipulat, existând două plăci diferite, sistemul păstrându-și acuratețea.
7. Numărul de gesturi poate fi mărit în cazul ambelor sisteme implementate.
8. Folosindu-se procesarea imaginilor, sistemul poate fi extins în sisteme antifurt sau în sisteme ce realizează detecții ale fețelor.
9. Pentru dezvoltări ulterioare, s-ar putea utiliza țesături conductoare în locul senzorilor de flexie pentru a permite realizarea unui sistem de o greutate mai mică.

DEZVOLTĂRI ULTERIOARE

Sistemul dezvoltat în cadrul lucrării a avut ca scop funcționalitatea, având rezultate favorabile. Astfel, acesta și-a îndeplinit scopul, însă, ca orice sistem, poate avea și îmbunătățiri.

Ca o primă direcție de dezvoltare, ar putea fi considerat aspectul legat de estetică. Astfel, ar putea fi îmbunătățit atât sistemul în Android – prin intermediul unor pictograme și animații pentru copii, cât și sistemul hardware prin intermediul unor unui PCB – există în anexe datele specifice acestui sistem ce conține PCB bazat pe microcontroler de tip PIC – sau prin intermediul unor fire conductoare.

O altă direcție de dezvoltare ar putea fi considerată cea legată de capabilitățile sistemului de emisie-recepție, astfel că, se dorește încercarea și a altor tipuri de module de comunicații de tip Wi-Fi sau ZigBee.

A treia direcție de dezvoltare este legată de aplicația Android. Se dorește reantrenarea rețelei cu mai multe gesturi și găsirea unei soluții pentru automatizarea acestui principiu pentru ca utilizatorul să își poată adăuga gesturile noi fără ajutor suplimentar.

O altă direcție de dezvoltare ar putea fi considerată cea de testare efectivă a soluției de către persoane cu dizabilități pentru a putea avea un feedback real.

BIBLIOGRAFIE

1. WHO, 2011. World report on disability. [online] Disponibil în cadrul link-ului: www.who.int/disabilities/world_report/2011/en. Accesat în Aug. 2019
2. Ministry of Labour, Family, Social Assistance and Elderly, 2014. [online] Disponibil în cadrul link-ului: http://www.mmuncii.ro/j33/index.php/ ro/2014-domenii/munca/programe-si-strategii. Accesat în Aug. 2019
3. International Medical Society, Anghelescu Aurelian, Bușcă Mihai, Constantin Adina, Andone Ioana, Anghelescu Lucia AnaMaria, Onose Gelu, ,,Employment of People with Disabilities in Romania", Vol. 9 No. 365 doi: 10.3823/2236, 2016, Disponibil în cadrul www.intarchmed.com și www.medbrary.com, Accesat în Aug. 2019
4. Flexion, Sensor.wiki, https://sensorwiki.org/sensors/flexion Accesat în Aug. 2019
5. Zimmerman T. G. 1985. Optical flex sensor. US 4 542 291, Accesat în Aug. 2019
6. A Review on Applications of Flex Sensors, Sreejan Alapati, Shivraj Yeole [online] Disponibil în cadrul link-ului: https://www.researchgate.net/publication/318850816_A_Review_on_Applications_of_Flex_Sensors Accesat în Sep. 2019
7. Senzori, Florin Mihai, [online] Disponibil în cadrul link-ului: https://www.scribd.com/document/58494725/Senzori Accesat în Sep. 2019
8. Senzori şi traductoare pentru aplicaţii de automatizare, http://electronica-azi.ro/2004/02/05/senzori-si-traductoare-pentru-aplicatii-de-automatizare-2/ Accesat în Sep. 2019
9. Senzori în logistica industrială, http://www.msdi.ro/senzori-in-logisticaindustriala Accesat în Sep. 2019
10. Ce este şi cum funcţionează senzorul Accelerometru?, https://www.wellcome.ro/ce-este-si-cum-functioneaza-senzorul-accelerometru, Accesat în Oct. 2019

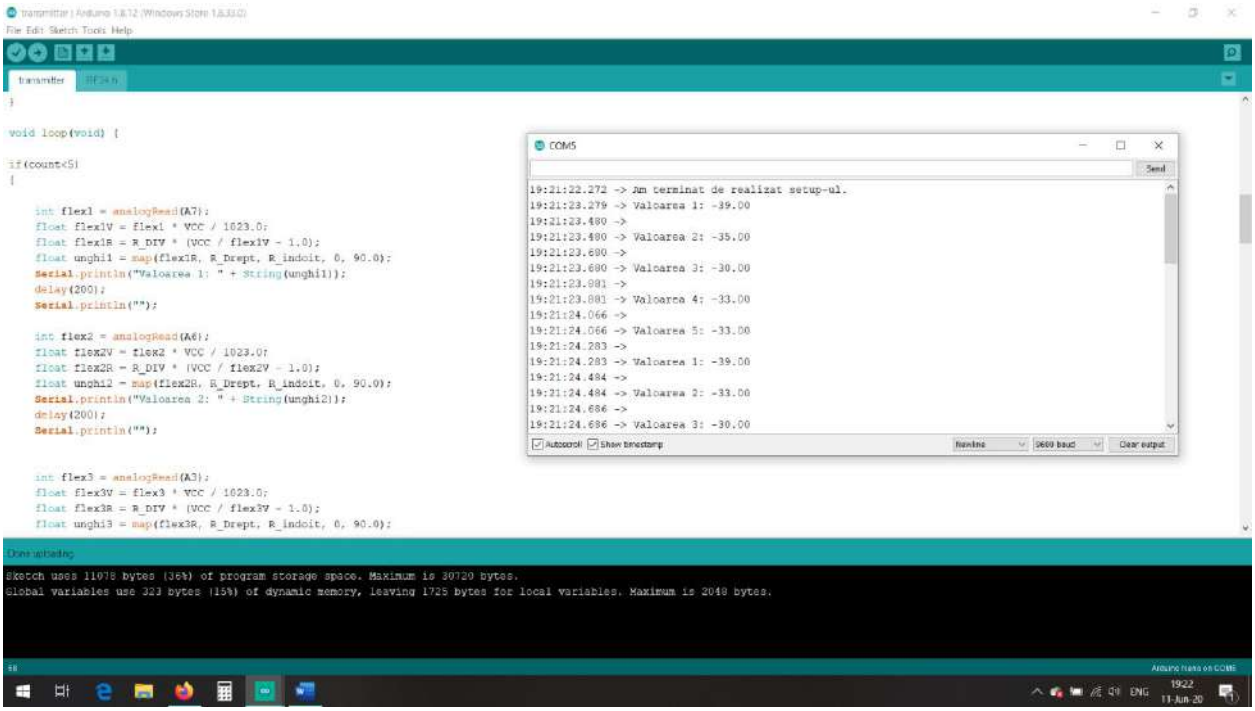
11. Gyroscope Technology and Applications: A Review in the Industrial Perspective, Vittorio M. N. Passaro, Antonello Cuccovillo, Lorenzo Vaiani, Martino De Carlo and Carlo Edoardo Campanella [online]. Disponibil în cadrul link-ului: www.mdpi.com/1424-8220/17/10/2284/pdf ,Accesat în Oct. 2019
12. Accelerometru, http://www.sensorwiki.org/doku.php/sensors/accelerometer , Accesat în Oct. 2019
13. Giroscop, https://biblioteca.regielive.ro/referate/mecanica/prezentare-giroscop-185723.html, Accesat în Oct. 2019
14. Curs de Fizică varianta scrisă, profesor E. Niculescu, 2014
15. Display-uri grafice LCD, http://electronica-azi.ro/2007/05/24/display-uri-grafice-lcd/ , Accesat în Oct. 2019
16. Curs microcontrolere, http://tet.pub.ro/pages/Microprocesoare2/curs_microcontrolere_P.pdf Accesat în Oct. 2017
17. Watchdog timer, https://en.wikipedia.org/wiki/Watchdog_timer Accesat în Oct. 2017
18. ESP8266EX, https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf Accesat în Oct. 2017
19. ISD1820, https://www.alldatasheet.com/view.jsp?Searchword=ISD1820 Accesat în Nov. 2019
20. Regulator de tensiune LT1129/LT1129-3.3/LT1129-5, http://www.analog.com/media/en/technical-documentation/data-sheets/112935ff.pdf Accesat în Oct. 2017
21. Regulator de tensiune LM3940, http://www.ti.com/lit/ds/symlink/lm3940.pdf, Accesat în Oct. 2017
22. Modul de înregistrare IS1820, https://www.allelectronics.com/mas_assets/media/allelectronics2018/spec/ME-63.pdf, Accesat în Dec. 2019
23. Cum pot recunoaște codul de bare cu kit-ul ML firebase? https://stackoverflow.com/questions/62215579/how-can-i-recognize-barcode-with-firebase-ml-kit, Accesat în Ian. 2020
24. Etichetați imagini cu kit-ul ML pe Android, https://firebase.google.com/docs/ml-kit/android/label-images Accesat în Ian. 2020
25. Modul RF nRF24L01, https://components101.com/wireless/nrf24l01-pinout-features-datasheet Accesat în Ian. 2020
26. Ce este NoSQL? https://www.mongodb.com/nosql-explained Accesat în Ian. 2020
27. Noțiuni introductive despre Arduino: Platforma de prototip electronic open source, Massimo Banzi, Michael Shiloh, Maker Media, [online] Disponibil în cadrul link-ului: https://books.google.ro/books?id=Xd3SBQAAQBAJ&printsec=frontcover&dq=arduino+book&hl=ro&sa=X&ved=0ahUKEwjKrqaY94vqAhVQ_SoKHcKLD54Q6AEIKDAA#v=onepage&q=arduino%20book&f=false Accesat în Apr. 2020
28. *Arduino Nano Un ghid pentru începători*, Agus Kurniawan, [online] Disponibil în cadrul link-ului: https://books.google.ro/books?id=pfiaDwAAQBAJ&printsec=frontcover&dq=arduino+book&hl=ro&sa=X&ved=0ahUKEwjKrqaY94vqAhVQ_SoKHcKLD54Q6AEINDAB#v=onepage&q=arduino%20book&f=false Accesat în Apr. 2020
29. *SamsTeachYourself: Android Application Development*, Lauren Darcey Shane Conder, Lauren Darcey, Shane Conder, U.S. Corporate and Government Sales

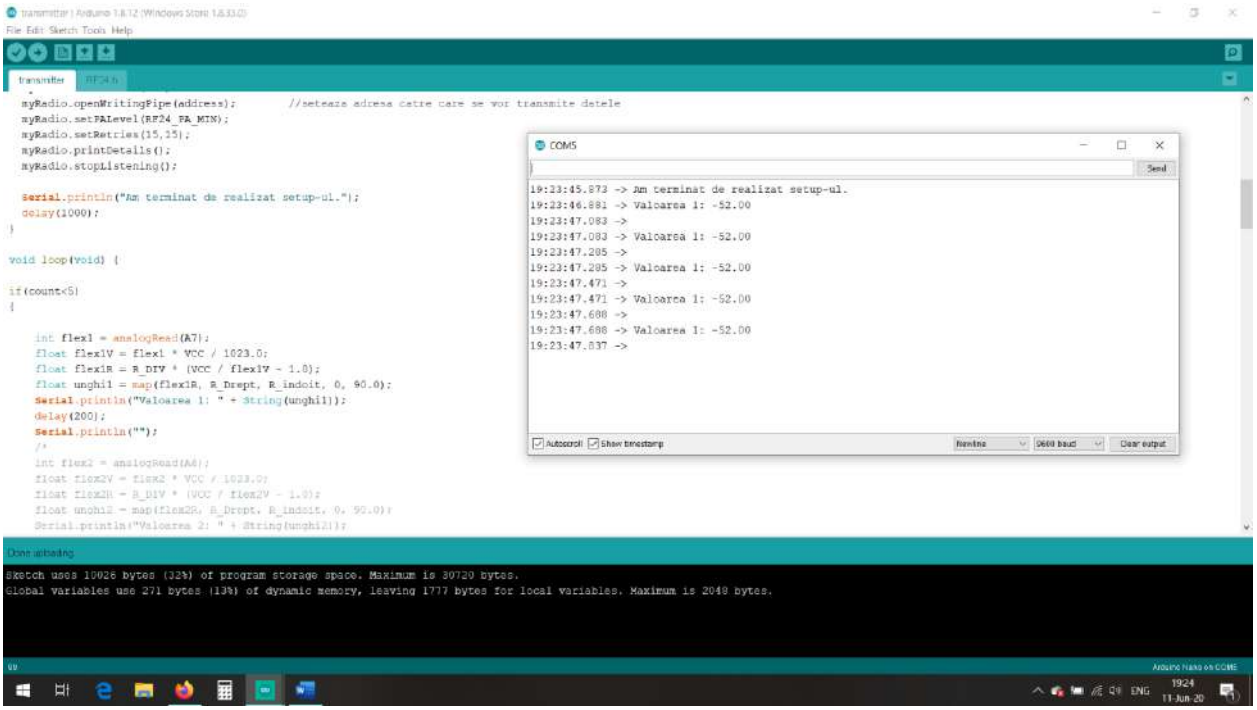
30. *Curs microcontrolere*, [online] Disponibil în cadrul link-ului: http://read.pudn.com/downloads166/sourcecode/editor/762521/Curs%20Microcontrollere.doc
31. *Hobby?,* Clubul Copiilor Petroșani, [online] Disponibil în cadrul link-ului: http://yo2kqk.kovacsfam.ro/revista/HOBBY.20.pdf
32. *Modul înregistrare voce ISD1820 și difuzor,* [online] Disponibil în cadrul link-ului: https://www.hobbymarket.ro/module-atasabile-c-2/modul-inregistrare-voce-isd1820-si-difuzor-p-363.html

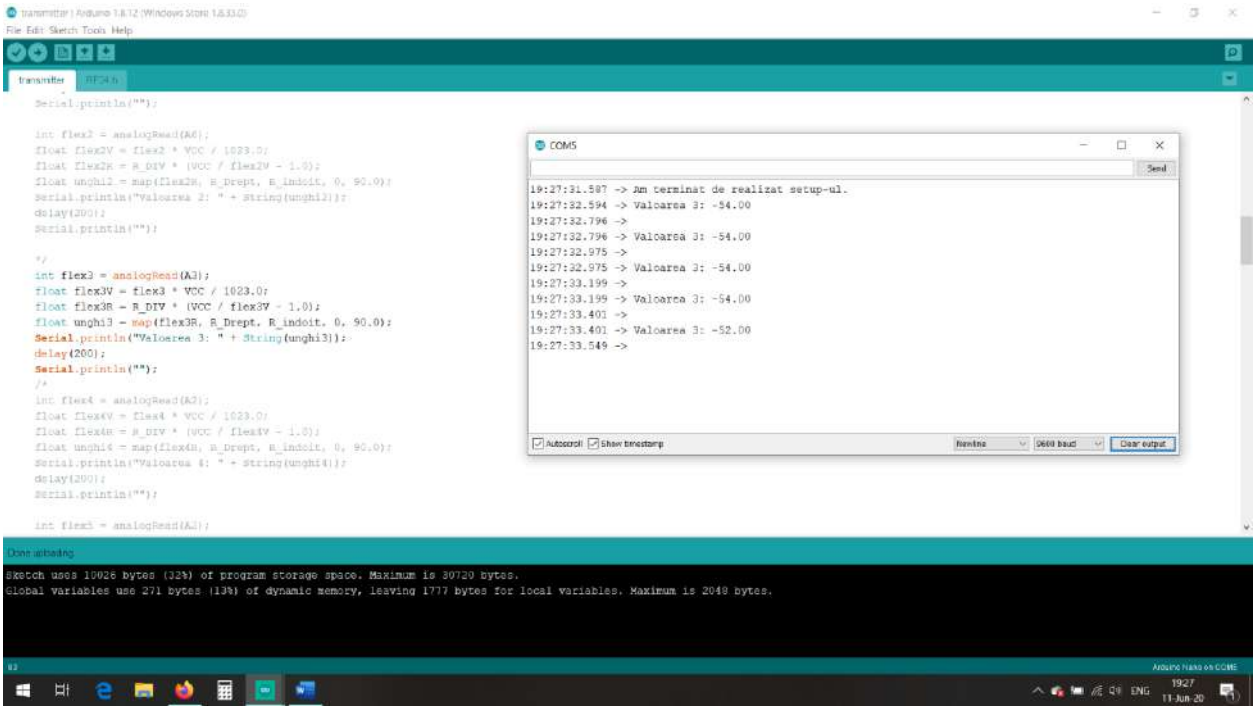
ANEXA 1

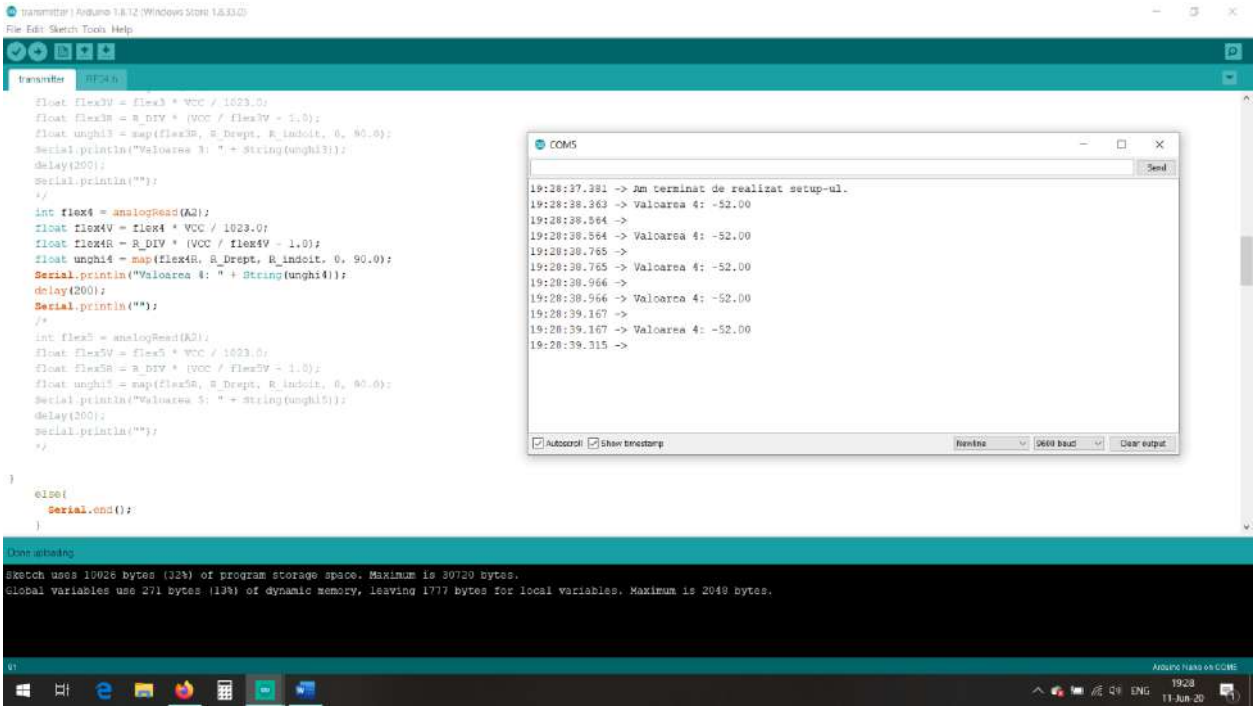
Măsurători efectuate de la senzori

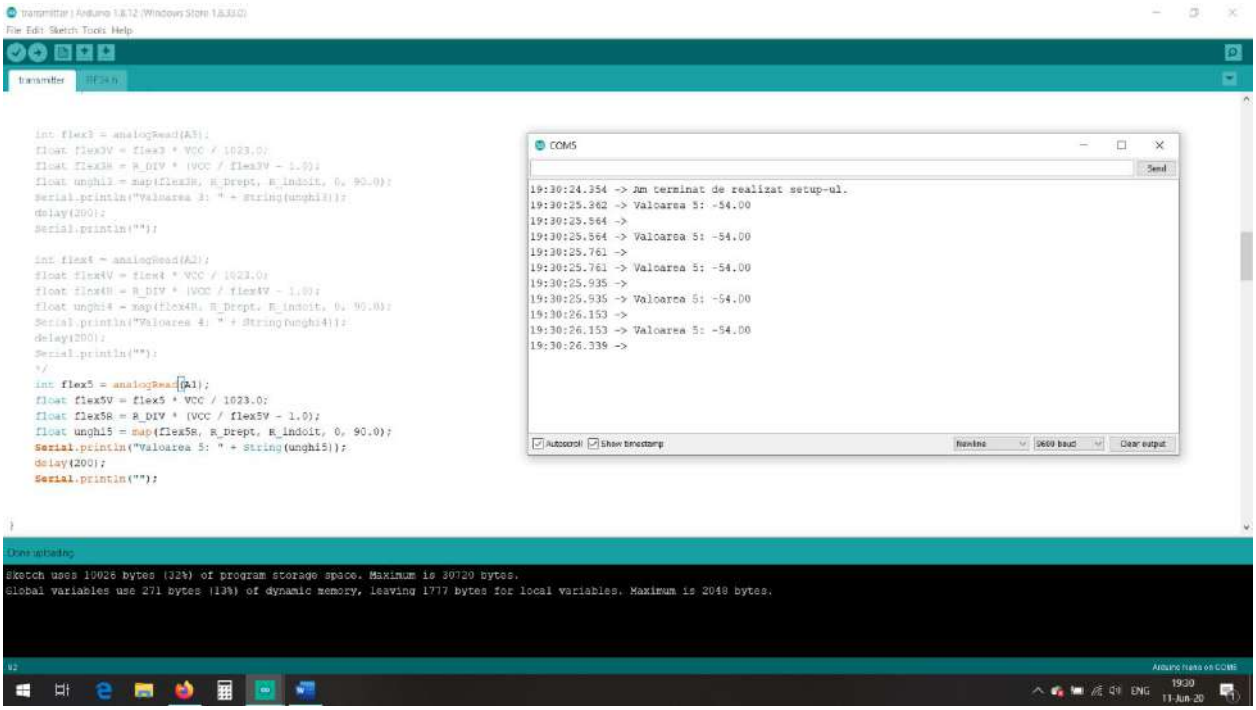
1. Senzori de flexie



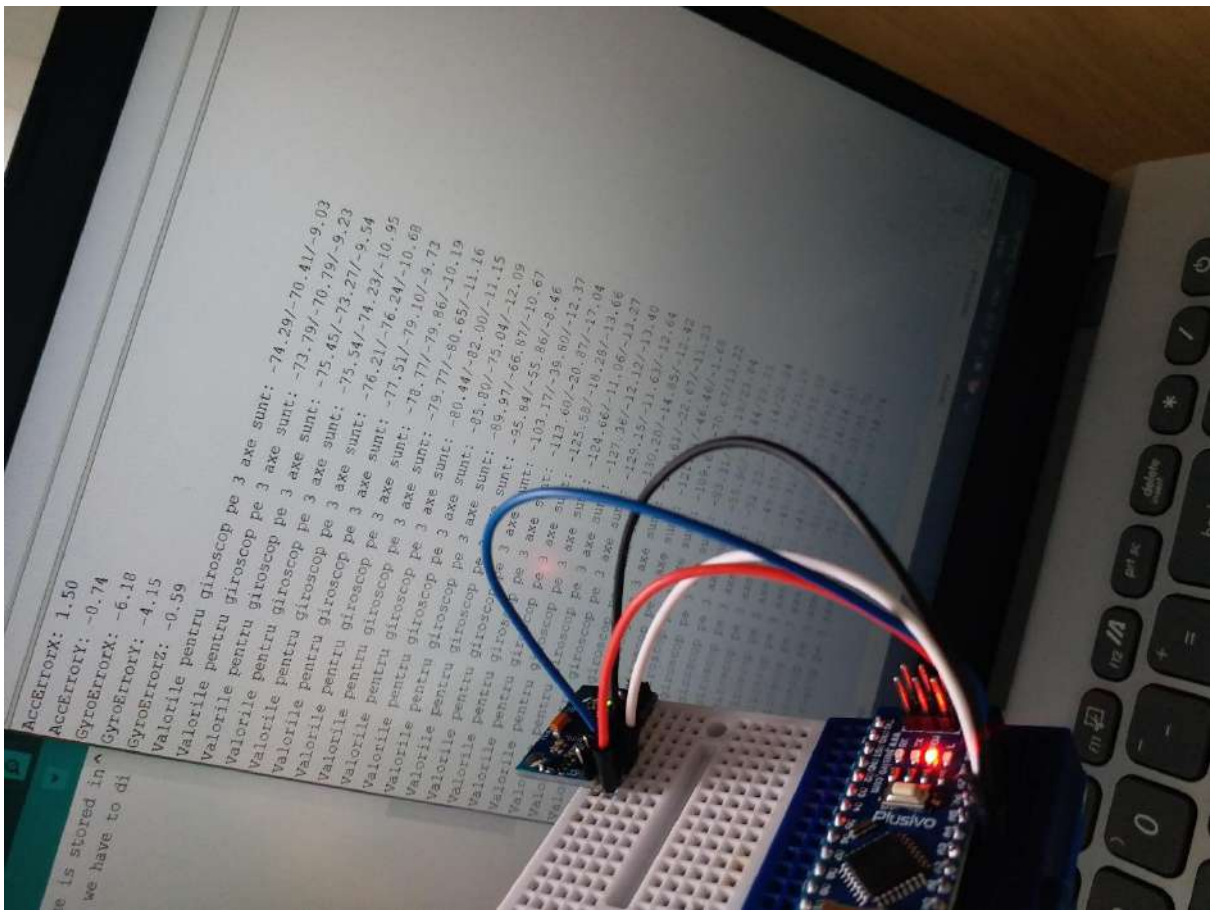


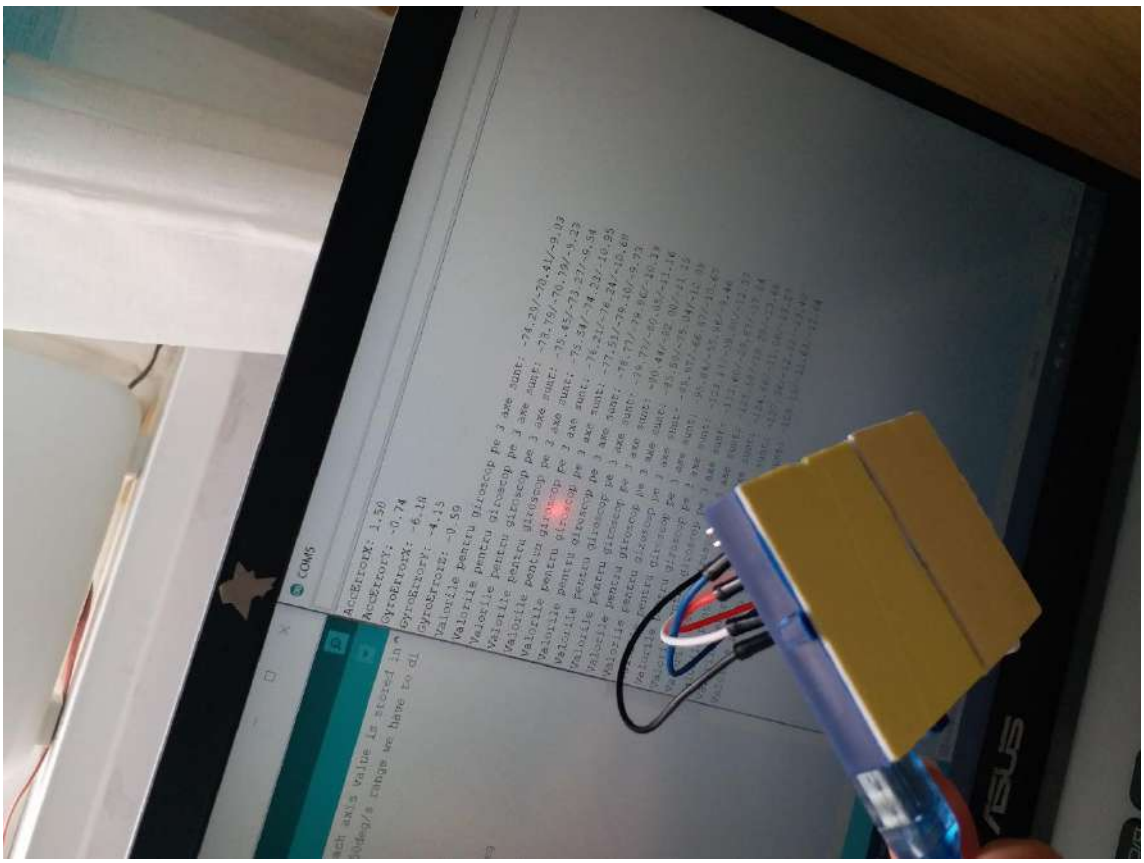

transmitter | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
transmitter
RF24.h
int flex3 = analogRead(A3);
float flex3V = flex3 * VCC / 1023.0;
float flex3R = R_DIV * (VCC / flex3V - 1.0);
float unghi3 = map(flex3R, R_Drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 3: " + String(unghi3));
delay(200);
Serial.println("");
int flex4 = analogRead(A2);
float flex4V = flex4 * VCC / 1023.0;
float flex4R = R_DIV * (VCC / flex4V - 1.0);
float unghi4 = map(flex4R, R_Drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 4: " + String(unghi4));
delay(200);
Serial.println("");
int flex5 = analogRead(A1);
COM5
Send
19:27:31.587 -> Am terminat de realizat setup-ul.
19:27:32.594 -> Valoarea 3: -54.00
19:27:32.796 ->
19:27:32.796 -> Valoarea 3: -54.00
19:27:32.975 ->
19:27:32.975 -> Valoarea 3: -54.00
19:27:33.199 ->
19:27:33.199 -> Valoarea 3: -54.00
19:27:33.401 ->
19:27:33.401 -> Valoarea 3: -52.00
19:27:33.549 ->
Autoscroll
Show timestamp
Newline
9600 baud
Clear output
Done uploading.
Sketch uses 10026 bytes (32%) of program storage space. Maximum is 30720 bytes.
Global variables use 271 bytes (13%) of dynamic memory, leaving 1777 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano on COM5
ENG
19:27
11-Jun-20

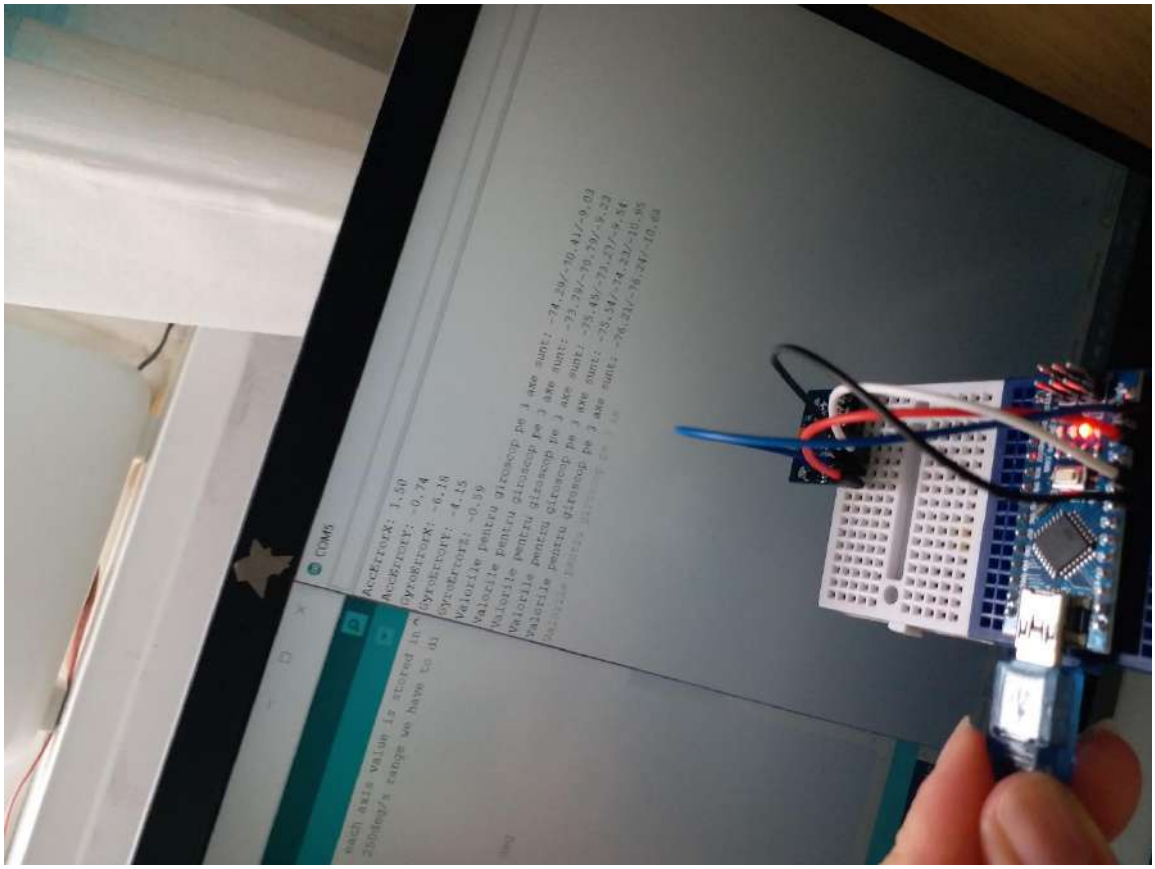

transmitter | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
transmitter
RF24.h
float flex3V = flex3 * VCC / 1023.0;
float flex3R = R_DIV * (VCC / flex3V - 1.0);
float unghi3 = map(flex3R, R_Drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 3: " + String(unghi3));
delay(200);
Serial.println("");
int flex4 = analogRead(A2);
float flex4V = flex4 * VCC / 1023.0;
float flex4R = R_DIV * (VCC / flex4V - 1.0);
float unghi4 = map(flex4R, R_Drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 4: " + String(unghi4));
delay(200);
Serial.println("");
int flex5 = analogRead(A1);
float flex5V = flex5 * VCC / 1023.0;
float flex5R = R_DIV * (VCC / flex5V - 1.0);
float unghi5 = map(flex5R, R_Drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 5: " + String(unghi5));
delay(200);
Serial.println("");
}
else{
Serial.end();
}
COM5
Send
19:28:37.381 -> Am terminat de realizat setup-ul.
19:28:38.363 -> Valoarea 4: -52.00
19:28:38.564 ->
19:28:38.564 -> Valoarea 4: -52.00
19:28:38.765 ->
19:28:38.765 -> Valoarea 4: -52.00
19:28:38.966 ->
19:28:38.966 -> Valoarea 4: -52.00
19:28:39.167 ->
19:28:39.167 -> Valoarea 4: -52.00
19:28:39.315 ->
Autoscroll
Show timestamp
Newline
9600 baud
Clear output
Done uploading.
Sketch uses 10026 bytes (32%) of program storage space. Maximum is 30720 bytes.
Global variables use 271 bytes (13%) of dynamic memory, leaving 1777 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano on COM5
ENG
19:28
11-Jun-20

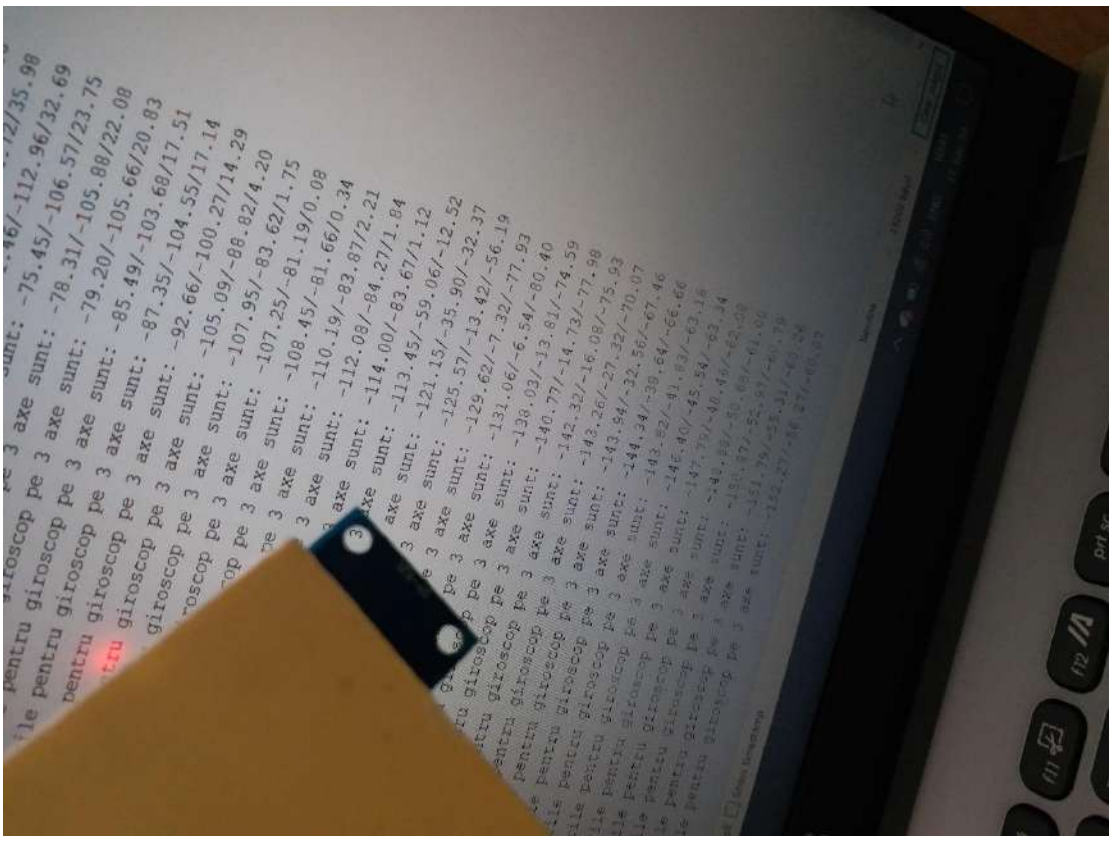

transmitter | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
transmitter
int flex3 = analogRead(A3);
float flex3V = flex3 * VCC / 1023.0;
float flex3R = R_DIV * (VCC / flex3V - 1.0);
float unghi3 = map(flex3R, R_drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 3: " + String(unghi3));
delay(200);
Serial.println("");
int flex4 = analogRead(A2);
float flex4V = flex4 * VCC / 1023.0;
float flex4R = R_DIV * (VCC / flex4V - 1.0);
float unghi4 = map(flex4R, R_drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 4: " + String(unghi4));
delay(200);
Serial.println("");
*/
int flex5 = analogRead(A1);
float flex5V = flex5 * VCC / 1023.0;
float flex5R = R_DIV * (VCC / flex5V - 1.0);
float unghi5 = map(flex5R, R_drept, R_indoit, 0, 90.0);
Serial.println("Valoarea 5: " + String(unghi5));
delay(200);
Serial.println("");
}
COM5
Send
19:30:24.354 -> Am terminat de realizat setup-ul.
19:30:25.362 -> Valoarea 5: -54.00
19:30:25.564 ->
19:30:25.564 -> Valoarea 5: -54.00
19:30:25.761 ->
19:30:25.761 -> Valoarea 5: -54.00
19:30:25.935 ->
19:30:25.935 -> Valoarea 5: -54.00
19:30:26.153 ->
19:30:26.153 -> Valoarea 5: -54.00
19:30:26.339 ->
Autoscroll
Show timestamp
Newline
9600 baud
Clear output
Done uploading.
Sketch uses 10026 bytes (32%) of program storage space. Maximum is 30720 bytes.
Global variables use 271 bytes (13%) of dynamic memory, leaving 1777 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano on COM5
ENG
19:30
11-Jun-20

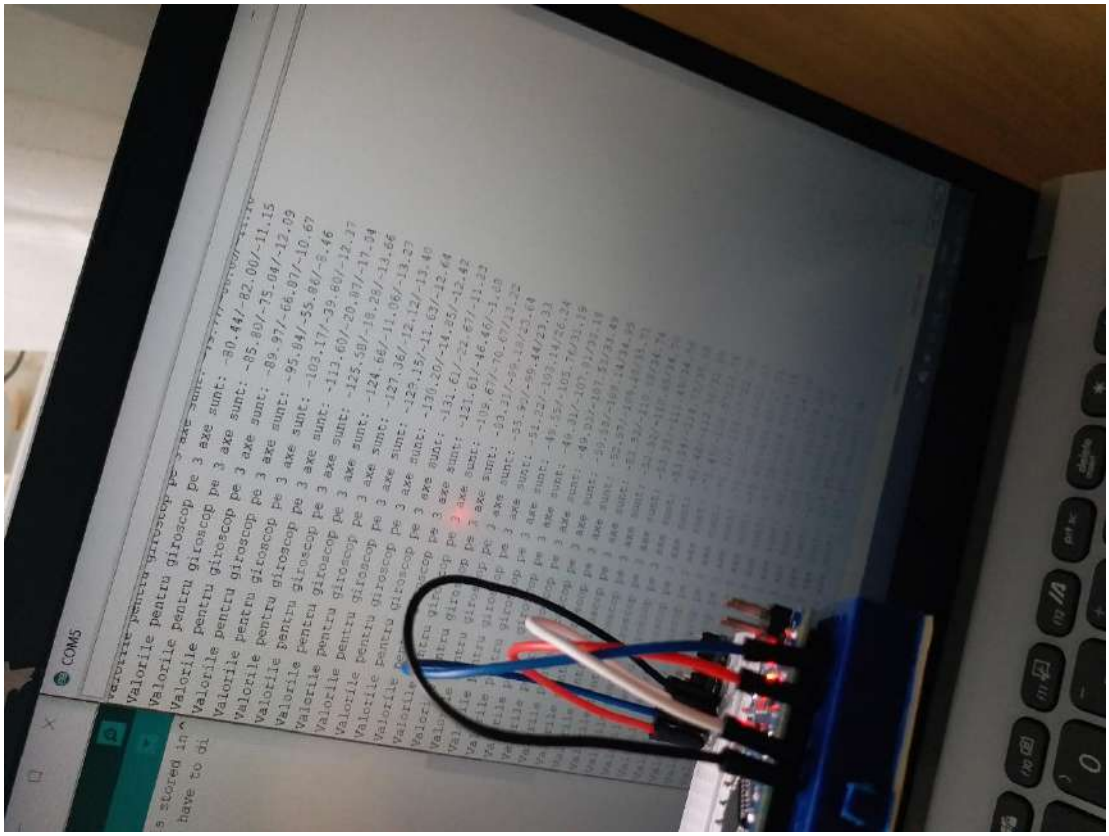
2. MPU6050


AccErrorX: 1.50
AccErrorY: -0.74
GyroErrorX: -6.18
GyroErrorY: -4.15
GyroErrorZ: -0.59
Valorile pentru giroscop pe 3 axe sunt: -74.29/-70.41/-9.03
Valorile pentru giroscop pe 3 axe sunt: -73.79/-70.79/-9.23
Valorile pentru giroscop pe 3 axe sunt: -75.45/-73.27/-9.54
Valorile pentru giroscop pe 3 axe sunt: -75.54/-74.23/-10.95
Valorile pentru giroscop pe 3 axe sunt: -76.21/-76.24/-10.68
Valorile pentru giroscop pe 3 axe sunt: -77.51/-79.10/-9.72
Valorile pentru giroscop pe 3 axe sunt: -78.77/-79.86/-10.19
Valorile pentru giroscop pe 3 axe sunt: -79.77/-80.65/-11.16
Valorile pentru giroscop pe 3 axe sunt: -80.44/-82.00/-11.15
Valorile pentru giroscop pe 3 axe sunt: -85.80/-75.04/-12.09
Valorile pentru giroscop pe 3 axe sunt: -89.97/-66.87/-10.67
Valorile pentru giroscop pe 3 axe sunt: -95.84/-55.86/-8.46
Valorile pentru giroscop pe 3 axe sunt: -103.17/-39.80/-12.37
Valorile pentru giroscop pe 3 axe sunt: -113.60/-20.87/-17.04
Valorile pentru giroscop pe 3 axe sunt: -125.50/-18.28/-13.66
Valorile pentru giroscop pe 3 axe sunt: -124.66/-11.06/-13.27
Valorile pentru giroscop pe 3 axe sunt: -129.15/-11.63/-13.40
Valorile pentru giroscop pe 3 axe sunt: -130.20/-14.15/-12.64
Valorile pentru giroscop pe 3 axe sunt: -128.07/-11.93
Valorile pentru giroscop pe 3 axe sunt: -126.46/-11.93
e is stored in
we have to di
plusivo



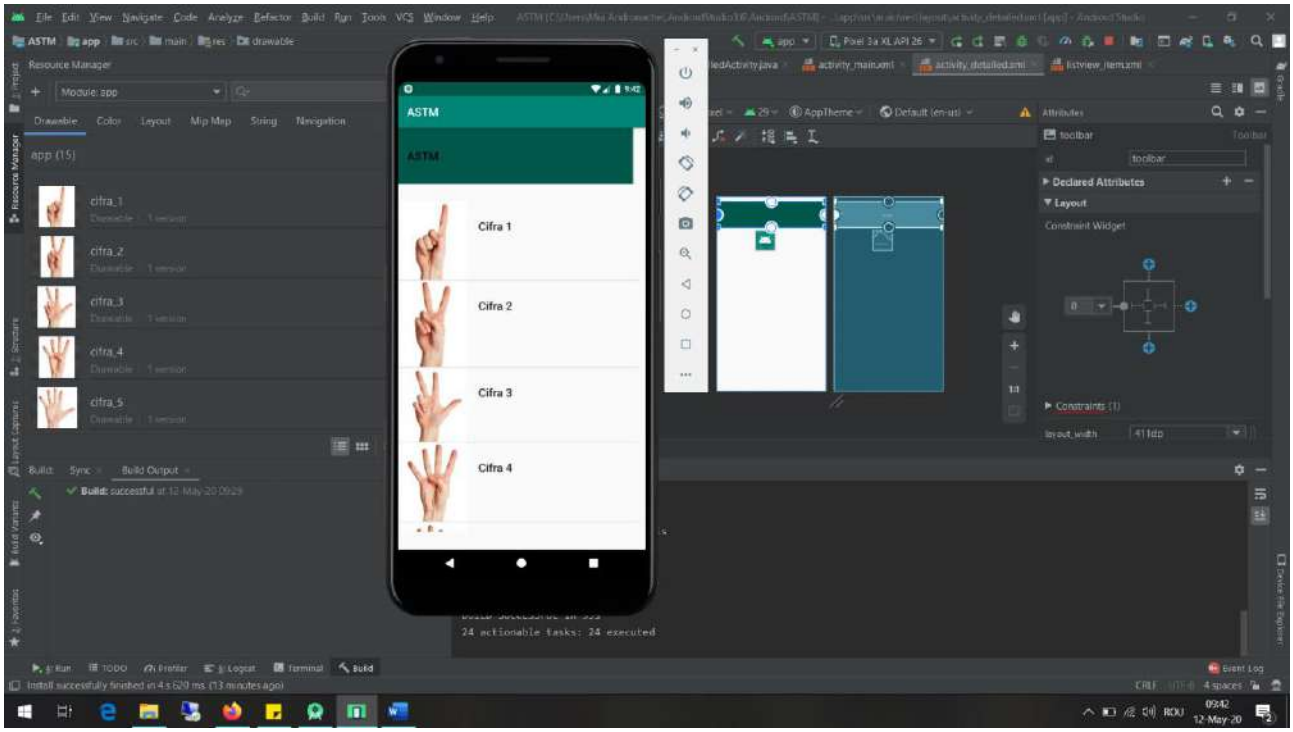


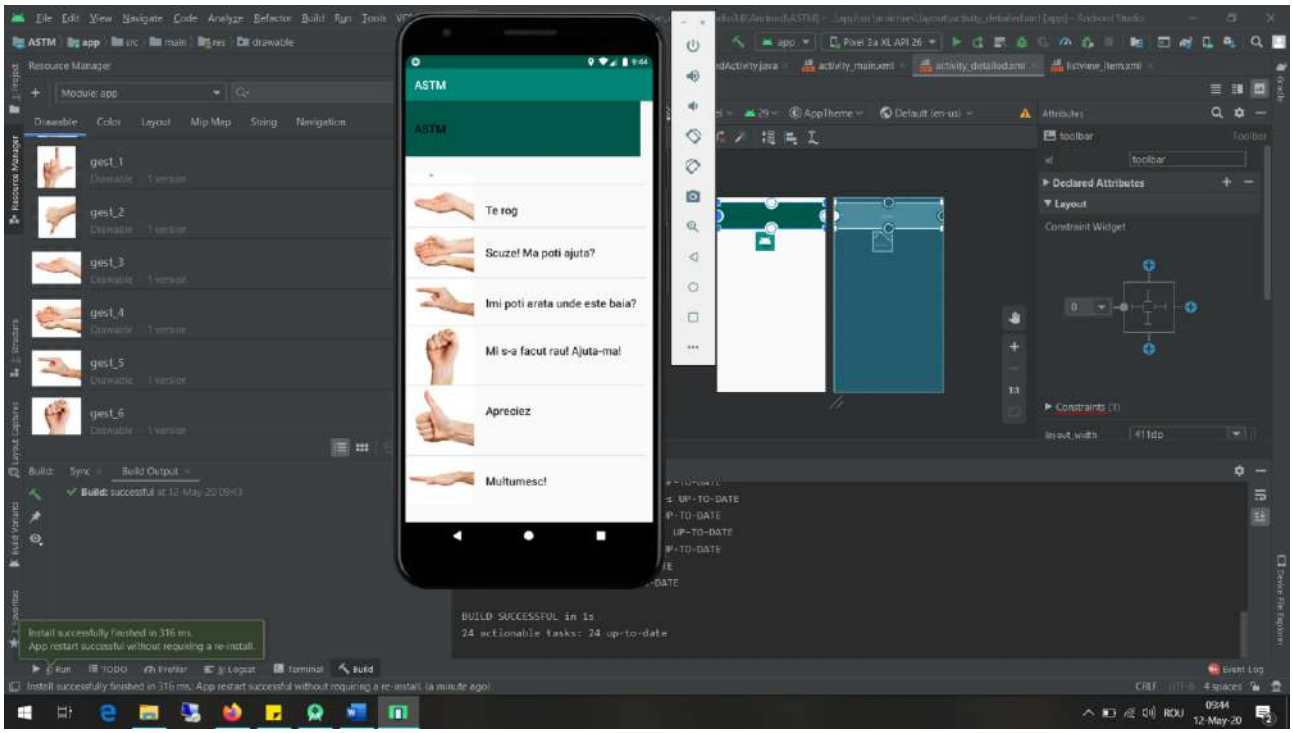

Valorile pentru giroscop pe 3 axe sunt: -75.45/-112.96/32.69
Valorile pentru giroscop pe 3 axe sunt: -78.31/-106.57/23.75
Valorile pentru giroscop pe 3 axe sunt: -79.20/-105.88/22.08
Valorile pentru giroscop pe 3 axe sunt: -85.49/-105.66/20.83
Valorile pentru giroscop pe 3 axe sunt: -87.35/-103.68/17.51
Valorile pentru giroscop pe 3 axe sunt: -92.66/-104.55/17.14
Valorile pentru giroscop pe 3 axe sunt: -105.09/-100.27/14.29
Valorile pentru giroscop pe 3 axe sunt: -107.95/-88.82/4.20
Valorile pentru giroscop pe 3 axe sunt: -107.25/-83.62/1.75
Valorile pentru giroscop pe 3 axe sunt: -108.45/-81.19/0.08
Valorile pentru giroscop pe 3 axe sunt: -110.19/-81.66/0.34
Valorile pentru giroscop pe 3 axe sunt: -112.08/-83.87/2.21
Valorile pentru giroscop pe 3 axe sunt: -114.00/-84.27/1.84
Valorile pentru giroscop pe 3 axe sunt: -113.45/-83.67/1.12
Valorile pentru giroscop pe 3 axe sunt: -121.15/-59.06/-12.52
Valorile pentru giroscop pe 3 axe sunt: -125.57/-35.90/-32.37
Valorile pentru giroscop pe 3 axe sunt: -129.62/-13.42/-56.19
Valorile pentru giroscop pe 3 axe sunt: -131.06/-7.32/-77.93
Valorile pentru giroscop pe 3 axe sunt: -138.03/-6.54/-80.40
Valorile pentru giroscop pe 3 axe sunt: -140.77/-13.81/-74.59
Valorile pentru giroscop pe 3 axe sunt: -142.32/-14.73/-77.98
Valorile pentru giroscop pe 3 axe sunt: -143.26/-16.08/-75.93
Valorile pentru giroscop pe 3 axe sunt: -143.94/-27.32/-70.07
Valorile pentru giroscop pe 3 axe sunt: -144.34/-32.56/-67.46
Valorile pentru giroscop pe 3 axe sunt: -143.82/-39.64/-66.66
Valorile pentru giroscop pe 3 axe sunt: -146.46/-41.83/-63.16
Valorile pentru giroscop pe 3 axe sunt: -147.79/-45.54/-63.34
Valorile pentru giroscop pe 3 axe sunt: -148.88/-50.68/-62.20
Valorile pentru giroscop pe 3 axe sunt: -150.57/-55.51/-61.08
Valorile pentru giroscop pe 3 axe sunt: -151.75/-55.51/-60.79
Valorile pentru giroscop pe 3 axe sunt: -152.29/-58.24/-60.62

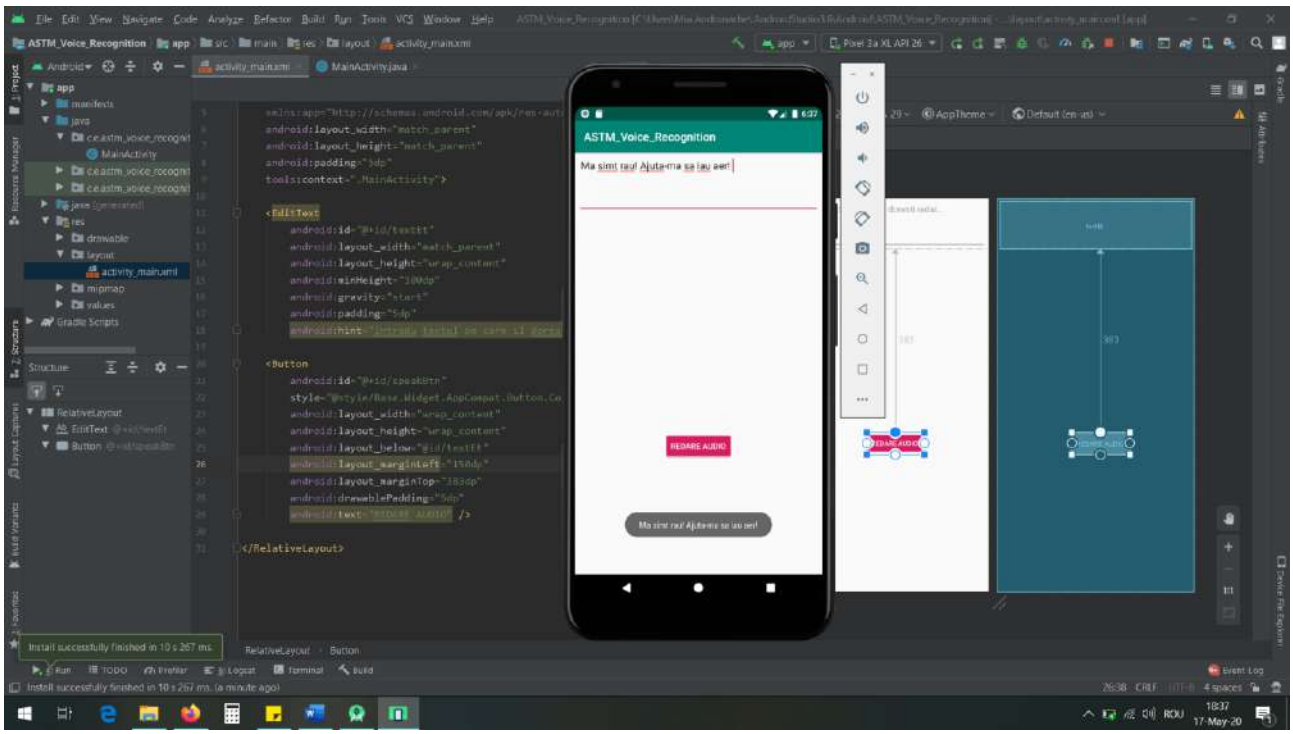

COM5

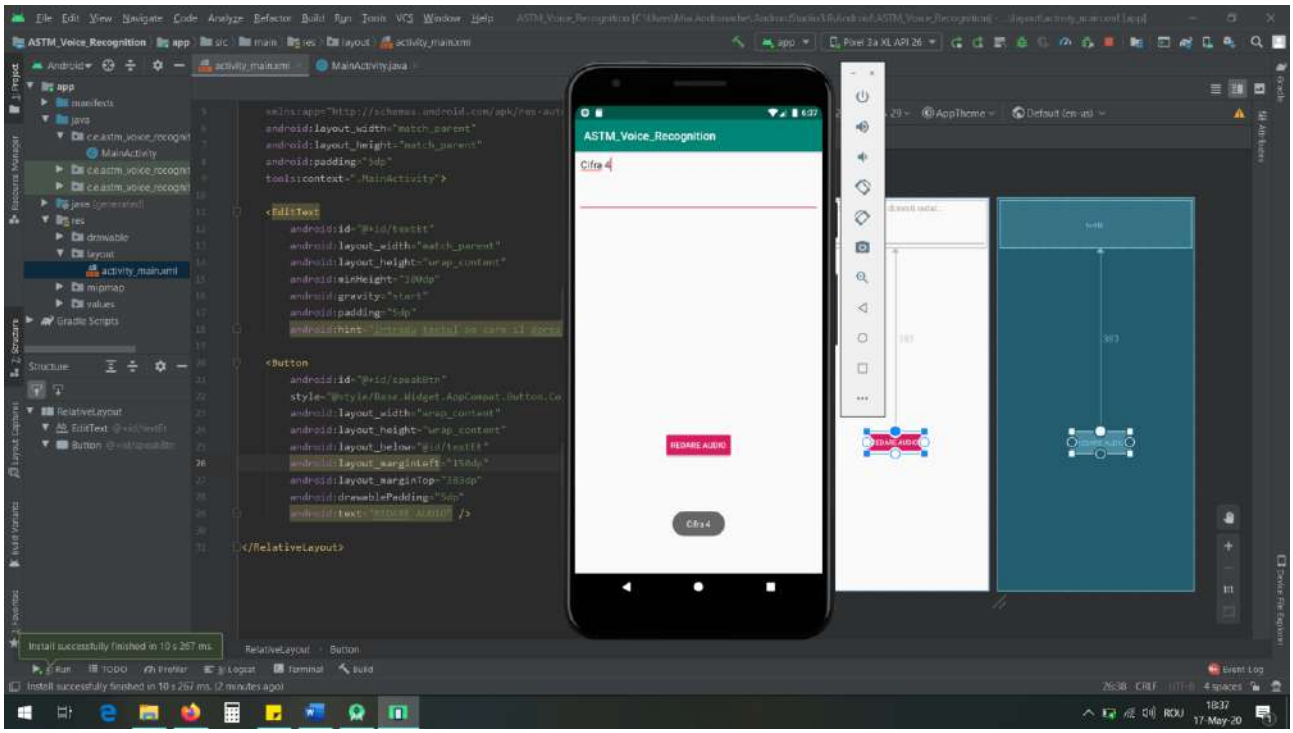
ANEXA 2

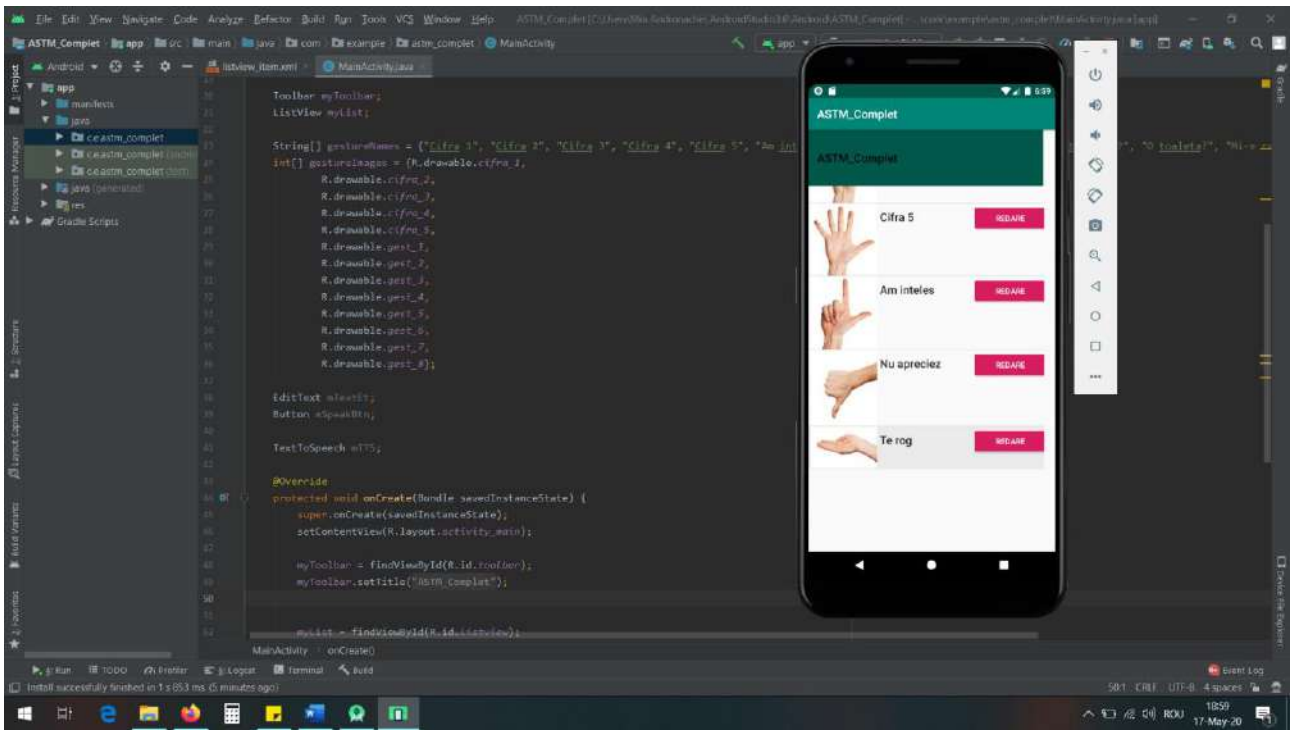
Părți intermediare pentru aplicațiile Android

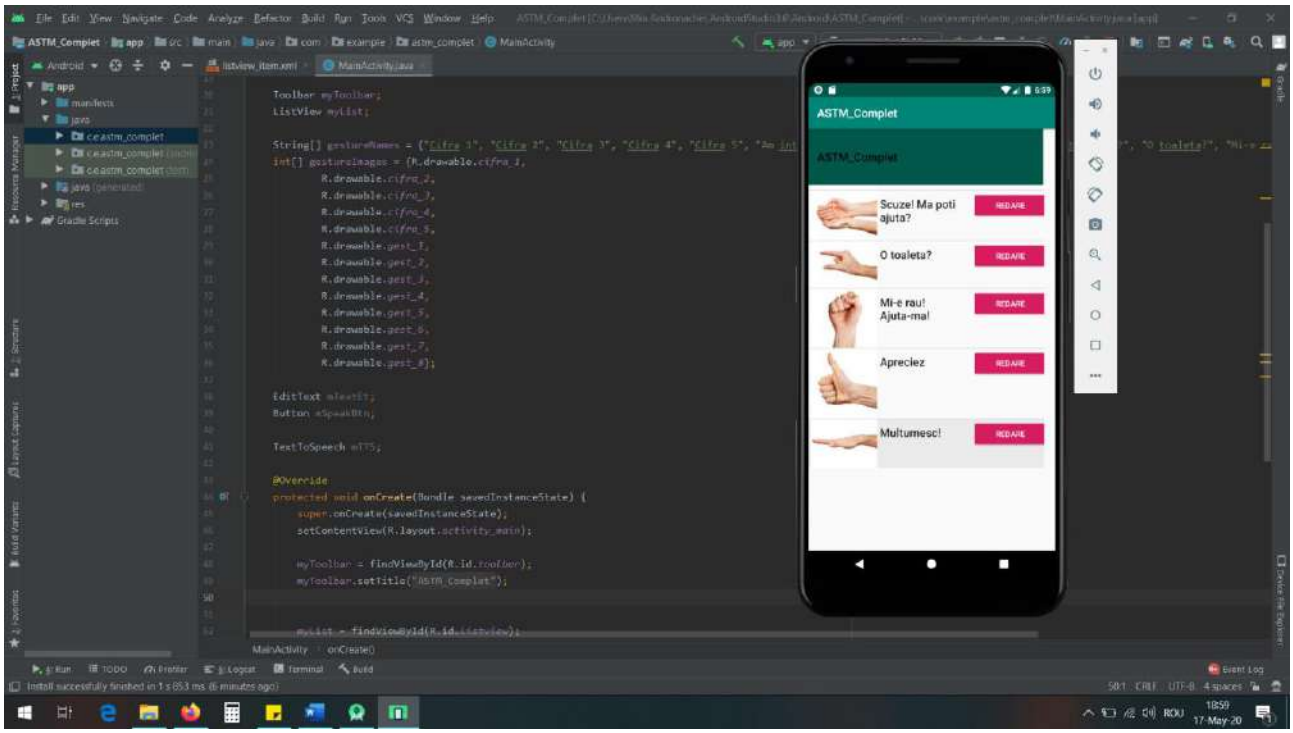





ASTM_Voice_Recognition
Ma simt rau! Ajuta-ma sa iau aer!
REDARE AUDIO
Ma simt rau! Ajuta-ma sa iau aer!

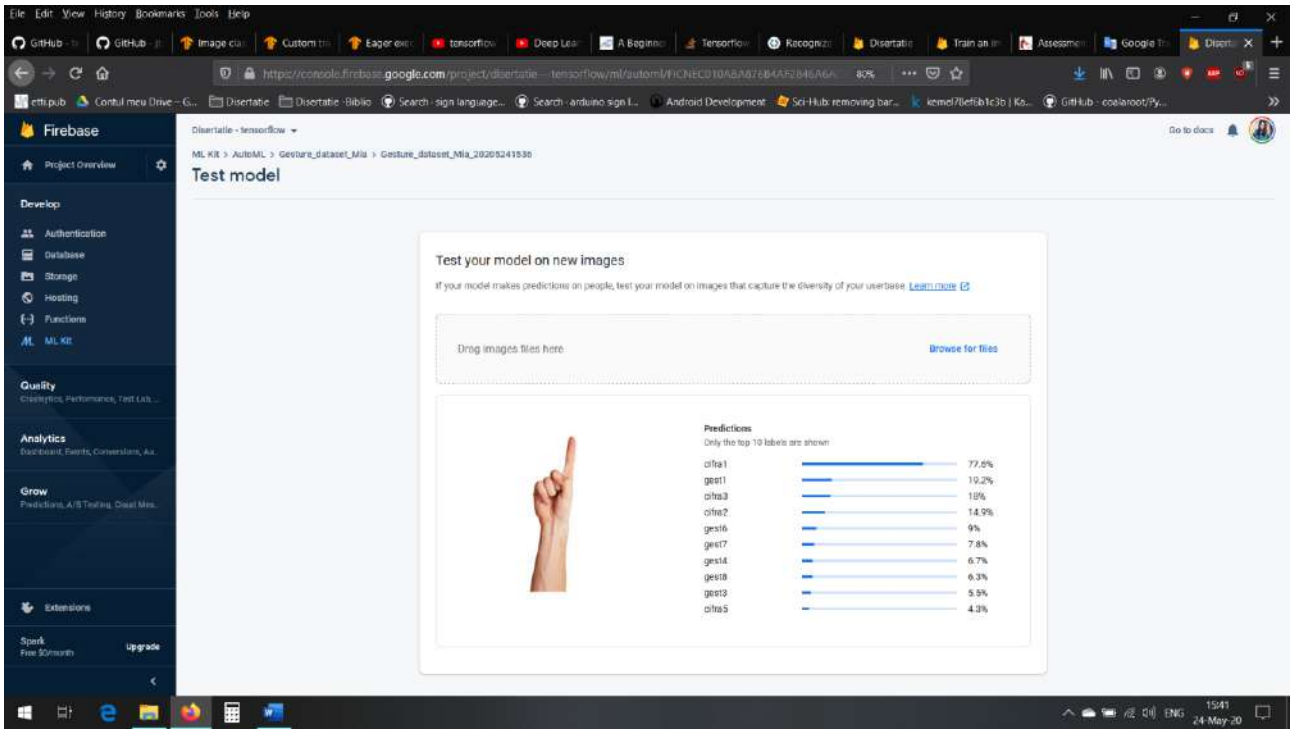

ASTM_Voice_Recognition
Cifra 4
REDARE AUDIO
Cifra 4

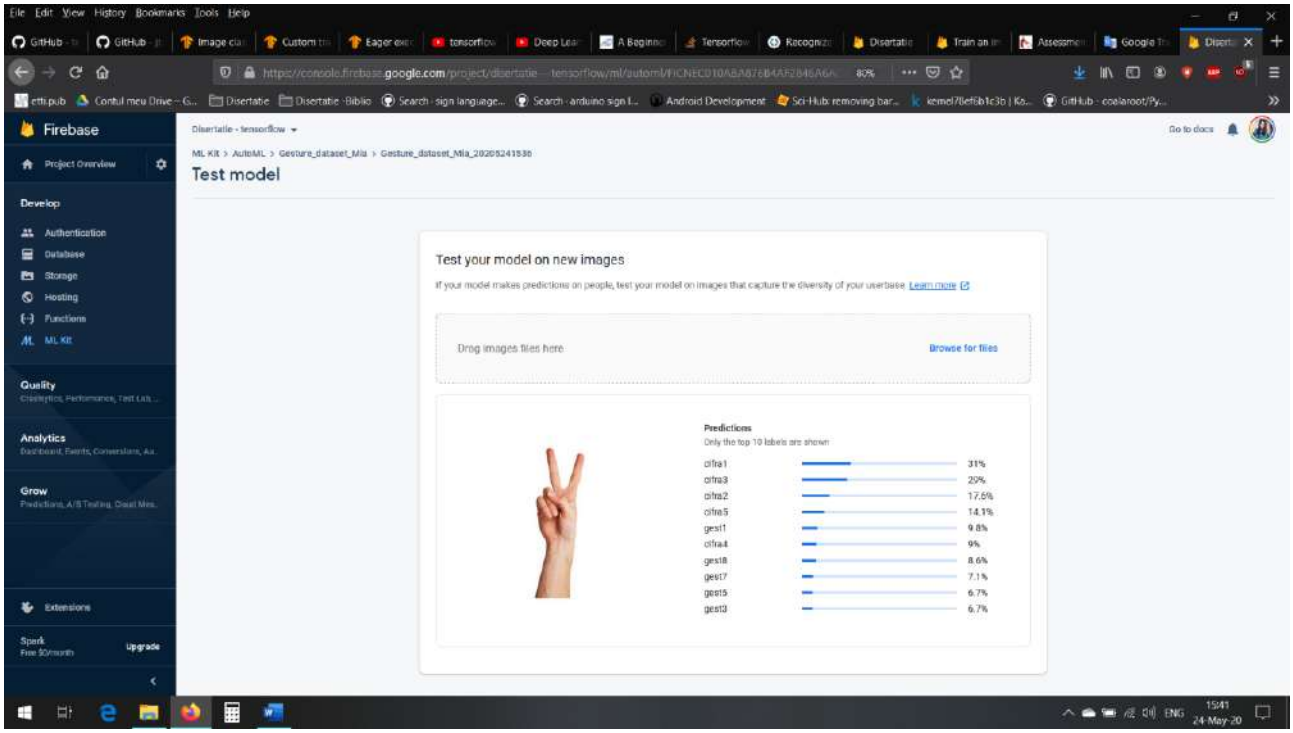

ASTM_Complet
Cifra 5
REDARE
Am inteles
REDARE
Nu apreciez
REDARE
Te rog
REDARE


ASTM_Complet
Scuze! Ma poti ajuta?
REDARE
O toaleta?
REDARE
Mi-e rau! Ajuta-ma!
REDARE
Apreciez
REDARE
Multumesc!
REDARE

ANEXA 3

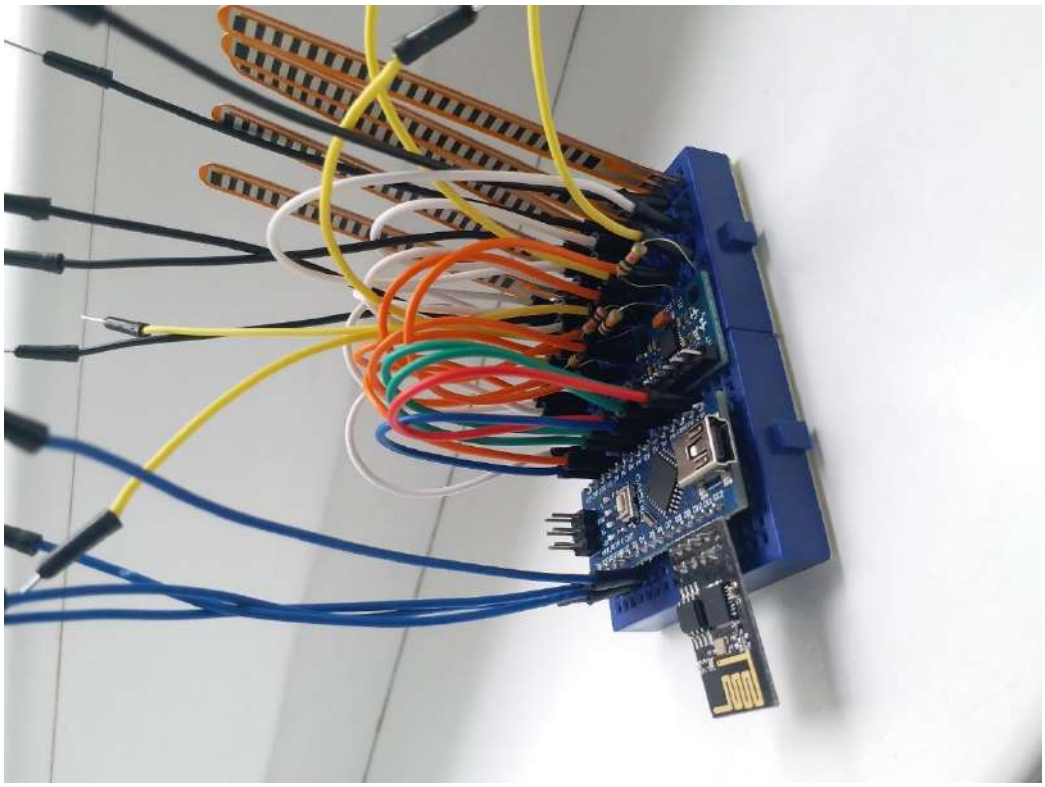
Set de date de antrenare

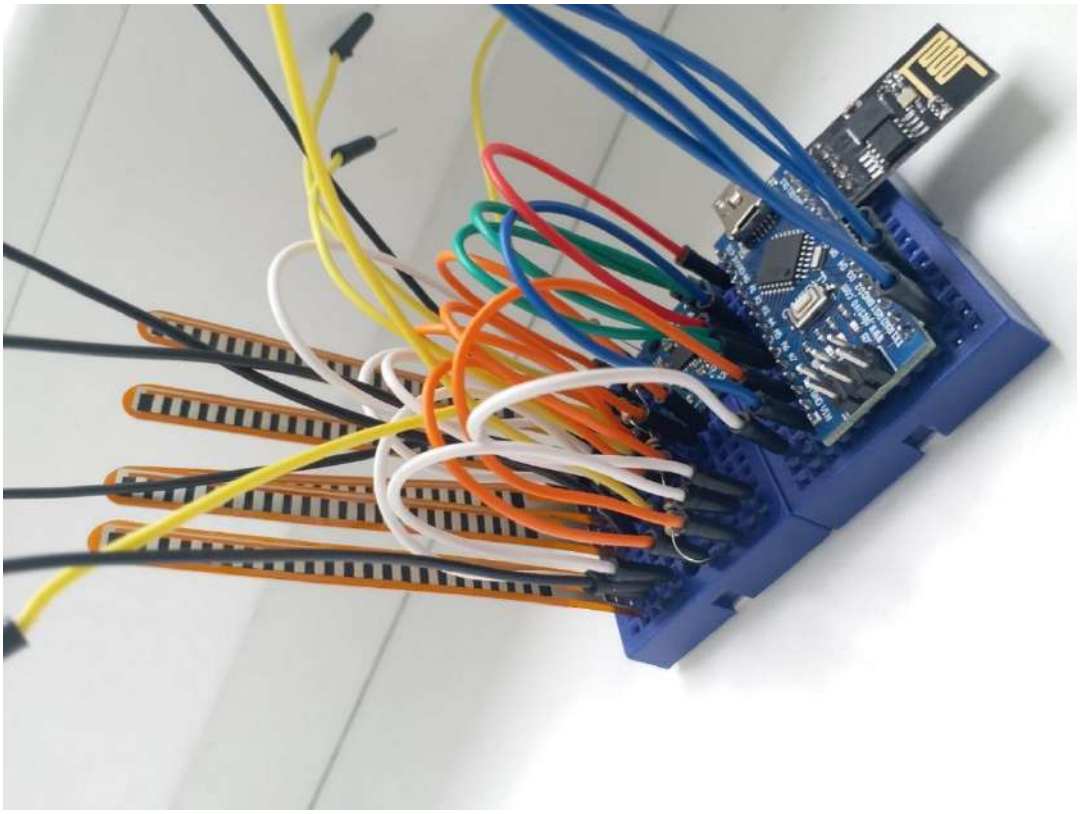


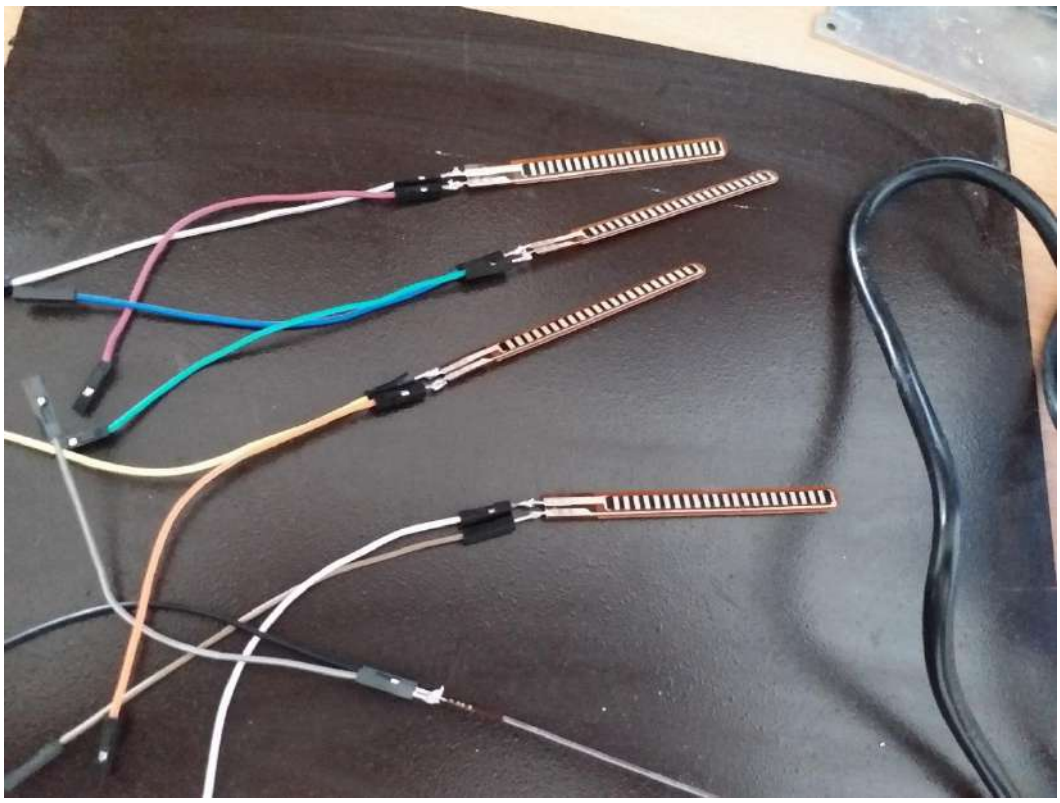


ANEXA 4

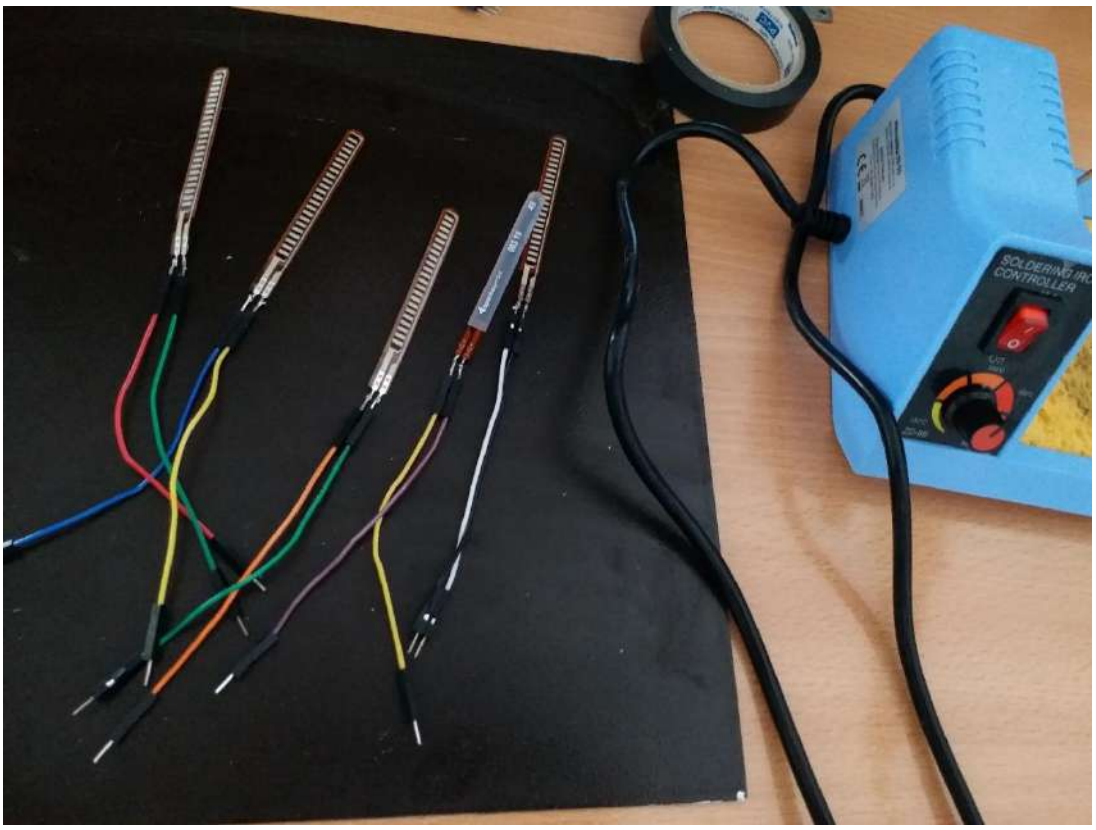
Pași intermediari în cadrul realizării circuitului de test

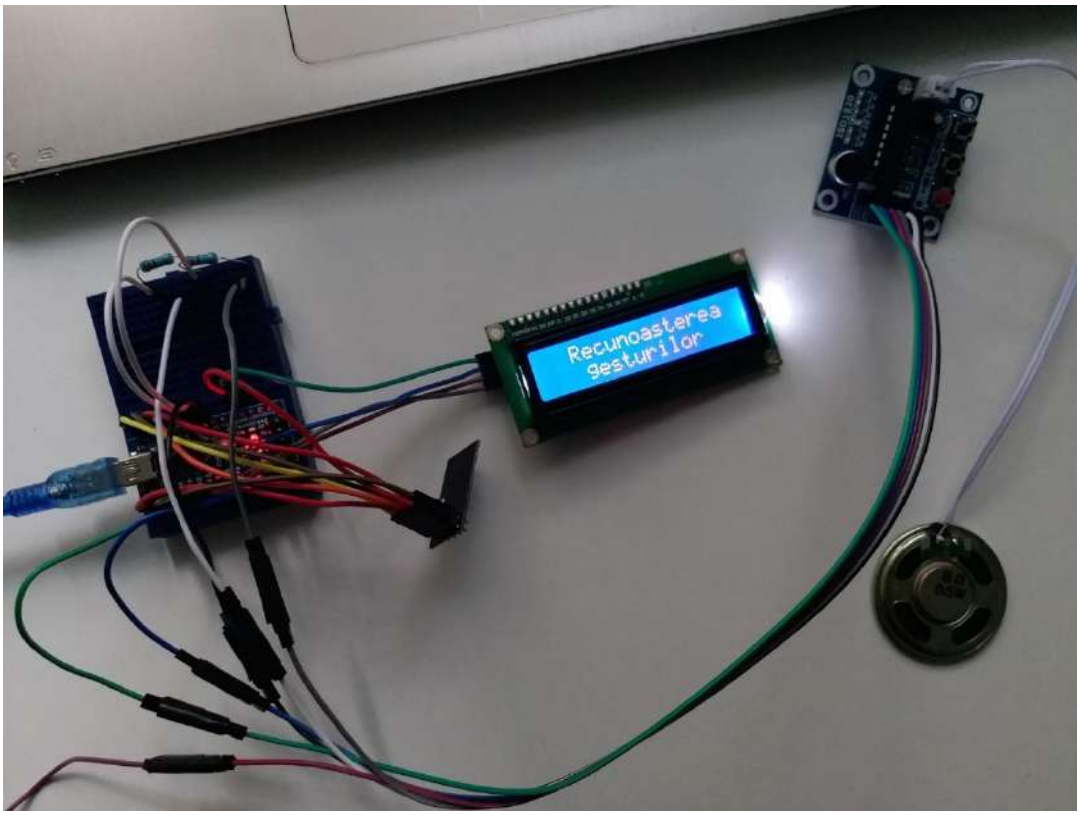







SOLDERING IRON
CONTROLLER

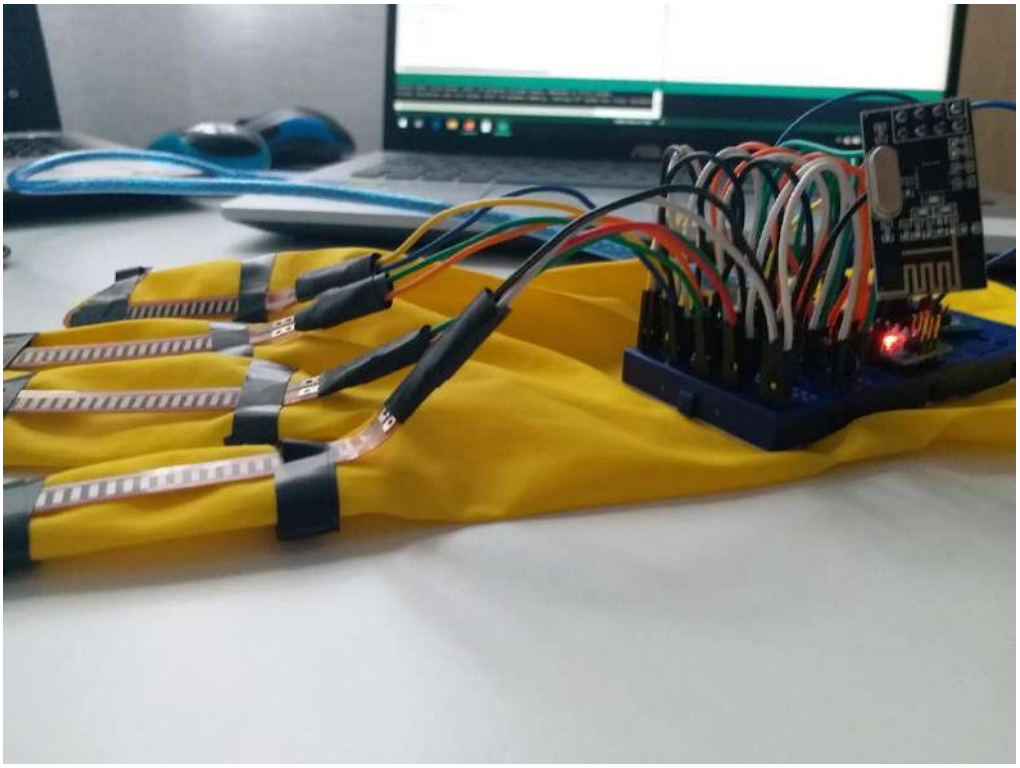

SOLDERING IRON
CONTROLLER

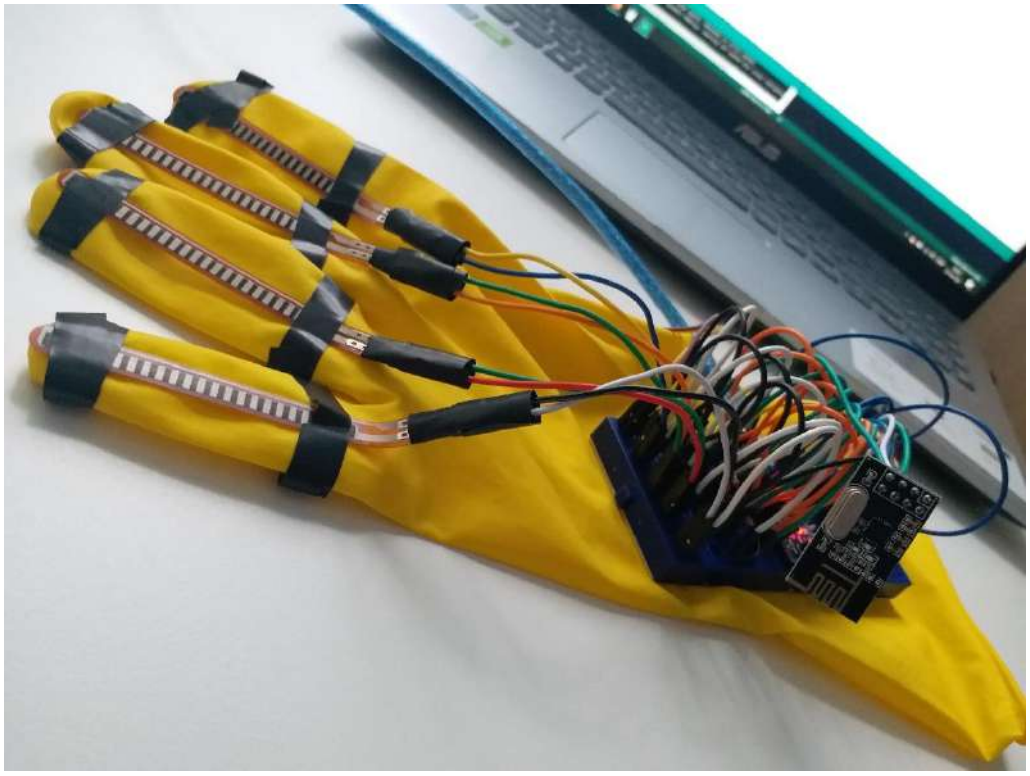

Recunoasterea
gesturilor

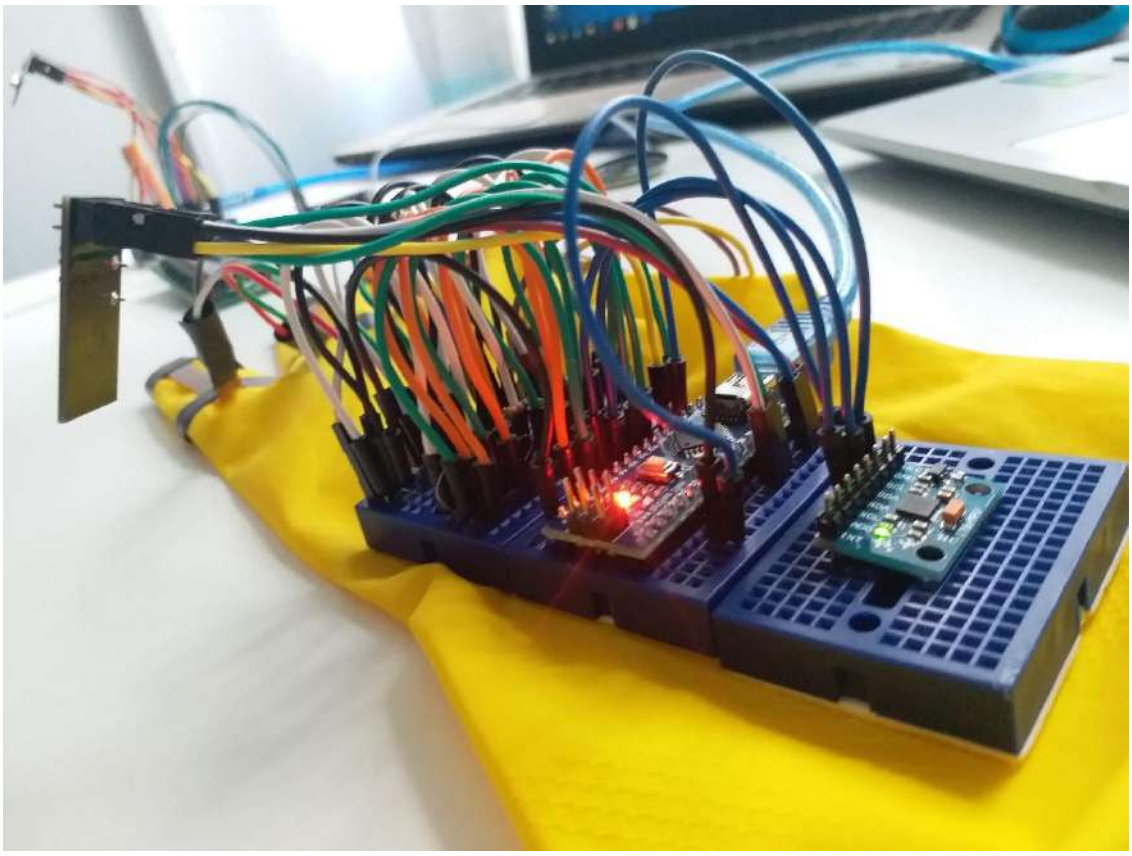
ANEXA 5

Imagini cu circuitul de test realizat

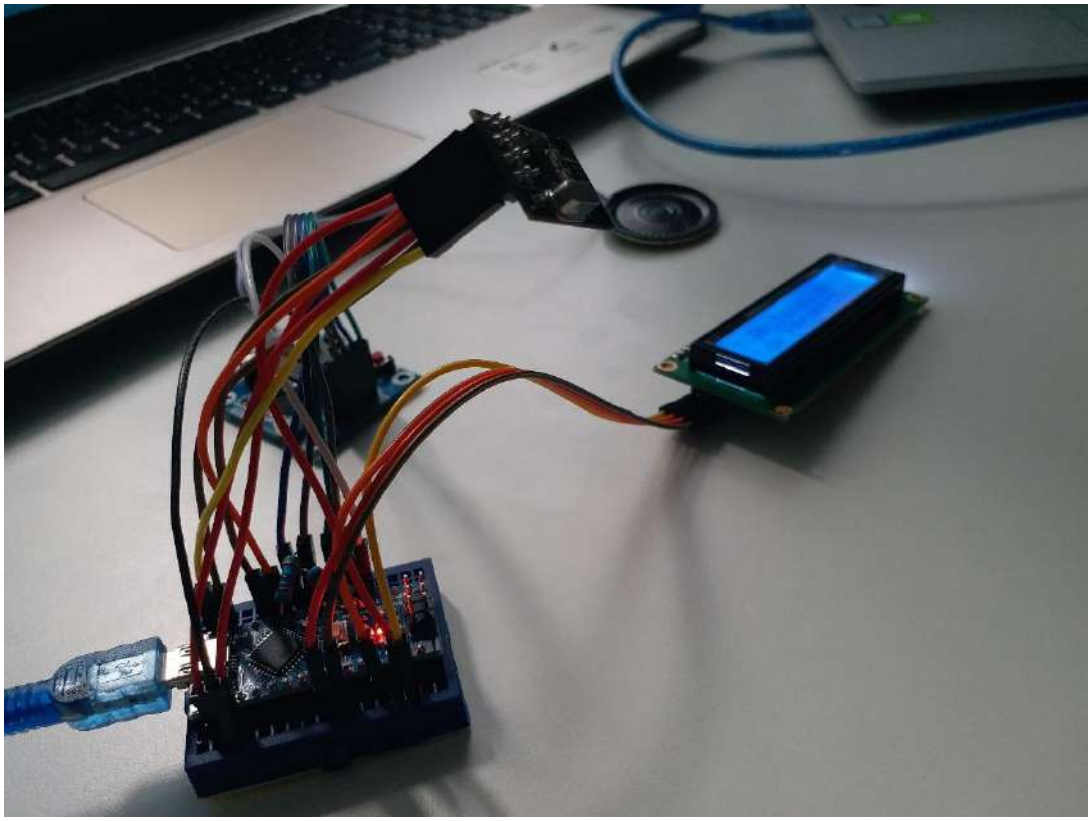
1. Circuitul de detecție a gestului

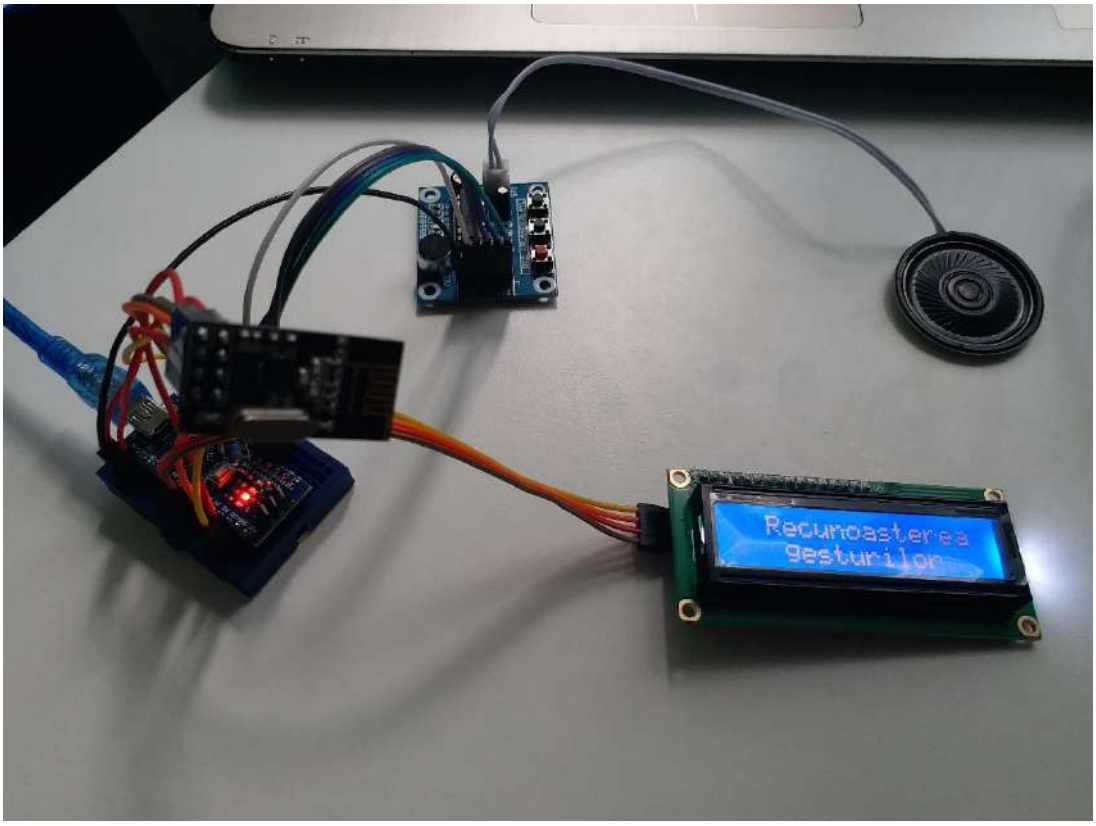




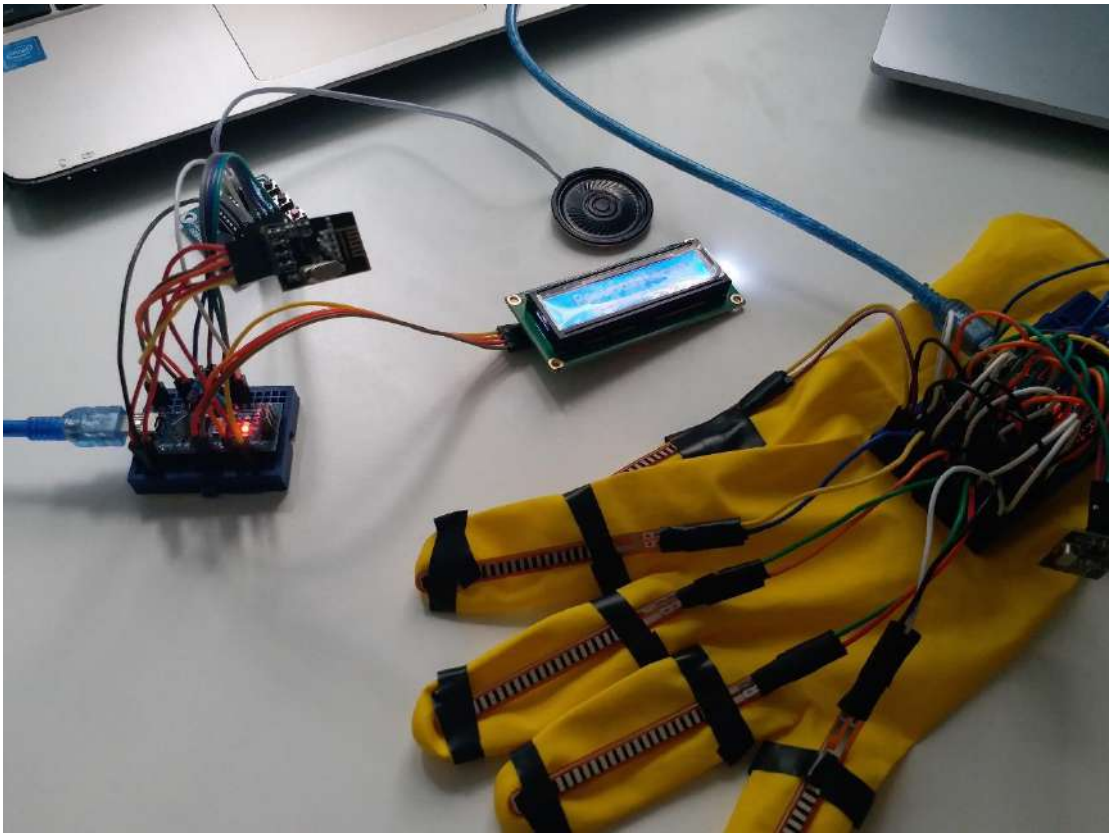


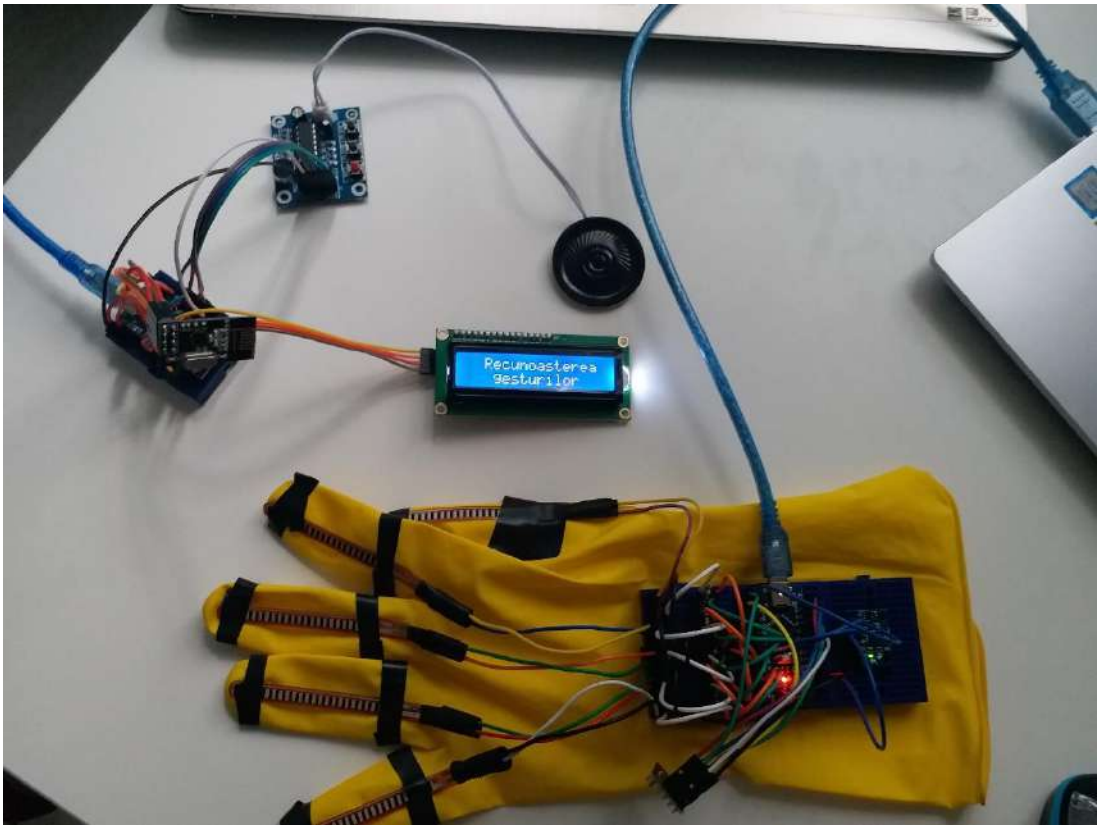
2. Circuitul de translație a gestului





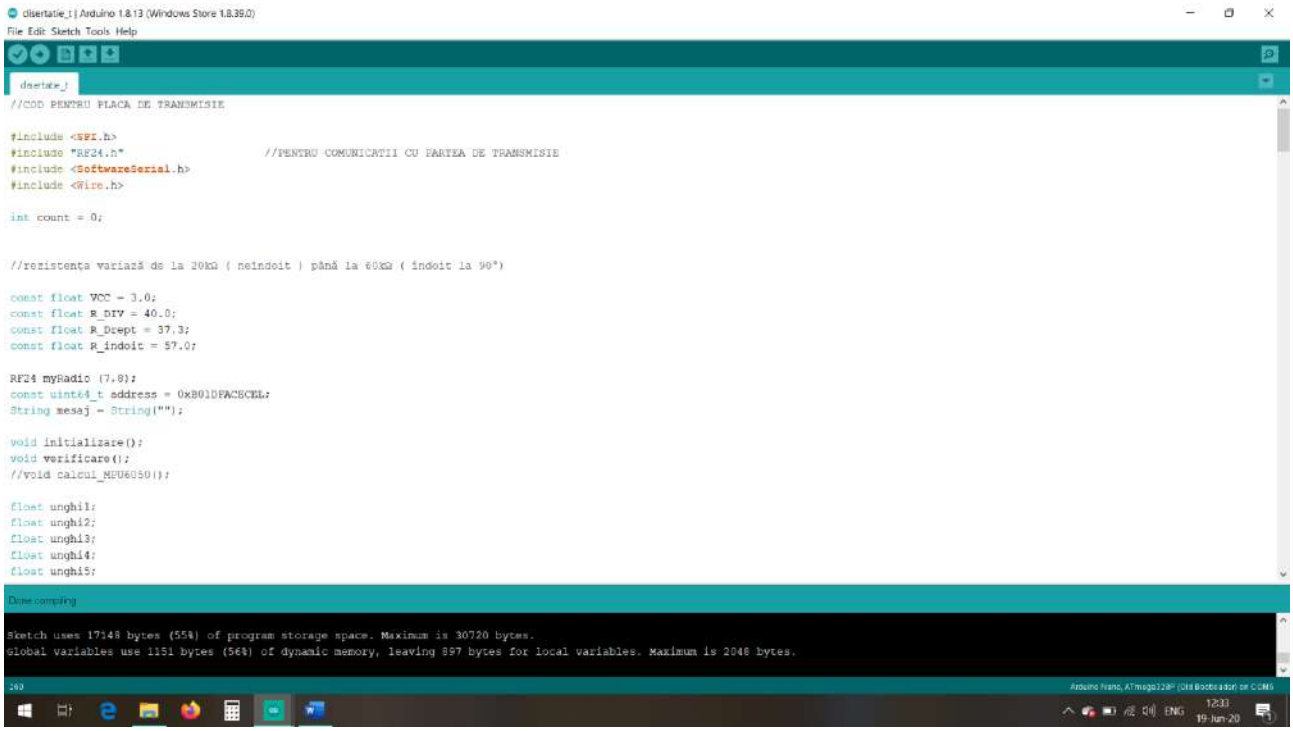
3. Circuitul de test final





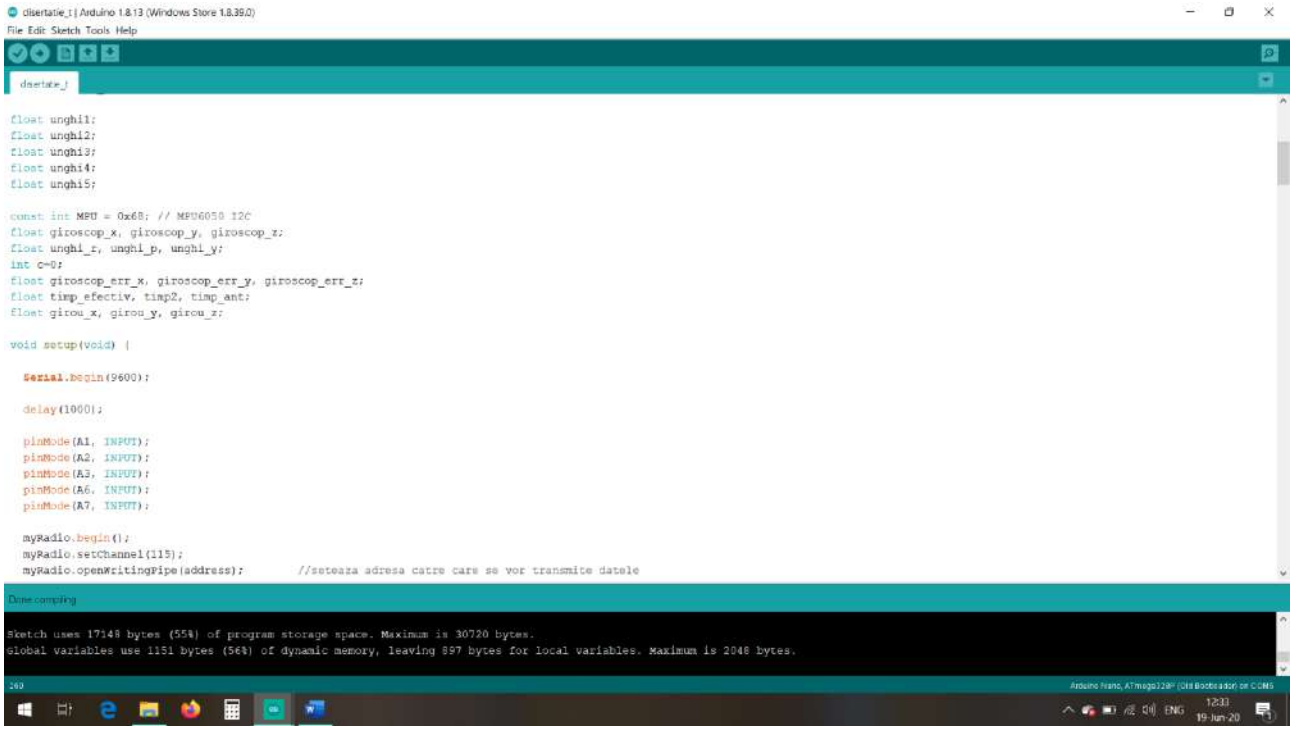
CODUL APLICAȚIEI ÎN ARDUINO

1. Partea de transmisie



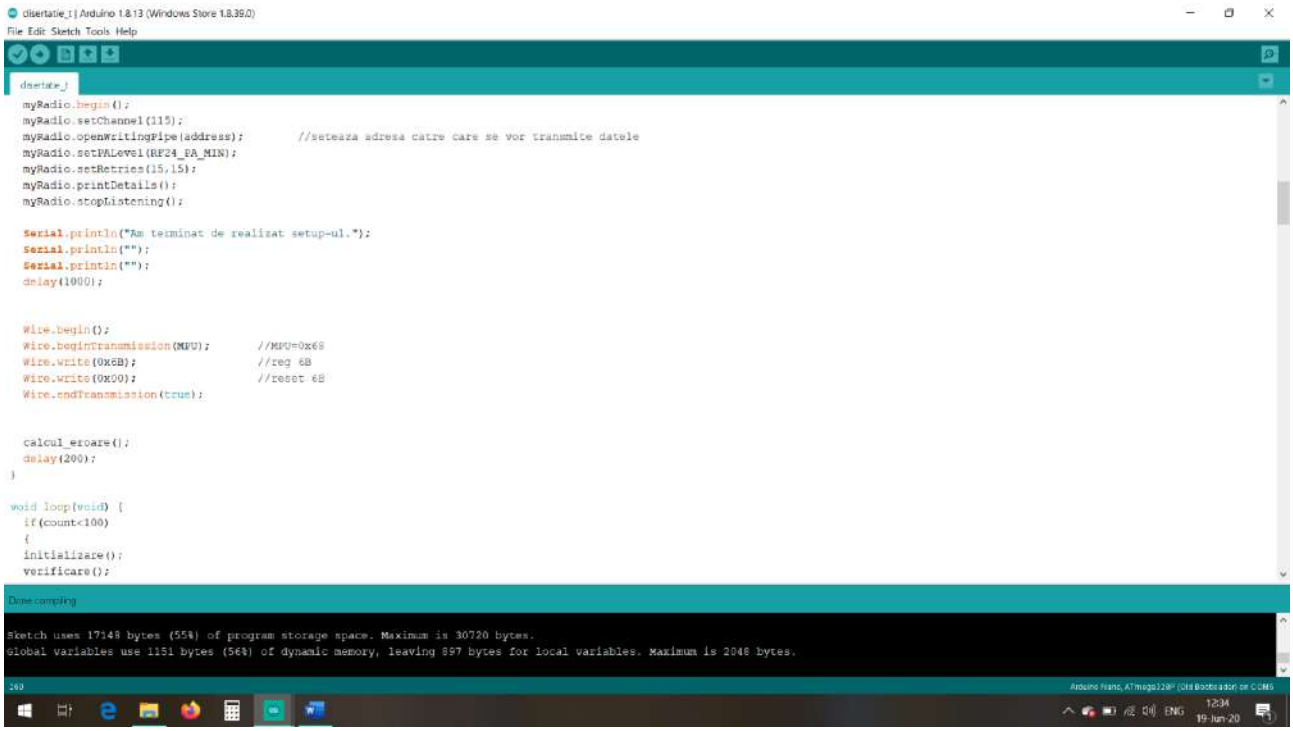
```
//COD PENTRU PLACA DE TRANSMISIE

#include <SPI.h>
#include "RF24.h"                    //PENTRU COMUNICATII CU PARTEA DE TRANSMISIE
#include <SoftwareSerial.h>
#include <Wire.h>

int count = 0;


//rezistența variază de la 20kΩ ( neîndoit ) până la 60kΩ ( îndoit la 90°)

const float VCC = 3.0;
const float R_DIV = 40.0;
const float R_Drept = 37.3;
const float R_Indoit = 57.0;

RF24 myRadio (7,8);
const uint64_t address = 0xB01DFAC8CELL;
String mesaj = String("");

void initializare();
void verificare();
//void calcul_MPU6050();

float unghi1;
float unghi2;
float unghi3;
float unghi4;
float unghi5;
```

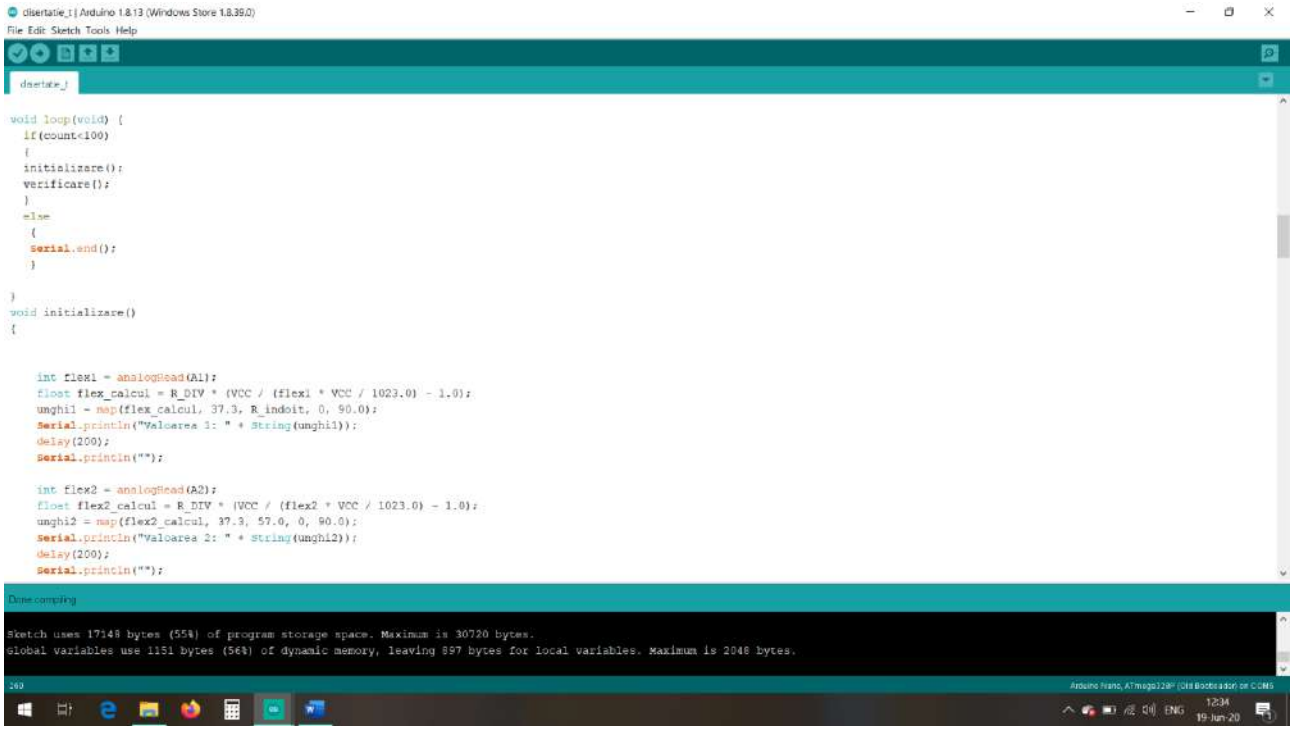
```
Done compiling

Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
```

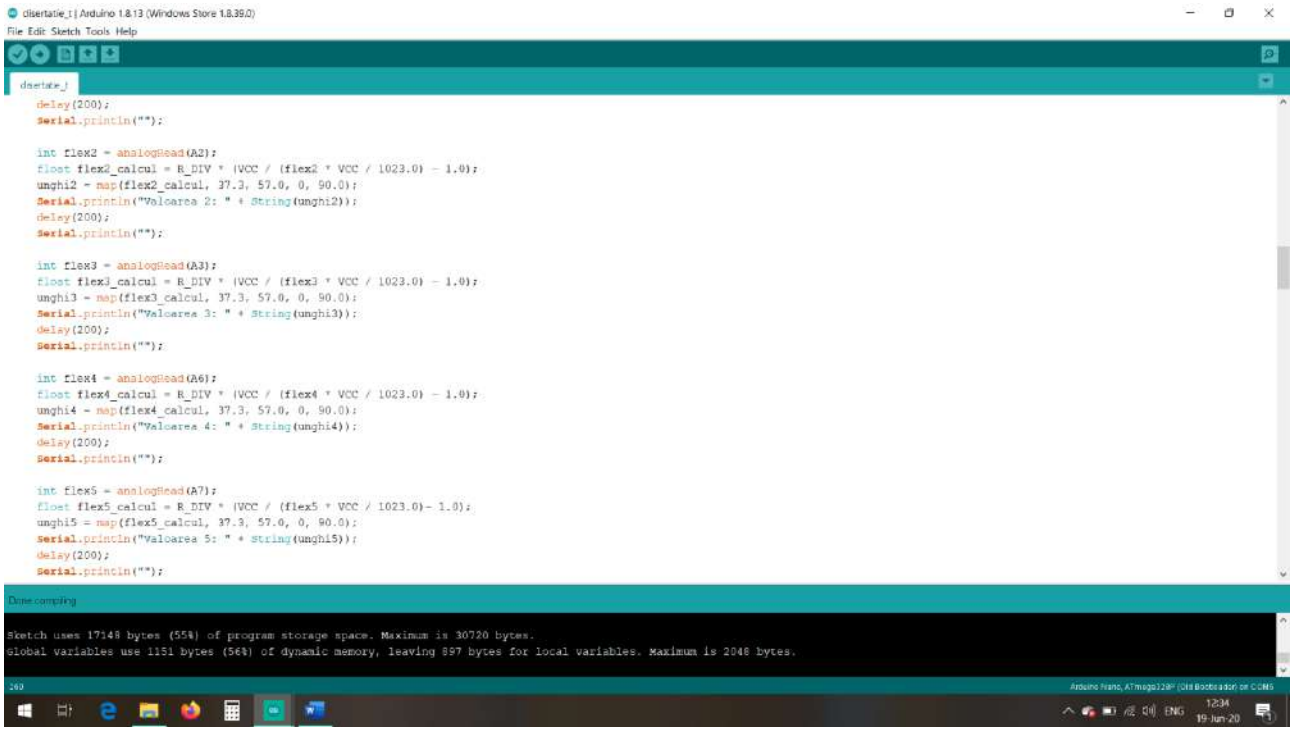


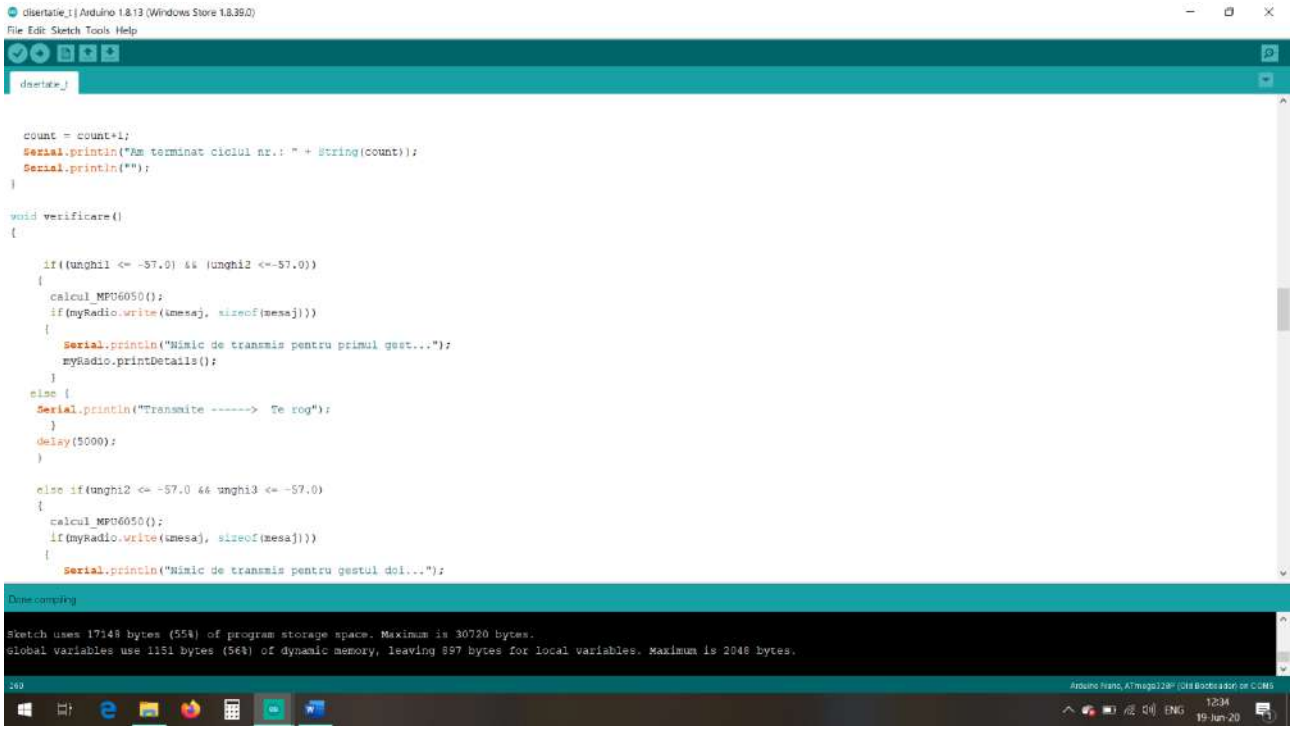
```
float unghi1;
float unghi2;
float unghi3;
float unghi4;
float unghi5;

const int MPU = 0x68; // MPU6050 I2C
float giroscop_x, giroscop_y, giroscop_z;
float unghi_r, unghi_p, unghi_y;
int c=0;
float giroscop_err_x, giroscop_err_y, giroscop_err_z;
float timp_efectiv, timp2, timp_ant;
float girou_x, girou_y, girou_z;

void setup(void) {

  Serial.begin(9600);

  delay(1000);

  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(A6, INPUT);
  pinMode(A7, INPUT);

  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.openWritingPipe(address);        //seteaza adresa catre care se vor transmite datele
```

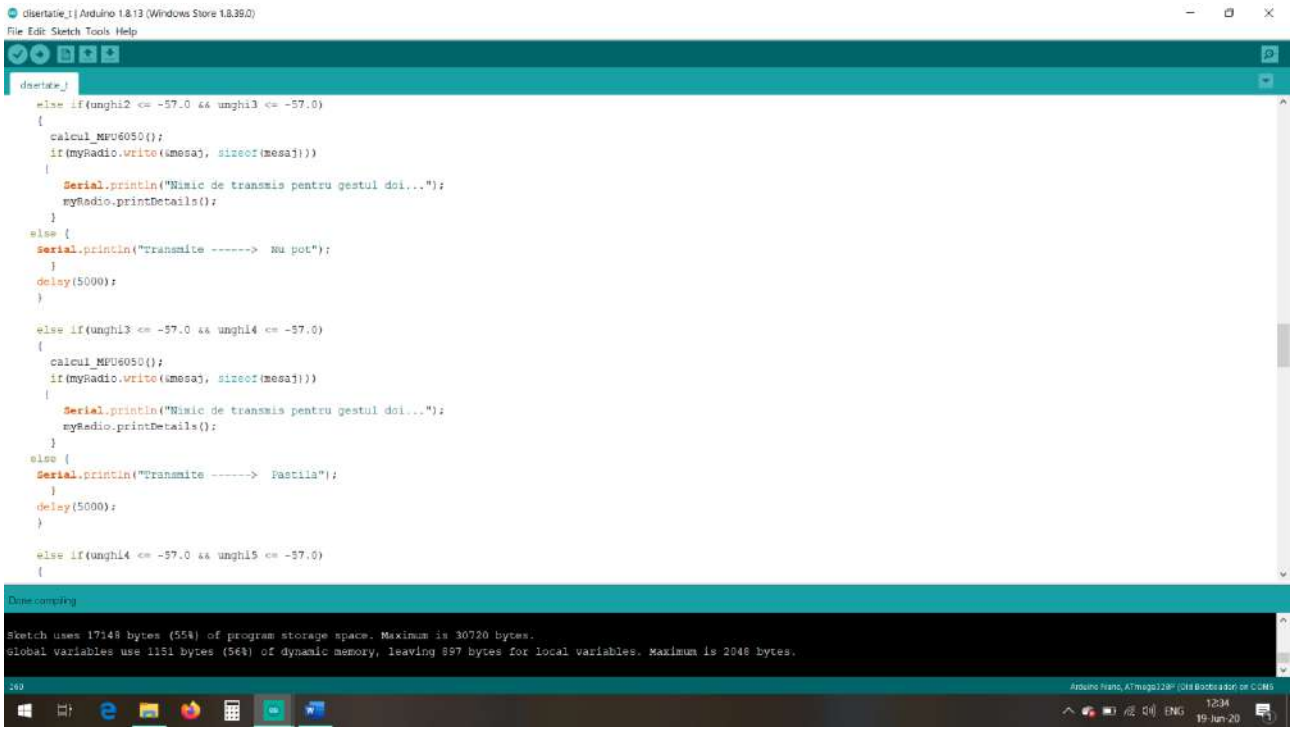
```
Done compiling

Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
```

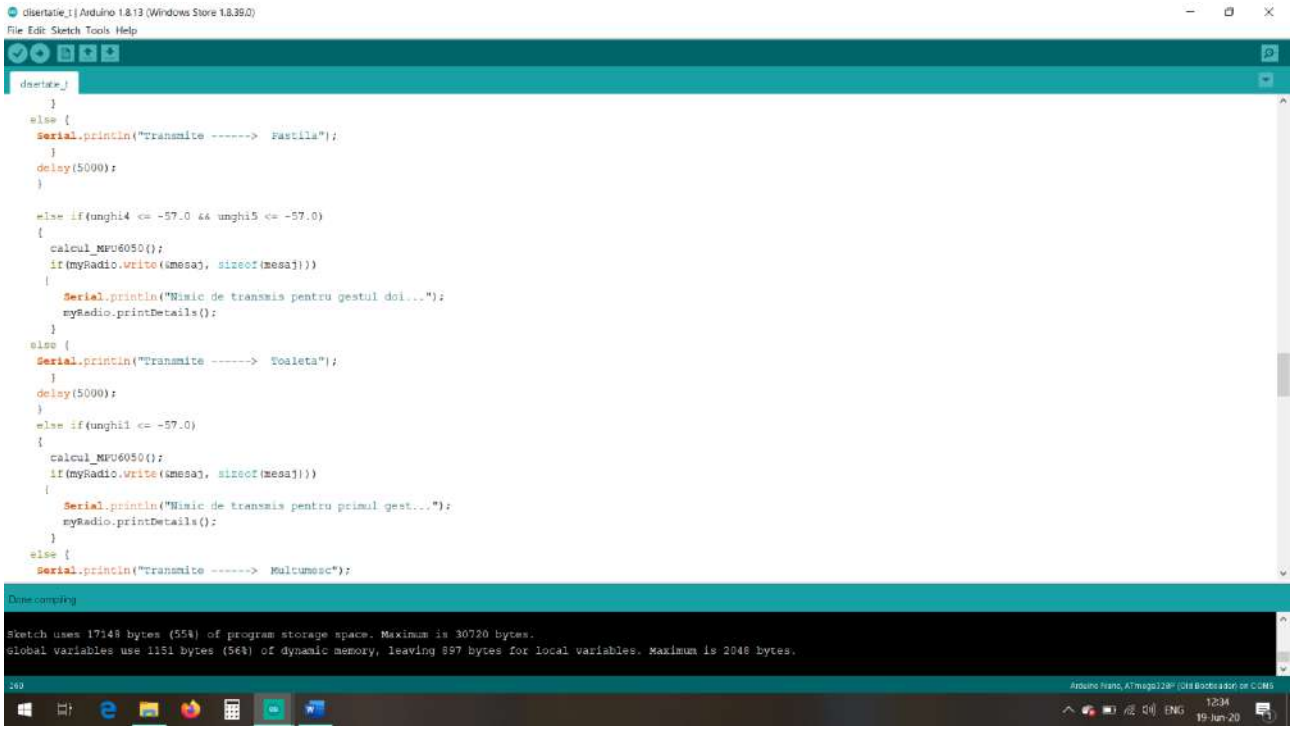

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
myRadio.begin();
myRadio.setChannel(115);
myRadio.openWritingPipe(address);
//seteaza adresa catre care se vor transmite datele
myRadio.setPALevel(RF24_PA_MIN);
myRadio.setRetries(15,15);
myRadio.printDetails();
myRadio.stopListening();
Serial.println("Am terminat de realizat setup-ul.");
Serial.println("");
Serial.println("");
delay(1000);
Wire.begin();
Wire.beginTransmission(MPU);
//MPU=0x68
Wire.write(0x6B);
//reg 6B
Wire.write(0x00);
//reset 6B
Wire.endTransmission(true);
calcul_eroare();
delay(200);
}
void loop(void) {
if(count<100)
{
initializare();
verificare();
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:34
19-Jun-20

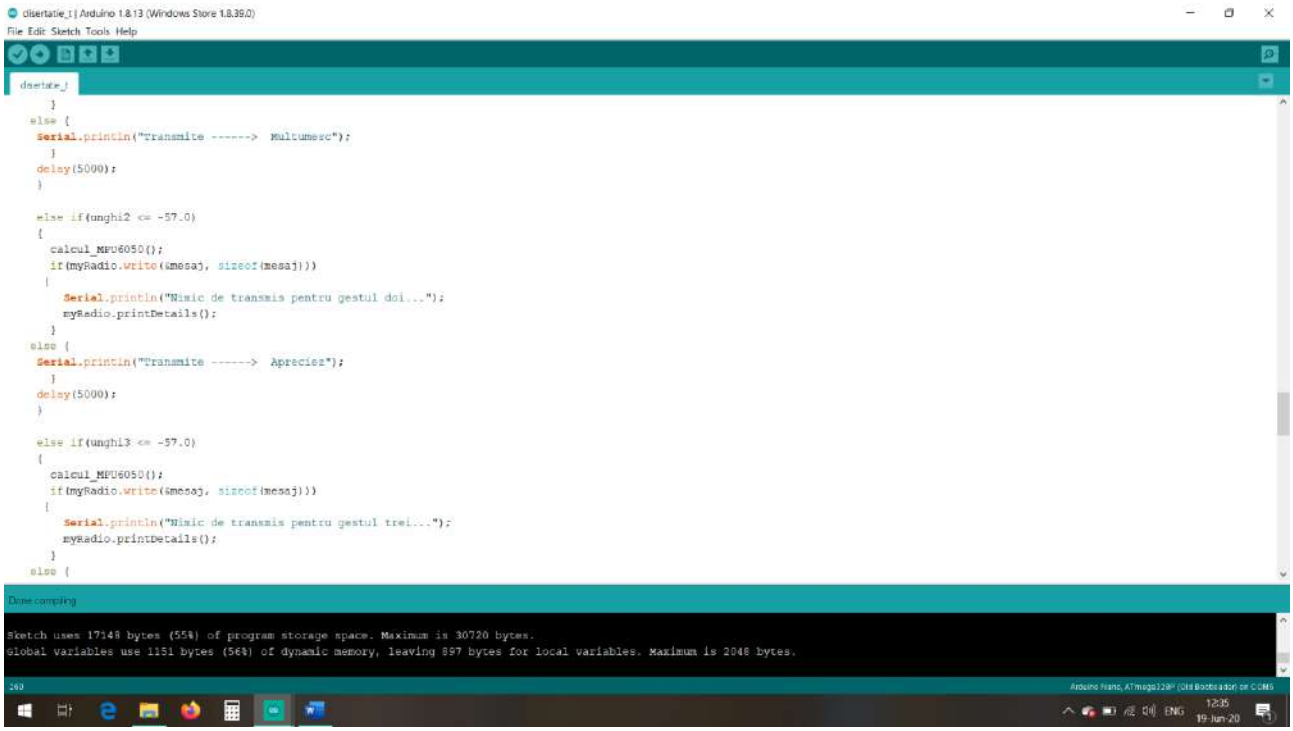

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
void loop(void) {
if(count<100)
{
initializare();
verificare();
}
else
{
Serial.end();
}
}
void initializare()
{
int flex1 = analogRead(A1);
float flex_calcul = R_DIV * (VCC / (flex1 * VCC / 1023.0) - 1.0);
unghi1 = map(flex_calcul, 37.3, R_indoit, 0, 90.0);
Serial.println("Valoarea 1: " + String(unghi1));
delay(200);
Serial.println("");
int flex2 = analogRead(A2);
float flex2_calcul = R_DIV * (VCC / (flex2 * VCC / 1023.0) - 1.0);
unghi2 = map(flex2_calcul, 37.3, 57.0, 0, 90.0);
Serial.println("Valoarea 2: " + String(unghi2));
delay(200);
Serial.println("");
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:34
19-Jun-20

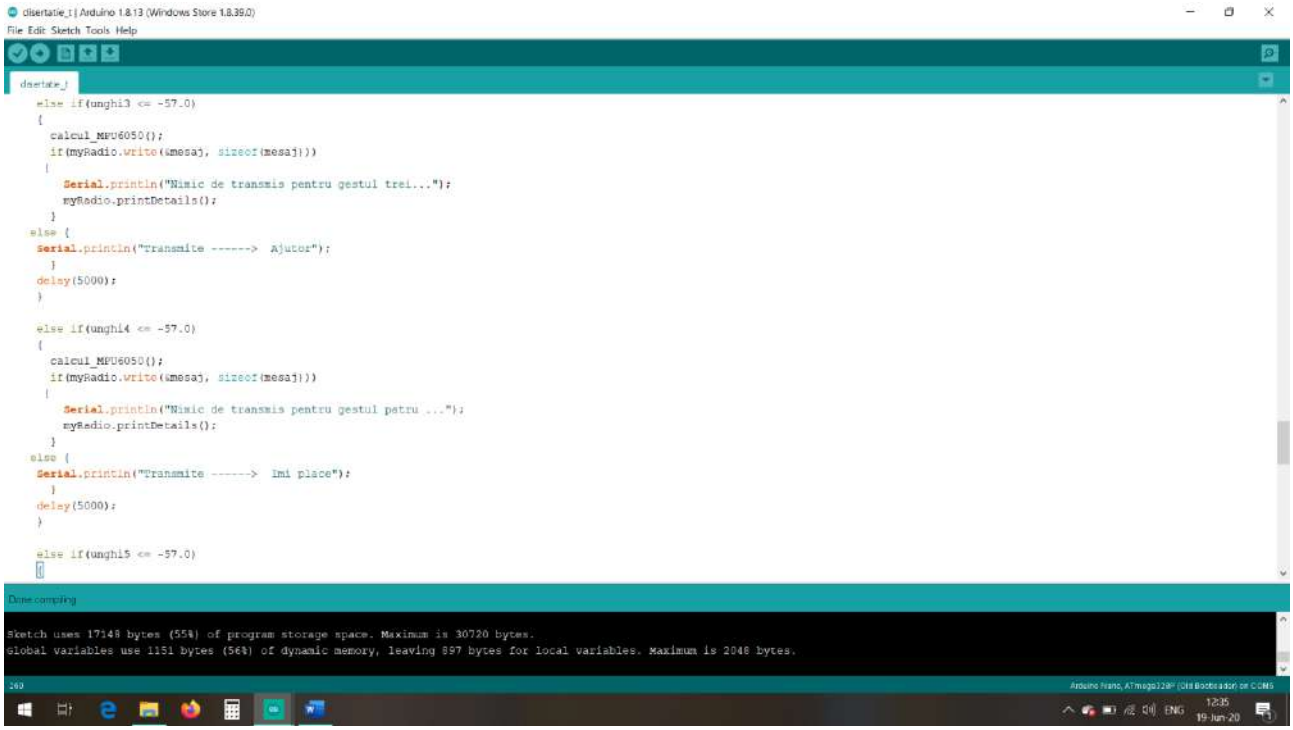

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
delay(200);
Serial.println("");
int flex2 = analogRead(A2);
float flex2_calcul = R_DIV * (VCC / (flex2 * VCC / 1023.0) - 1.0);
unghi2 = map(flex2_calcul, 37.3, 57.0, 0, 90.0);
Serial.println("Valoarea 2: " + String(unghi2));
delay(200);
Serial.println("");
int flex3 = analogRead(A3);
float flex3_calcul = R_DIV * (VCC / (flex3 * VCC / 1023.0) - 1.0);
unghi3 = map(flex3_calcul, 37.3, 57.0, 0, 90.0);
Serial.println("Valoarea 3: " + String(unghi3));
delay(200);
Serial.println("");
int flex4 = analogRead(A6);
float flex4_calcul = R_DIV * (VCC / (flex4 * VCC / 1023.0) - 1.0);
unghi4 = map(flex4_calcul, 37.3, 57.0, 0, 90.0);
Serial.println("Valoarea 4: " + String(unghi4));
delay(200);
Serial.println("");
int flex5 = analogRead(A7);
float flex5_calcul = R_DIV * (VCC / (flex5 * VCC / 1023.0)- 1.0);
unghi5 = map(flex5_calcul, 37.3, 57.0, 0, 90.0);
Serial.println("Valoarea 5: " + String(unghi5));
delay(200);
Serial.println("");
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:34
19-Jun-20

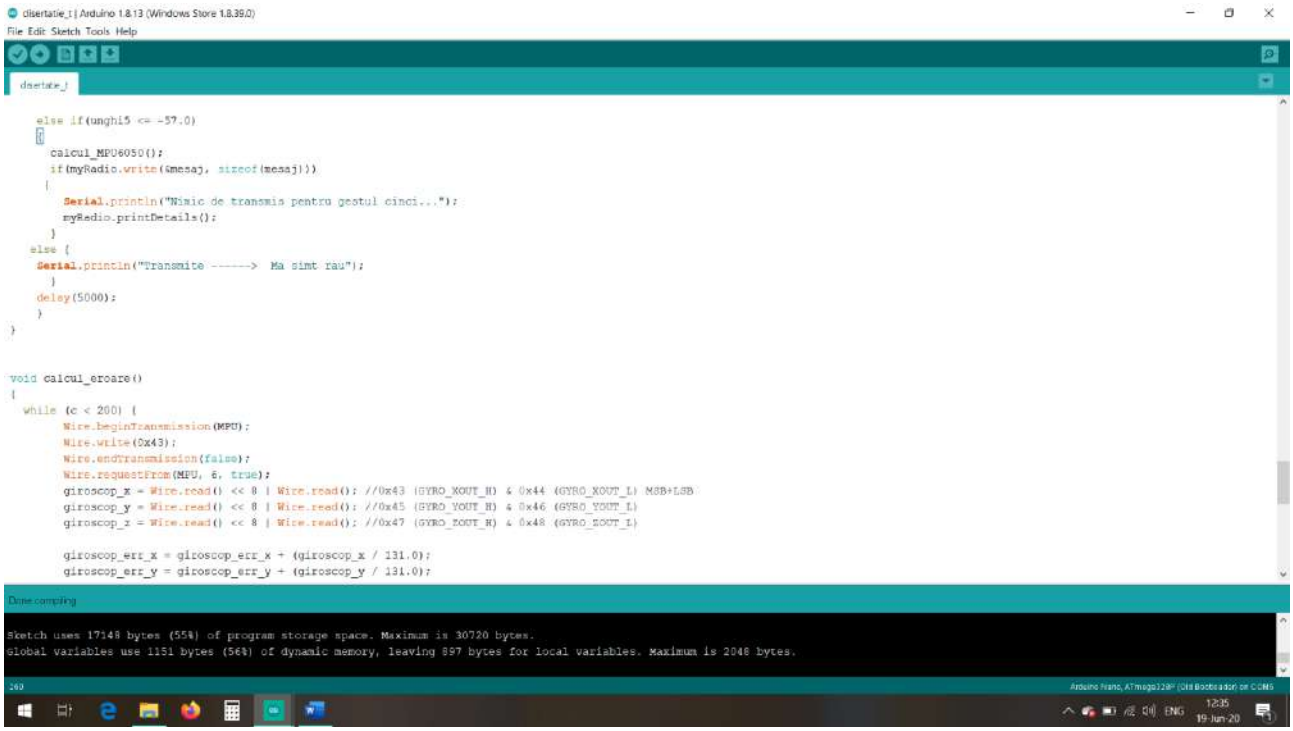

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
count = count+1;
Serial.println("Am terminat ciclul nr.: " + String(count));
Serial.println("");
}
void verificare()
{
if((unghi1 <= -57.0) && (unghi2 <=-57.0))
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru primul gest...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Te rog");
}
delay(5000);
}
else if(unghi2 <= -57.0 && unghi3 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul doi...");
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:34
19-Jun-20

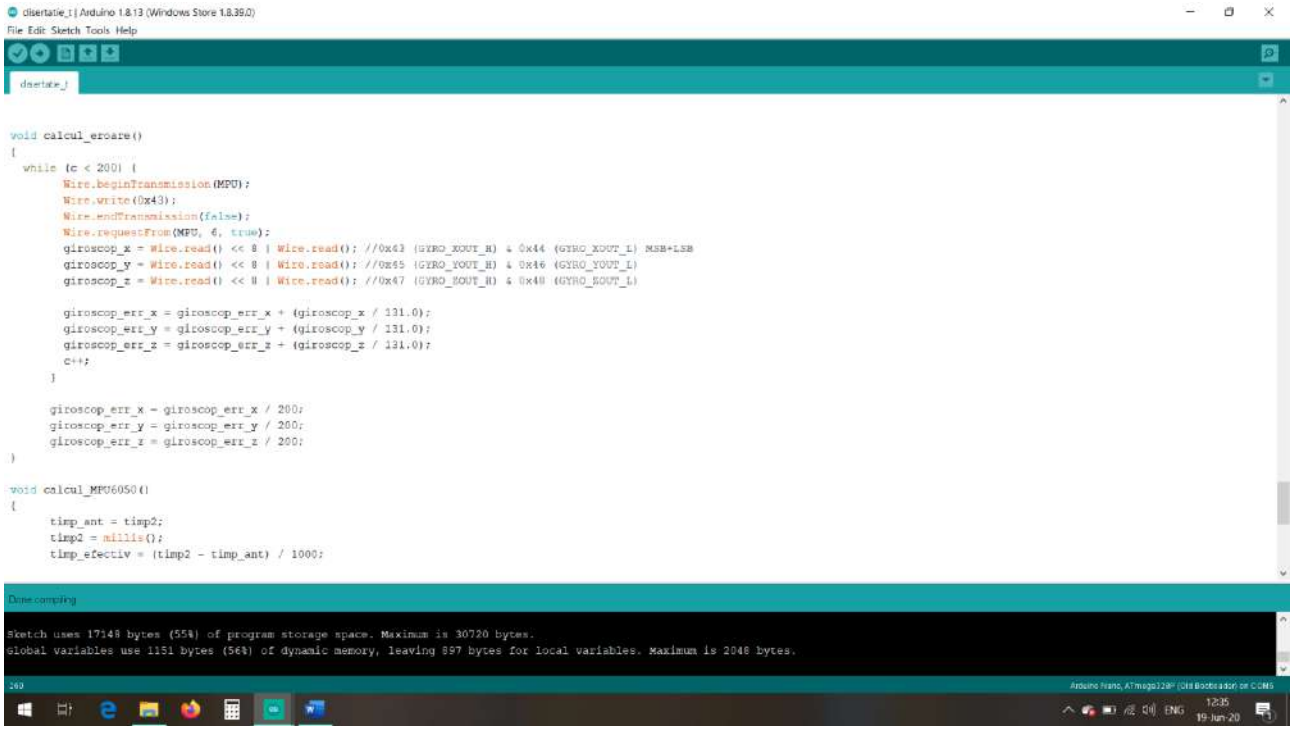

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
else if(unghi2 <= -57.0 && unghi3 >= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul doi...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Nu pot");
}
delay(5000);
}
else if(unghi3 <= -57.0 && unghi4 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul doi...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Pastila");
}
delay(5000);
}
else if(unghi4 <= -57.0 && unghi5 >= -57.0)
{
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG 12:34 19-Jun-20

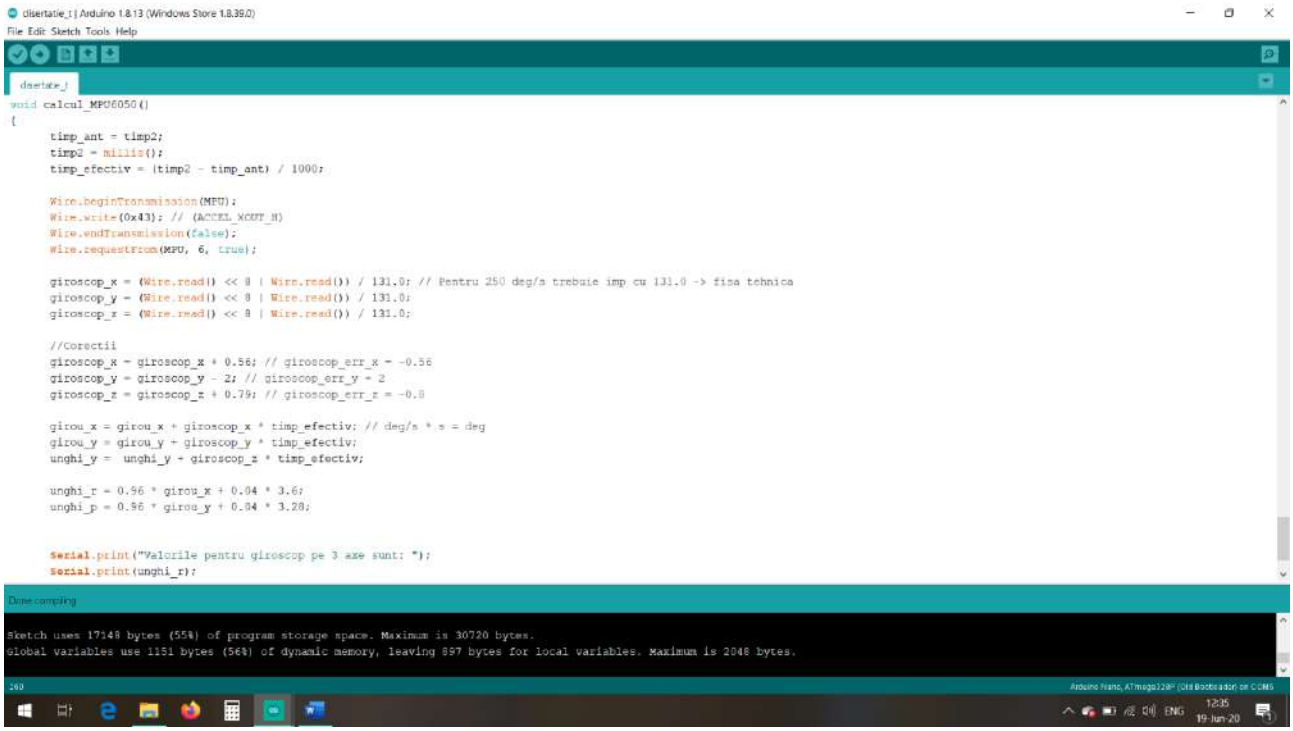

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
}
else {
Serial.println("Transmite ------> Pastila");
}
delay(5000);
}
else if(unghi4 <= -57.0 && unghi5 >= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul doi...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Toaleta");
}
delay(5000);
}
else if(unghi1 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru primul gest...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Multumesc");
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG 12:34 19-Jun-20

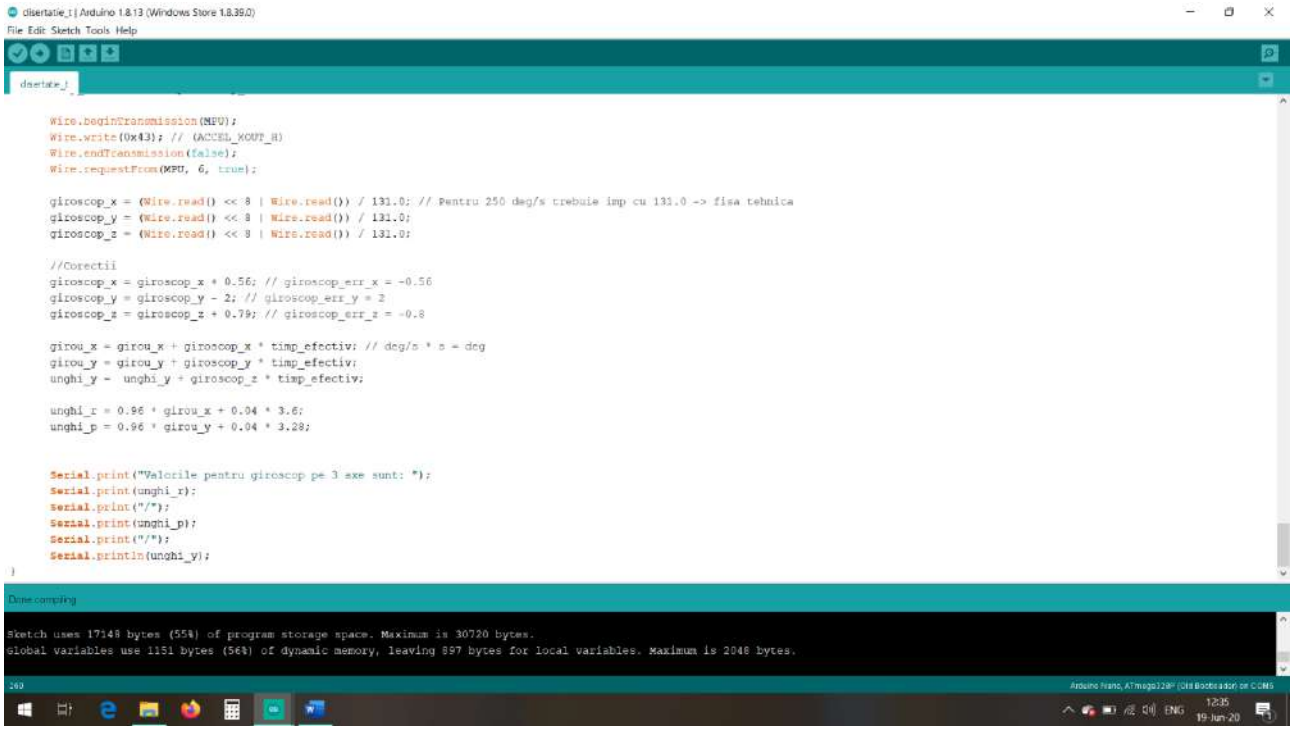

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
}
else {
Serial.println("Transmite ------> Multumesc");
}
delay(5000);
}
else if(unghi2 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul doi...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Apreciez");
}
delay(5000);
}
else if(unghi3 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul trei...");
myRadio.printDetails();
}
else {
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:35
19-Jun-20


disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
else if(unghi3 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul trei...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Ajutor");
}
delay(5000);
}
else if(unghi4 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul patru ...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Imi place");
}
delay(5000);
}
else if(unghi5 <= -57.0)
{
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:35
19-Jun-20

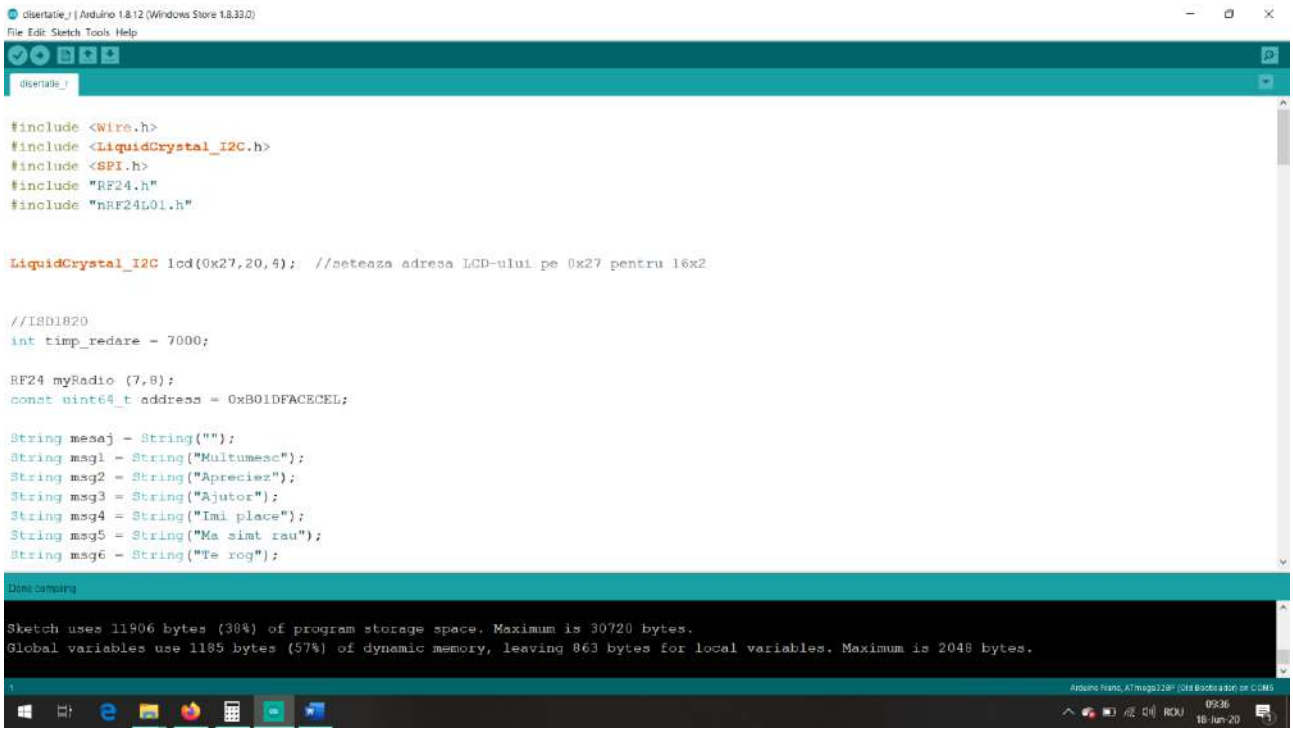

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
else if(unghi5 <= -57.0)
{
calcul_MPU6050();
if(myRadio.write(&mesaj, sizeof(mesaj)))
{
Serial.println("Nimic de transmis pentru gestul cinci...");
myRadio.printDetails();
}
else {
Serial.println("Transmite ------> Ma simt rau");
}
delay(5000);
}
}
void calcul_eroare()
{
while (c < 200) {
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
giroscop_x = Wire.read() << 8 | Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L) MSB+LSB
giroscop_y = Wire.read() << 8 | Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
giroscop_z = Wire.read() << 8 | Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
giroscop_err_x = giroscop_err_x + (giroscop_x / 131.0);
giroscop_err_y = giroscop_err_y + (giroscop_y / 131.0);
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:35
19-Jun-20


disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
void calcul_eroare()
{
while (c < 200) {
Wire.beginTransmission(MPU);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
giroscop_x = Wire.read() << 8 | Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L) MSB+LSB
giroscop_y = Wire.read() << 8 | Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
giroscop_z = Wire.read() << 8 | Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
giroscop_err_x = giroscop_err_x + (giroscop_x / 131.0);
giroscop_err_y = giroscop_err_y + (giroscop_y / 131.0);
giroscop_err_z = giroscop_err_z + (giroscop_z / 131.0);
c++;
}
giroscop_err_x = giroscop_err_x / 200;
giroscop_err_y = giroscop_err_y / 200;
giroscop_err_z = giroscop_err_z / 200;
}
void calcul_MPU6050()
{
timp_ant = timp2;
timp2 = millis();
timp_efectiv = (timp2 - timp_ant) / 1000;
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
Arduino Nano, ATmega328P (Old Bootloader) on COM5
ENG
12:35
19-Jun-20


disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
void calcul_MPU6050()
{
timp_ant = timp2;
timp2 = millis();
timp_efectiv = (timp2 - timp_ant) / 1000;
Wire.beginTransmission(MPU);
Wire.write(0x43); // (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
giroscop_x = (Wire.read() << 8 | Wire.read()) / 131.0; // Pentru 250 deg/s trebuie imp cu 131.0 -> fisa tehnica
giroscop_y = (Wire.read() << 8 | Wire.read()) / 131.0;
giroscop_z = (Wire.read() << 8 | Wire.read()) / 131.0;
//Corectii
giroscop_x = giroscop_x + 0.56; // giroscop_err_x = -0.56
giroscop_y = giroscop_y - 2; // giroscop_err_y = 2
giroscop_z = giroscop_z + 0.79; // giroscop_err_z = -0.8
girou_x = girou_x + giroscop_x * timp_efectiv; // deg/s * s = deg
girou_y = girou_y + giroscop_y * timp_efectiv;
unghi_y = unghi_y + giroscop_z * timp_efectiv;
unghi_r = 0.96 * girou_x + 0.04 * 3.6;
unghi_p = 0.96 * girou_y + 0.04 * 3.28;
Serial.print("Valorile pentru giroscop pe 3 axe sunt: ");
Serial.print(unghi_r);
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
ENG
12:35
19-Jun-20

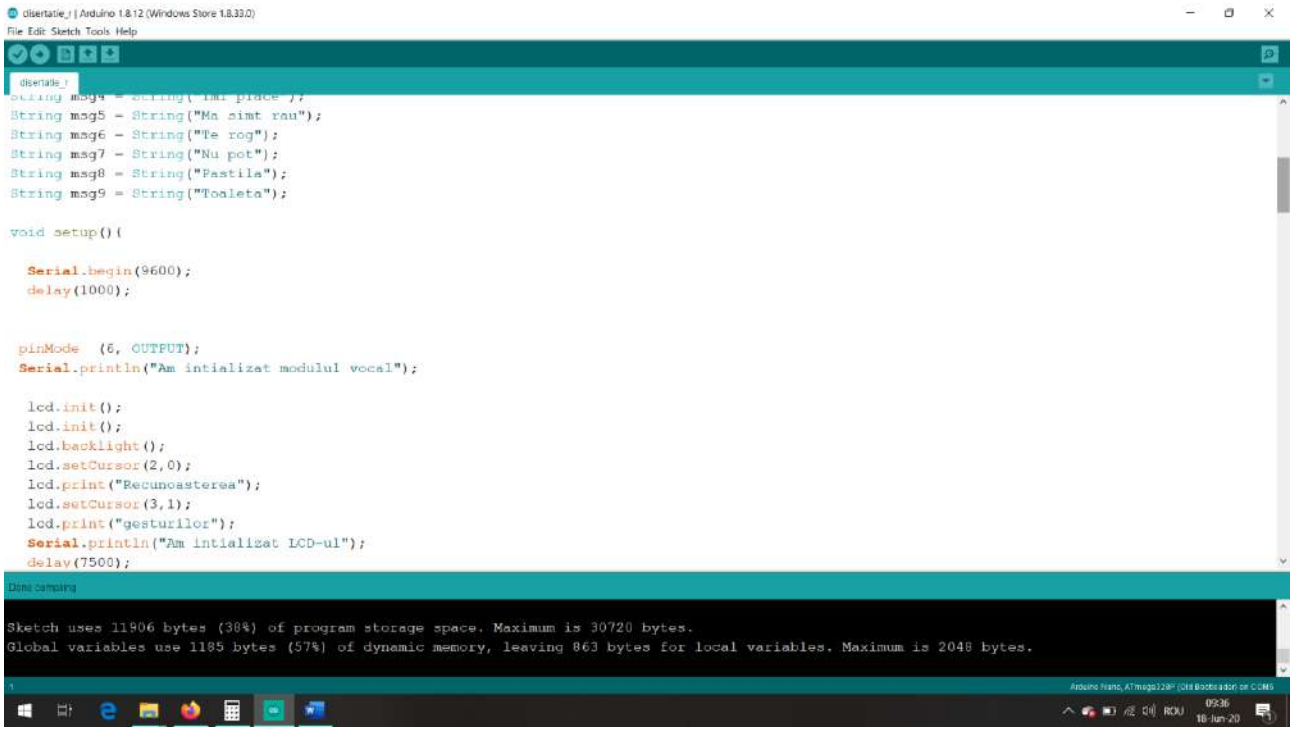

disertatie_1 | Arduino 1.8.13 (Windows Store 1.8.39.0)
File Edit Sketch Tools Help
disertatie_1
Wire.beginTransmission(MPU);
Wire.write(0x43); // (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
giroscop_x = (Wire.read() << 8 | Wire.read()) / 131.0; // Pentru 250 deg/s trebuie imp cu 131.0 -> fisa tehnica
giroscop_y = (Wire.read() << 8 | Wire.read()) / 131.0;
giroscop_z = (Wire.read() << 8 | Wire.read()) / 131.0;
//Corectii
giroscop_x = giroscop_x + 0.56; // giroscop_err_x = -0.56
giroscop_y = giroscop_y - 2; // giroscop_err_y = 2
giroscop_z = giroscop_z + 0.79; // giroscop_err_z = -0.8
girou_x = girou_x + giroscop_x * timp_efectiv; // deg/s * s = deg
girou_y = girou_y + giroscop_y * timp_efectiv;
unghi_y = unghi_y + giroscop_z * timp_efectiv;
unghi_r = 0.96 * girou_x + 0.04 * 3.6;
unghi_p = 0.96 * girou_y + 0.04 * 3.28;
Serial.print("Valorile pentru giroscop pe 3 axe sunt: ");
Serial.print(unghi_r);
Serial.print("/");
Serial.print(unghi_p);
Serial.print("/");
Serial.println(unghi_y);
}
Done compiling
Sketch uses 17148 bytes (55%) of program storage space. Maximum is 30720 bytes.
Global variables use 1151 bytes (56%) of dynamic memory, leaving 897 bytes for local variables. Maximum is 2048 bytes.
ENG
12:35
19-Jun-20

2. Partea de recepție



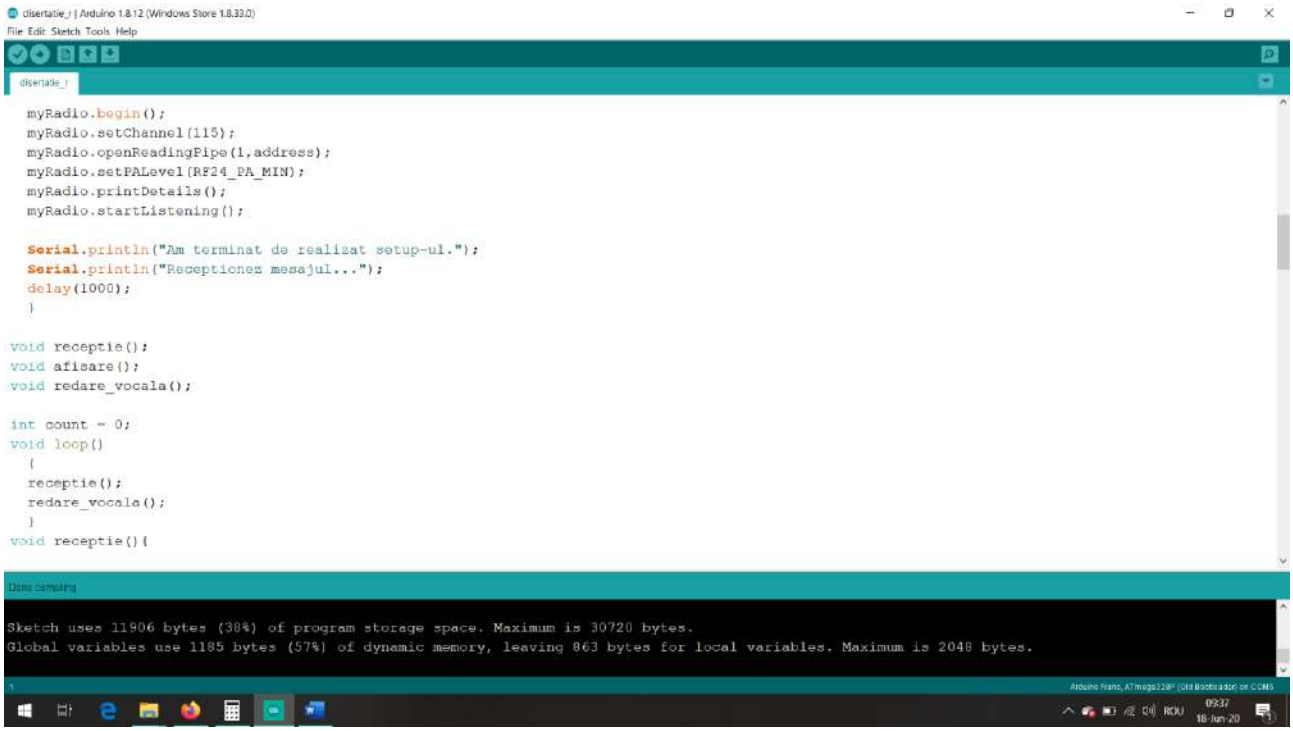
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include "RF24.h"
#include "nRF24L01.h"


LiquidCrystal_I2C lcd(0x27,20,4);  //seteaza adresa LCD-ului pe 0x27 pentru 16x2


//ISD1820
int timp_redare = 7000;

RF24 myRadio (7,8);
const uint64_t address = 0xB01DFACECEL;

String mesaj = String("");
String msg1 = String("Multumesc");
String msg2 = String("Apreciez");
String msg3 = String("Ajutor");
String msg4 = String("Imi place");
String msg5 = String("Ma simt rau");
String msg6 = String("Te rog");
```

```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```



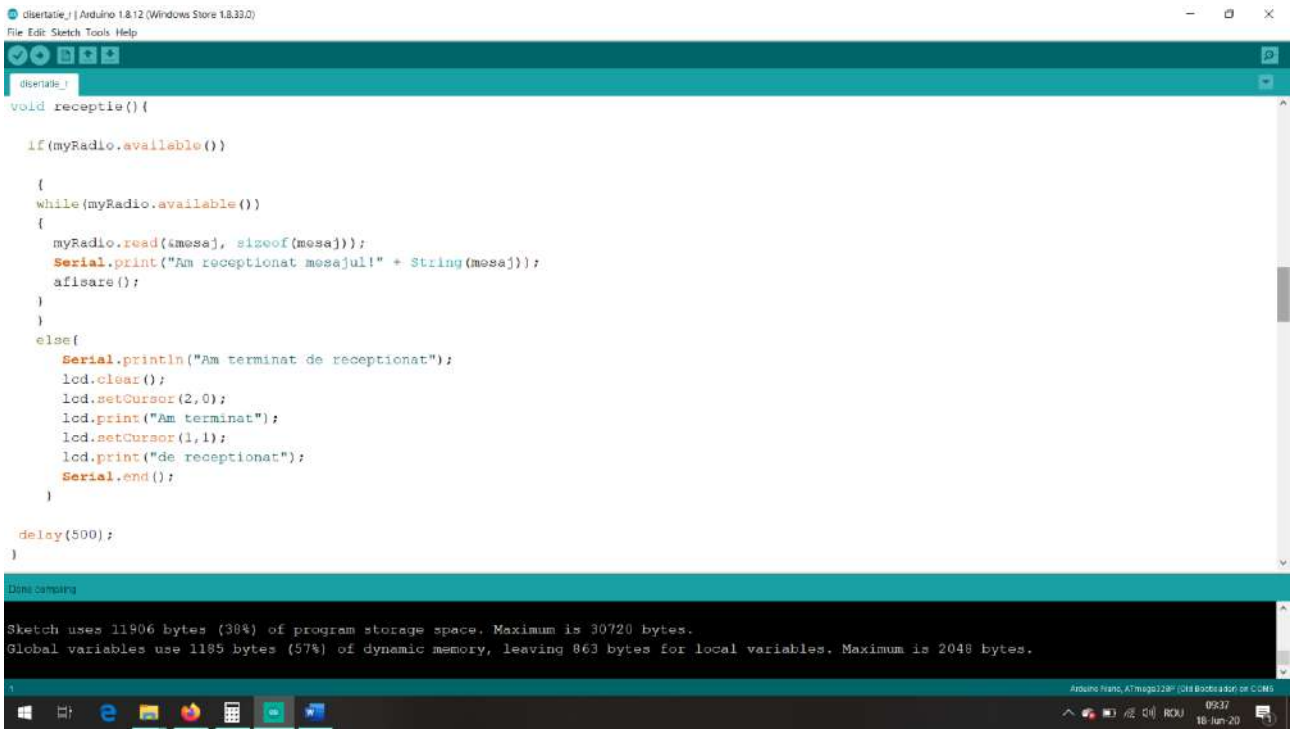
```
String msg5 = String("Ma simt rau");
String msg6 = String("Te rog");
String msg7 = String("Nu pot");
String msg8 = String("Pastila");
String msg9 = String("Toaleta");

void setup(){

  Serial.begin(9600);
  delay(1000);


 pinMode  (6, OUTPUT);
 Serial.println("Am intializat modulul vocal");

  lcd.init();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(2,0);
  lcd.print("Recunoasterea");
  lcd.setCursor(3,1);
  lcd.print("gesturilor");
  Serial.println("Am intializat LCD-ul");
  delay(7500);
```

```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```



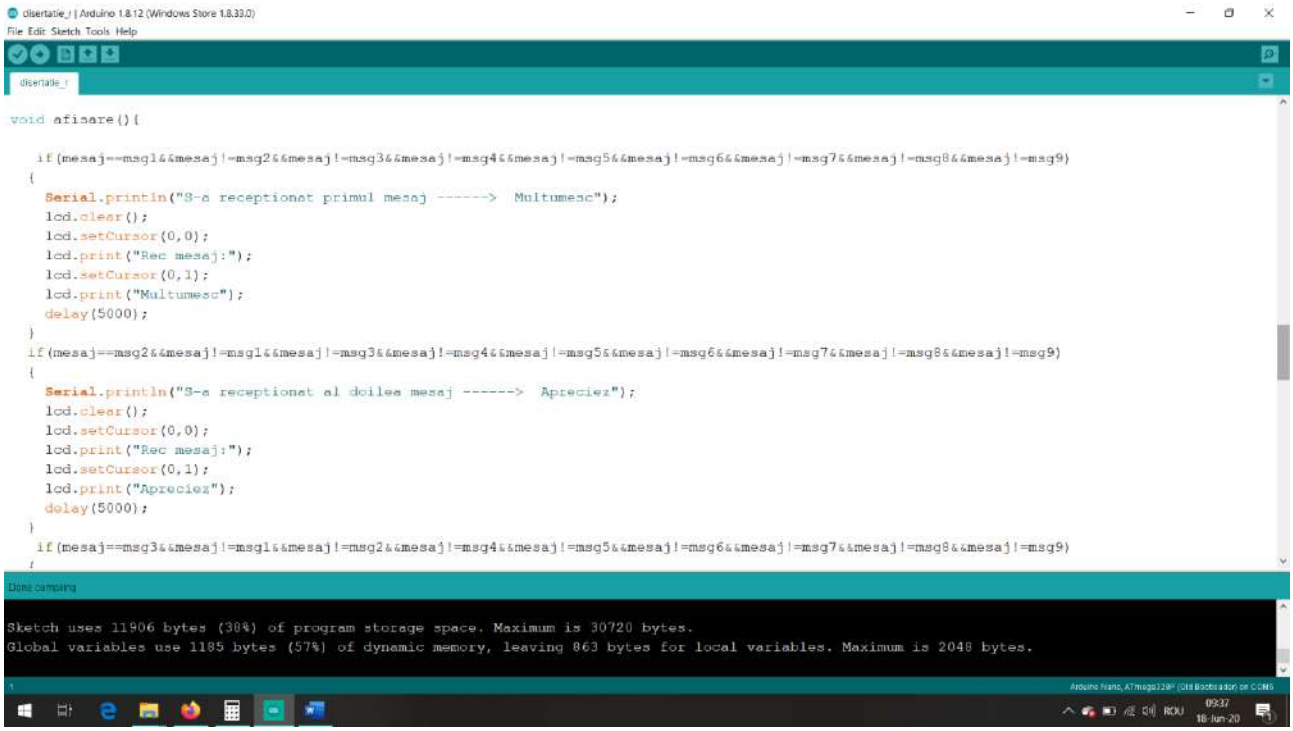
```
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.openReadingPipe(1,address);
  myRadio.setPALevel(RF24_PA_MIN);
  myRadio.printDetails();
  myRadio.startListening();

  Serial.println("Am terminat de realizat setup-ul.");
  Serial.println("Receptionez mesajul...");
  delay(1000);
  }

void receptie();
void afisare();
void redare_vocala();

int count = 0;
void loop()
  {
  receptie();
  redare_vocala();
  }
void receptie(){
```

```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```



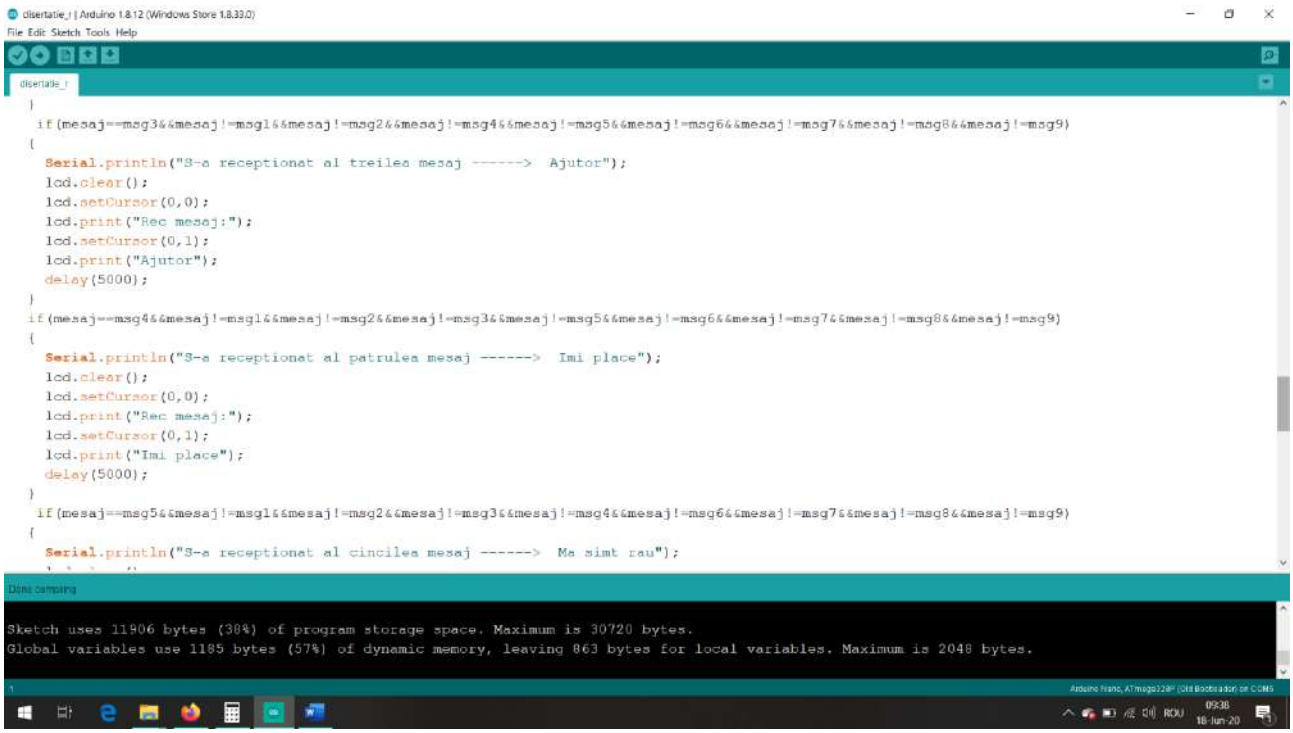
```
void receptie(){

  if(myRadio.available())

   {
   while(myRadio.available())
   {
     myRadio.read(&mesaj, sizeof(mesaj));
     Serial.print("Am receptionat mesajul!" + String(mesaj));
     afisare();
   }
   }
   else{
      Serial.println("Am terminat de receptionat");
      lcd.clear();
      lcd.setCursor(2,0);
      lcd.print("Am terminat");
      lcd.setCursor(1,1);
      lcd.print("de receptionat");
      Serial.end();
    }

 delay(500);
}
```

```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```



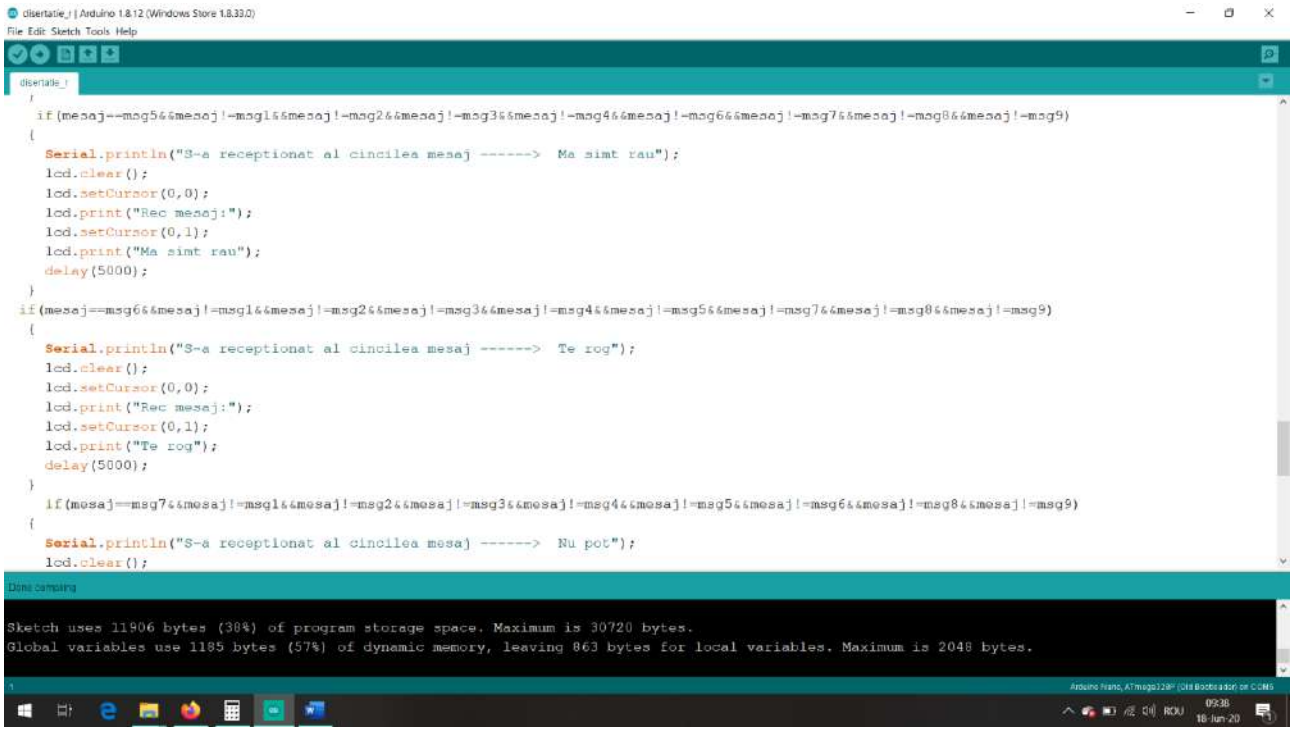
```
void afisare(){

  if(mesaj==msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
  {
    Serial.println("S-a receptionat primul mesaj ------>  Multumesc");
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Rec mesaj:");
    lcd.setCursor(0,1);
    lcd.print("Multumesc");
    delay(5000);
  }
  if(mesaj==msg2&&mesaj!=msg1&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
  {
    Serial.println("S-a receptionat al doilea mesaj ------>  Apreciez");
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Rec mesaj:");
    lcd.setCursor(0,1);
    lcd.print("Apreciez");
    delay(5000);
  }
   if(mesaj==msg3&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
```

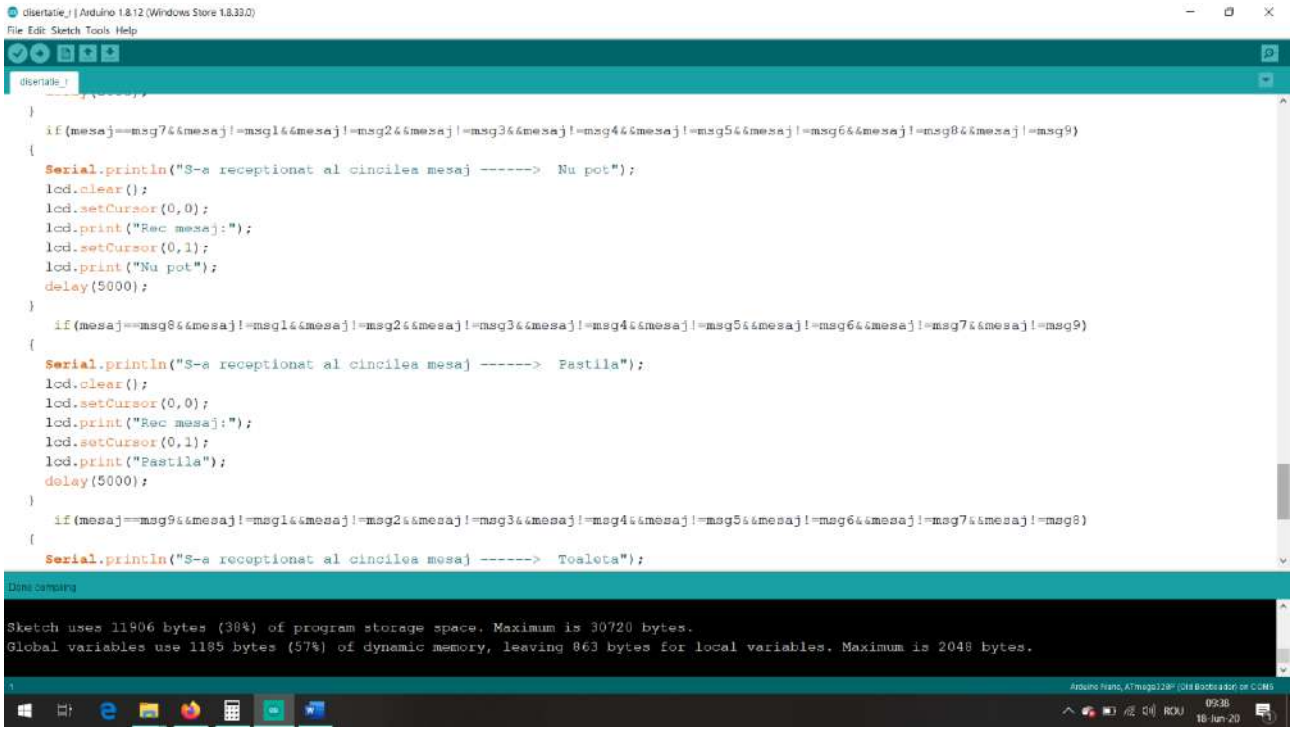
```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```

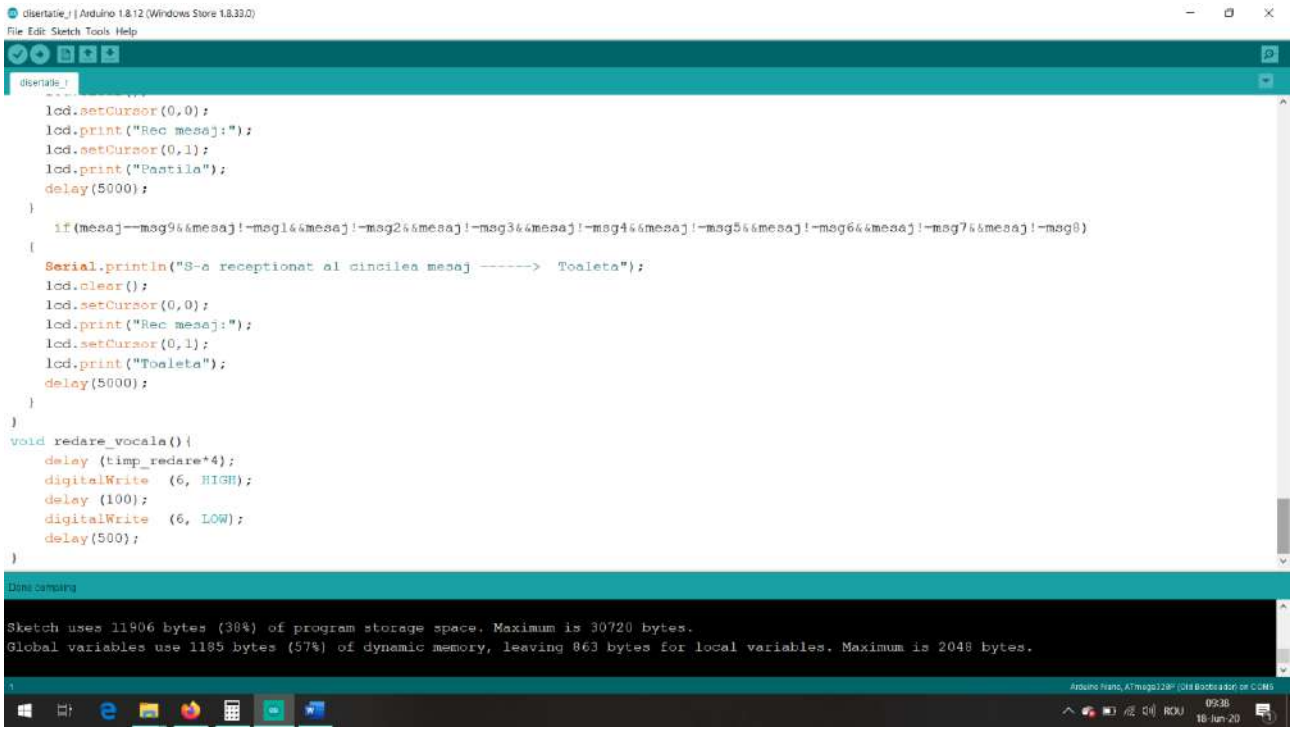


```
   }
   if(mesaj==msg3&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
   {
     Serial.println("S-a receptionat al treilea mesaj ------>  Ajutor");
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("Rec mesaj:");
     lcd.setCursor(0,1);
     lcd.print("Ajutor");
     delay(5000);
   }
   if(mesaj==msg4&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
   {
     Serial.println("S-a receptionat al patrulea mesaj ------>  Imi place");
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("Rec mesaj:");
     lcd.setCursor(0,1);
     lcd.print("Imi place");
     delay(5000);
   }
    if(mesaj==msg5&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
   {
     Serial.println("S-a receptionat al cincilea mesaj ------>  Ma simt rau");
```

```
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
```


disertatie_1 | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
disertatie_1
if(mesaj==msg5&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Ma simt rau");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Ma simt rau");
delay(5000);
}
if(mesaj==msg6&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg7&&mesaj!=msg8&&mesaj!=msg9)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Te rog");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Te rog");
delay(5000);
}
if(mesaj==msg7&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg8&&mesaj!=msg9)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Nu pot");
lcd.clear();
Done compiling
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
ROU
09:38
18-Jun-20


disertatie_1 | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
disertatie_1
}
if(mesaj==msg7&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg8&&mesaj!=msg9)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Nu pot");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Nu pot");
delay(5000);
}
if(mesaj==msg8&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg9)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Pastila");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Pastila");
delay(5000);
}
if(mesaj==msg9&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8)
{
Serial.println("S-a receptionat al cincilea mesaj ------> Toaleta");
Done compiling
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
ROU
09:38
18-Jun-20


disertatie_! | Arduino 1.8.12 (Windows Store 1.8.33.0)
File Edit Sketch Tools Help
disertatie_!
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Pastila");
delay(5000);
if(mesaj==msg9&&mesaj!=msg1&&mesaj!=msg2&&mesaj!=msg3&&mesaj!=msg4&&mesaj!=msg5&&mesaj!=msg6&&mesaj!=msg7&&mesaj!=msg8)
Serial.println("S-a receptionat al cincilea mesaj ------> Toaleta");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Rec mesaj:");
lcd.setCursor(0,1);
lcd.print("Toaleta");
delay(5000);
void redare_vocala(){
delay (timp_redare*4);
digitalWrite (6, HIGH);
delay (100);
digitalWrite (6, LOW);
delay(500);
Done compiling
Sketch uses 11906 bytes (38%) of program storage space. Maximum is 30720 bytes.
Global variables use 1185 bytes (57%) of dynamic memory, leaving 863 bytes for local variables. Maximum is 2048 bytes.
ROU
09:38
18-Jun-20