

- Universitatea Tehnica Iasi
Fac. Automatica si Calculatoare

Project Sisteme Incorporate 2019-2020

Project Based Learning

*“Tell me and I forget
Show me and I may remember
Involve me and I understand”*

Cadru general proiecte:

Sa se dezvolte o solutie de wearable computing care va permite identificarea studentului prezent in laborator , efectuarea unei actiuni specifice si furnizarea datelor despre venit/plecat. Actiunile vor fi activate doar prin prezenta. Actiunile vor fi realizate conform optiunilor studentului.

Actiunile vor putea fi activate si de catre alți colegi acceptati in prealabil.

Modele de proiecte:

In acest spatiu se vor amplasa rapoartele etapelor de proiect conform diagramei de desfasurare.

Nume, prenume , imagine 2x3 cm

Tema de proiect:

Rezumat:

1. Explorare documentara asupa temei,raport sintetic alternative solutie:

.....
.....
.....

Nume:

- Ghitun Stefania

Tema proiect:

- Activare contact pornire masina prin recunoastere vocala

Rezumat:

- Solutia consta in realizarea unui sistem de pornire automata a masinii folosind un Raspberry PI 3 Model B, un serviciu de recunoastere vocala si un microfon. Aceasta va fi realizata ca un nod intr-un cluster de RPIs in vederea automatizarii mai multor functionalitati ale masinii.

Explorare documentara asupra temei:

- Aplicatia va fi activata la intrarea in laboratorul de SI. RPI isi va trimite credentialele catre un RPI master aflat in laborator pentru a fi inregistrat in cluster si a autoriza ce alte noduri ii pot realiza actiunea.
- Actiunea este pornita si opresa printr-un set de doua comenzi "start" si "stop". Pentru aceasta va fi nevoie de un periferic sa indeplineasca functia de microfon. In urma explorarii unor solutii alternative am concluzionat ca cel mai facil de utilizat e un periferic cu USB cum ar fi o camera cu microfon sau un microfon USB (care e mai costisitor si greu de obtinut prin comparatie).
- O data conectat si verificat microfonul, aplicatia va produce ca rezultat un fisier .wav care va fi procesat de un serviciu de recunoastere vocala. In aceasta privinta exista multe alternative, atat online cat si offline. Jasper functioneaza offline, dar prioritizeaza

acuratetea pentru viteza si din moment ce sistemul necesita in mod implicit conexiune la internet am optat pt un API online. Dintre optiunile posibile se enumera [Alexa](#) sau [Siri](#) dar aceastea au dezavantajul de a necesita un proces extensiv de setup si ofera o gama mai larga de servicii decat simpla conversie voce-text, astfel un API usor in resurse e o alegere mult mai logica. Un astfel de API este [solutia oferita de Steven Hickson](#) ce are la baza serviciul de recunoastere vocala Google, dar dezvoltat special pt sistemele linux si RPI.

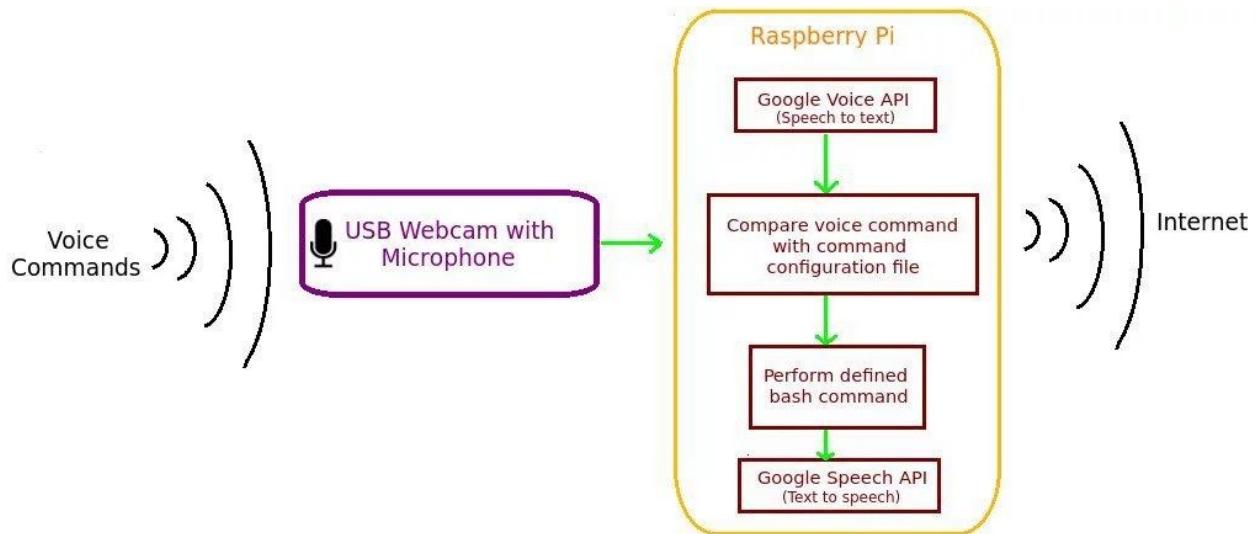
- API-ul dezvoltat de Steven vine cu o serie de configurari cum ar fi utilizarea unui cuvant cheie sau optiunea de a cauta pe youtube, dar acestea sunt redundante pt proiectul prezentat, iar singurele optiuni ce vor fi instalate sunt modul de ascultare continuu si modul mut pentru ca nu e necesar ca API-ul sa raspunda vocal.
- Tot in fisierul de configurare se pot adauga comenzi bash ce vor fi executate daca textul convertit se potriveste cu cel din lista. Vor fi adaugate doua cuvinte cheie "start" si "stop" fiecare indicand catre un script de aprindere sau stingere a unui led.

Raport sintetic:

- Hardware folosit:
 - Raspberry Pi 3 Model B
 - Camera web cu usb
 - LED



- Software folosit:
 - Serviciul de recunoastere vocala al lui Steven Hickson
- Principiu de functionare:
 - RPI asculta incontinuu comenzi prin microfon. Fisierele .wav sunt procesate pe internet de catre serviciul Google de recunoastere vocala prin intermediul API-ului oferit de Steven Hickson
 - Textul primit este comparat cu o serie de fraze dintr-un fisier de configurare si daca se potrivesc, se executa scriptul bash
 - Scriptul bash porneste o aplicatie python ce aprinde sau stinge un led in functie de argumentele de intrare.



A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The window shows a nano text editor with the file '/home/pi/.commands.conf'. The content of the file is as follows:

```

!filler==0
#!duration==2
#!com_dur==3
#!hardware==plughw:1,0
#Here are the commands
show me==/home/pi/AUI/Imaging/test 2
track me==/home/pi/AUI/Imaging/test 1
download==download ...
play $1 season $2 episode $3==playvideo -s $2 -e $3 $1
download $1 season $2 episode $3==download $1 s$2e$3
play==playvideo -r -f ...
multiple==playvideo -r -m -c 5 ...
check==ping google.com
download==download ...

```

The terminal window includes a menu bar with File, Edit, Tabs, Help, and tabs for pi@raspberrypi... and pi@raspberrypi... The bottom status bar shows the time as 08:57 and various icons.

Solutii alternative:

- Un microfon analog, dar din moment ce RPI nu are ADC, ar fi necesar si un modul de ADC
- Servicii de recunoastere vocala precum Jasper (functioneaza offline, dar nu e foarte precis), Alexa sau Siri (sunt API-uri majore cu multe functionalitati si un setup mult mai complex decat solutia oferita)

Ciorna de solutii:

- Hardware folosit:
 - Raspberry Pi Model 3 B
 - iPhone
 - LED 3.3v
 - 100 ohm rezistenta
- Software folosit:
 - SiriControl python framework
- Implementare:
 - Primul pas reprezinta inregistrarea unui cont gmail cu securitate joasa care va fi folosit de Siri pentru a trimite comenzi interpretate prin gmail notes.
 - Apoi contul google va fi adaugat in iPhone si optiunea de a adauga note in contul inregistrat.
 - Se instaleaza SiriControl si se realizeaza un nou modul python de aprindere si stingere a unui LED ce reprezinta contactul masinii.

```
pi@raspberrypi:~ $ sudo apt-get install git-core
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  git-core
0 upgraded, 1 newly installed, 0 to remove and 96 not upgraded.
Need to get 1,410 B of archives.
After this operation, 8,192 B of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian stretch/main armhf git-core all 1:2.11.0-3+deb9u2 [1,410 B]
Fetched 1,410 B in 0s (2,109 B/s)
Selecting previously unselected package git-core.
(Reading database ... 122983 files and directories currently installed.)
Preparing to unpack .../git-core_1%3a2.11.0-3+deb9u2_all.deb ...
Unpacking git-core (1:2.11.0-3+deb9u2) ...
Setting up git-core (1:2.11.0-3+deb9u2) ...
pi@raspberrypi:~ $ git clone https://github.com/theraspberryguy/SiriControl-System
Cloning into 'SiriControl-System'...
remote: Counting objects: 52, done.
remote: Total 52 (delta 0), reused 0 (delta 0), pack-reused 52
Unpacking objects: 100% (52/52), done.
pi@raspberrypi:~ $ scrot
```

Nume : Pipirig Mihai

Tema proiect: Realizarea unui mecanism de control a luminii din habitacul pe baza conditiilor de mediu.

Idee: lumina multicolora (RGB)

Moduri de functionare:

Daca soferul nu este prezent in masina sistemul se va opri

Este prezent sistemul va porni in modul auto

-auto:

-In functie de gradul de iluminare exterior

- manual
 - 3 trepte de iluminare
- stins
- aprins
- intermediar (folosind un potentiometru)

- gradient circular

In prezent, daca vorbim de solutii pentru iluminat trebuie sa luam in calcul si sa invatam din experienta celor de la OSRAM, organizatie ce ocupa locul 1 mondial in iluminatul auto, drept pentru care in fiecare an, jumata din automobilele fabricate in toata lumea sunt echipate cu lampi OSRAM.

Acestia lucreaza la multiple proiecte si ofera solutii de iluminat pentru diferite domenii de activitate precum:

- cel mai important, iluminatul auto
- Digital Platforms, IOT software, smart light
- Horticulture Lighting
- Hospitality Lighting
- Entertainment Lighting

Un subiect care se discuta si pentru care se investesc suficienti bani in cercetare este (cred eu interesant) Human Centric Lightning (HCL), subiect care este abordat si de cei de la OSRAM.

The human biological clock is controlled via the light

Scientists have been studying the biological impacts of light for decades. But it was not until 2002 that they discovered ganglion cells in the retina that are not used for seeing. The newly identified cells respond most sensitively to visible blue light and set the biological clock that synchronises our bodies with the external cycle of day and night.

A major output of the biological clock system is the production of the hormone melatonin – a “sleep hormone”. This production in the pineal gland varies with the time of day. Melatonin is secreted at night and has minimal levels during daytime. A larger melatonin suppression, triggered by light exposure, often coincides with increased feelings of alertness and higher sustained attention.

HCL in education

Schools are excellent for tuneable white light features. Here the light is used not only to control daily rhythms for pupils and staff, but also to improve alertness during tests and concentration tasks.

The teacher may switch on an intensive, cool white light during these activities, or a warm white, dimmed light for relaxation and group talk.

The right light in the morning, with sufficient brightness and blue-enriched, can help to get you ready for the day. Especially in education, a conscious mind is important for a good concentration during the lessons. It doesn't matter if the person is an elementary scholar or a teacher. Both can benefit from an optimized lighting environment in a direct or indirect way.

HCL in healthcare

Healthcare environments are well suited to implement lighting cycles containing sunrise, sunset and daylight simulations. The effects on the patients or residents are higher activity levels during daytime, better sleep at night, shorter recovery times and reduced intake of anti-depressants.

HCL in offices

For countries with little daylight during the winter months, “tunable white light” luminaires may reduce winter depressions and other seasonal affective disorders. The same lighting settings may also give short-term concentration and alertness effects, when applied correctly.

The timing, duration and spectral composition of the light exposure all play important roles in these non-image forming effects. Moreover, research has shown that these effects may depend on the environmental context, type of activity, person characteristics and employees' momentary level of fatigue.

<https://glamox.com/hcl/> ----HCL Demo

Pentru proiectul propriu-zis am identificat necesitatea urmatoarelor componente, precum si a alternativelor aferente

Componente Hardware

- partea de interconectivitate

Raspberry Pi Zero W

- componenta de comanda si control

XMC 2GO

XMC 1100 Bootkit

XMC 4700

Adafruit Metro Mini 328 - 5V, 16 MHz

Teensy 3.5

Arduino Uno

- componenta actuatoare

Modul SparkFun Lumenati în linie cu 8 LED-uri

Inel de LED-uri RGB WS2812 cu 16 LED-uri

RGB LED HAT IC pentru Raspberry Pi

- hw auxiliar

- modul RTC, in cazul in care componenta de control nu are inclus un astfel de modul
- sensor PIR de prezenta sau HC-SR 04 ultrasunet

- Senzor de Lumina Ambientala OPT3001 sau

Senzor Digital de Lumină IR, UV și Vizibil SI1145

Modul Senzor de Lumină Ambientală TEMT6000

Senzor de Lumină UV ML8511

Modul Senzor de Lumină Ambientală MAX44009

Componente Software

- Raspbian

- python interpreter

- pentru a implementa comunicatia cu partea de control. In functie de decizia aleasa in privinta hw folosit pentru acest proiect, voi opta pentru o comunicatie folosind RS232 cu fir, Bluetooth, I2C, SPI

- apache server

- html: pentru a adauga elemente grafice pe pagina principala a server-ului

- css: pentru a stiliza elementele grafice

- php: folosit in special pentru invocarea unor scripturi ca urmare a interactiunii utilizatorului cu pagina web

- in functie de componenta hw folosita voi opta pentru urmatoarele sdk-uri

- The Teensy Loader Application

- Dave

- Arduino IDE

Ciorna de solutie:

Proiectul poate fi impartit in 3 componente

- Server

- Apache

- Butoane pentru stabilirea modului de lucru

- Butoane pentru stabilirea nivelului de iluminare

- Interconectare

- Scripturi python pentru transmisia mesajelor pe seriala

- Controlul actuatoarelor si senzorilor

Folosesc XMC 1100 impreuna cu mediul de programare Dave, dar si o componenta mult mai importanta si anume RTX.

Comunicatia cu server-ul va fi realizata utilizand interfata seriala prin intermediul careia se vor transmite mesaje care codifica schimbarea modului de lucru, setarea unui anumit nivel de iluminare, etc

Actualizarea permanenta a datelor de la senzori va reprezenta alt task. Acest thread va interoga senzorii de prezenta si cel de lumina si va scrie in niste variabile globale informatiile colectate.

Activitatea actuatoare va fi un alt task si va fi responsabila pentru transferul datelor catre Inel-ul de LED-uri RGB WS2812 in functie de modul de functionare receptionat de alt task.

1405 A

Rotariu Cosmin -paul

Directia si controlul directiei unei masini pe baza unui servomotor .

Directie Masina

Explorare Documentara asupra temei :

Sistemul de directie este unul dintre cele mai complexe sisteme instalate pe un autovehicul, acesta avand multiple roluri:

- trebuie sa nu influenteze pozitia rotilor.
- trebuie sa nu fie influentat de oscilatiile suspensiilor la denivelarile drumului.
- trebuie sa nu transmita volanului socrurile primite de la roti.
- trebuie sa permita soferului sa efectueze schimbarea directiei de mers rapid si fara efort.

Unghiurile rotilor directoare ajuta la stabilitatea acestora, adica la tendinta de a reveni si de a-si pastra pozitia neutra in cazul in care au fost deviate de la aceasta prin rotirea volanului de catre sofer sau datorita factorilor externi (denivelari ori obstacolele de pe drum).

Unghiul de convergenta. Acest unghi asigura paralelismul planurilor de miscare al rotilor directoare.

Unghiul de inclinare laterală a pivotului. Acesta are rolul de a mari tendinta rotilor directoare de reveni la pozitia pentru mersul in linie dreapta si de a usura manipularea volanului in cazul in care acestea au fost scoase din pozitia neutra.

Unghiul de inclinare longitudinala a pivotului. Acest unghi este cunoscut si drept „*unghi de fuga*” si asigura stabilitatea autovehiculului in timpul mersului.

Masinile de generatie mai noua si toate cele construite la momentul actual au sisteme de directie servo-asistate, sisteme care ajuta soferul sa schimbe pozitia rotilor fara efort inclusiv atunci cand autovehiculul sta pe loc.

Defectiuni ale sistemului de directie:

- **manevrarea greoarie a volanului.** Aceasta poate fi determinata de defectarea pompei de presiune a servodirectiei, de faptul ca pompa nu baga suficient ulei sau datorita griparii pivotilor.
- **resimtirea loviturilor puternice in volan.** Aceasta defectiune este determinata de prezența aerului in instalatia hidraulica.
- **aparitia zgomotului in timpul virarii rotilor.** Acest simptom nu poate fi considerat neaparat o defectiune deoarece este dat de o cantitate prea mica de ulei in instalatie, lucru care se remediaza imediat ce vasul a fost reumplut pana la limita normala de functionare.
- **aparitia jocului la volan.** Jocul maxim la volan admis conform legislatiei in vigoare este de 15 grade, orice depasire a acestuia reprezentand o defectiune a sistemului de directie. Acest joc poate aparea datorita angrenajelor prea uzate in caseta de directie, datorita uzurii capetelor de bara sau a pivotilor sau datorita jocului aparut la articulatia cardanica a volanului.

Solutii: sistem de translatie a miscarii volanului

Sistem bazat pe actuatori electrici de schimbare a unghiului de virare (servo/ actuatori liniari bazati pe translatie)

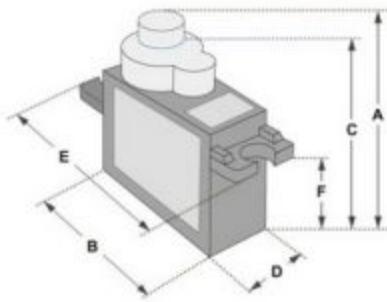
Sistem bazat pe steppere

SERVO MOTOR SG90

DATA SHEET



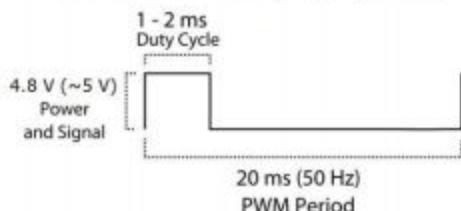
Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

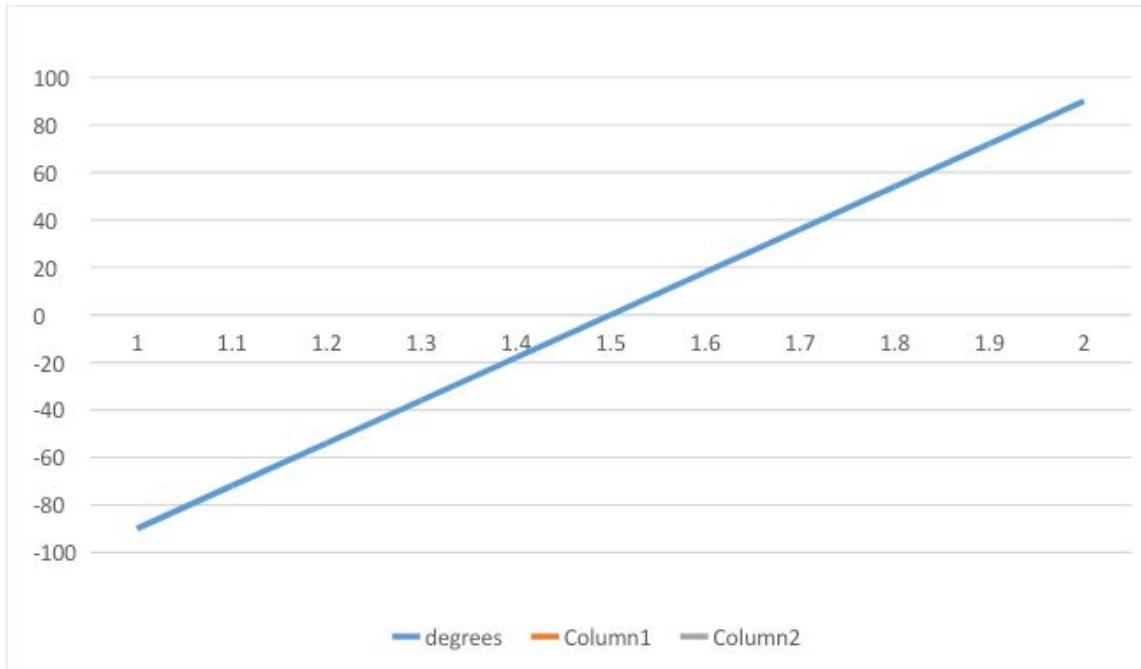


Dimensions & Specifications	
A (mm)	: 32
B (mm)	: 23
C (mm)	: 28.5
D (mm)	: 12
E (mm)	: 32
F (mm)	: 19.5
Speed (sec)	: 0.1
Torque (kg-cm)	: 2.5
Weight (g)	: 14.7
Voltage	: 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

PWM=Orange (↑↑)
Vcc = Red (+)
Ground=Brown (-)





Ciorna Solutii :

Folosind semnale pwm cu frecventa de 50 HZ iar latchul pozitiv fiind intre 1ms si 2ms putem seta un grad Servomotorului astfel incat sa poata realiza steeringul la roti

Aceasta componenta va primi mesaje de la componenta VOLAN (aceasta va fi simulata prin comenzi pe seriala .)(steering-Wheel) pentru a practica fiecare grad in functie de comanda primita de la aceasta componenta.

Mesajele primite vor fi valori intregi , chiar si reale(float) care cu restrictia :

→ -90<x<90

PINi necesari pentru motor :

- > 1 PIN pentru PWM
- > 1 PIN VCC (5V)
- > 1 PIN GND

Componente necesare pentru “Directia masinii” :

- 1 servomotor
- 1 raspberry pi //pentru testare cu serverul
- Python
- Pigpio lib (Pentru a elimina greselile de PWM generation) //in caz de facem pe raspberry
- 1 xmc4800 cu un bluetooth care sa primeasca intructiuni de la raspberry pi si feedback
- Dave pentru a controla servomotorul prin comenzi seriale

server.py

```
Import pigpio
```

```
pi = pigpio.pi()

G=2

D=3


from flask import Flask, request,
render_template, redirect
app = Flask(__name__)

@app.route('/', methods=['POST'])

def post():

    str = request.form['degree1']

    print(str,str2)

    pi.set_servo_pulsewidth(G,int(100/9*int(str)+500))

    return
    render_template("index.html",error=str,error1=str2)


@app.route("/")

def home():

    return render_template("index.html")


if __name__ == "__main__":
    try:
        app.run(host='0.0.0.0')
    except KeyboardInterrupt:
        #raise
        pi.set_servo_pulsewidth(G,0)
```

```
pi.set_servo_pulsewidth(D,0)  
pi.stop()
```

index.html

```
<!DOCTYPE html>  
  
<html>  
  
<body>  
  
<meta name="viewport" content="width=device-width, initial-scale=1">  
  
<style>  
  
.slidecontainer {  
  
    width: 100%;  
  
}  
  
  
.slider {  
  
    -webkit-appearance: none;  
  
    width: 100%;  
  
    height: 25px;  
  
    background: #d3d3d3;  
  
    outline: none;  
  
    opacity: 0.7;  
  
    -webkit-transition: .2s;  
  
    transition: opacity .2s;  
  
}  
  
  
.slider:hover {  
  
    opacity: 1;  
  
}  
  
  
.slider::-webkit-slider-thumb {  
  
    -webkit-appearance: none;  
  
    appearance: none;
```

```

width: 25px;
height: 25px;
background: #4CAF50;
cursor: pointer;
}

.slider::-moz-range-thumb {
width: 25px;
height: 25px;
background: #4CAF50;
cursor: pointer;
}

</style>

<form name="auto" method="post">

{%
  if error %
    <input type="range" name="degree1" min="0"
value={{error}} max="180" class="slider" >
  {%
    else %
      <input type="range" name="degree1" min="0" max="180"
class="slider" >
    {%
      endif %
    }
  {%
    endif %
  }
</form>

<script type="text/javascript">
  window.onclick=function(event) {
    var auto = setTimeout(function(){ autoRefresh(); }, 100);

    function submitform() {
      document.forms["auto"].submit();
    }
  }
</script>

```

```

        }

    function autoRefresh() {
        clearTimeout(auto);
        auto = setTimeout(function() { submitform(); autoRefresh(); }, 100);
    }
}

</script>
</body>
</html>

```

Nume, prenume:

- Vereştiuc Daniel-Silviu

Tema de proiect:

- Reglarea pozițiilor oglinzilor

Rezumat:

Modulul va regla automat poziția oglinzilor retrovizoare în funcție de fiecare șofer care va conduce mașina. Poziția preferată a oglinzilor va fi stocată în memoria programului, pentru fiecare șofer ce va utiliza (sau a utilizat) la un moment dat mașina.

Explorare documentara asupa temei (raport sintetic alternative solutie):

Solutia constă în realizarea unui sistem de pornire automată a sistemului de poziționare a oglinzilor unei mașini, folosind un Raspberry PI 3 Model B, cu ajutorul unor servomotoare (sau cu ajutorul unor motoare de curent continuu cu feedback).

Butonul “virtual” ce controleaza oglinzelile are 3 pozitii, stanga, dreapta si in mijloc (oprit), in ultima situatie, reglarea oglinzilor este oprită, aceasta pozitie dezactivand orice impuls de

actionare a oglinzii, pentru a nu atinge accidental acel buton, aceasta fiind defapt o masura de siguranta impotriva actionarii nedorite din timpul mersului. Acest buton apare in modul “Ajustare”, pentru ca soferul actual sa poata regla preferintele.

Pentru initiere, masina va avea o pozitie “default” a oglinzilor. Daca nu se adauga un “sofer principal”, atunci ele vor ramane in pozitia prestabilita. Dupa ce se adauga un “sofer principal”, ele se vor deschide de fiecare data conform preferintelor selectate in meniu. Ulterior, acesta va avea posibilitatea de a adauga si alti soferi, care vor putea regla pozitia preferata de ei, ca oglinzelile sa se seteze in pozitia respectiva in momentul conducerii masinii.

Moduri de functionare:

- Oprit -> dacă şoferul nu este prezent în maşină sistemul va avea oglinzelile în poziţia ”închis”.
- Auto -> modulul detectează automat identitatea şoferului şi reglează în mod automat poziţia oglinzilor.
- Ajustare -> Şoferul va avea posibilitatea de a modifica preferinţele.
- Manual -> dacă şoferul nu este în memorie, atunci acesta va trebui să seteze singur poziţia oglinzilor, care va fi memorată pentru utilizarea ulterioară a maşinii.

Nume, prenume:

- Florea Oana

Tema de proiect:

- Controlul geamului unei maşini (Window Lifter)

Rezumat:

Modulul va controla pozitia geamului unei masini (Window Lifter). Modulul are patru actiuni posibile: auto up/down si manual up/down.

Explorare documentara asupa temei (raport sintetic alternative solutie):

Soluţia constă în realizarea unul sistem de control remote a geamurilor unei maşini, folosind un Raspberry PI 3 Model B, cu ajutorul unor motoare de curent continuu.

În prezent, pentru window lifter ele de pe masini, nu se mai folosesc senzori Hall pentru stabilirea pozitiei geamului, aceştia au fost înlocuiţi cu algoritmul SLP (Sensorless Positioning). Acesta împarte cursa geamului într-un număr de regiuni (ex: 32 de regiuni pentru maşinile din gama Hyundai). De fiecare data cand se porneste maşina, geamul este decalibrat. Pentru a se calibra, acesta trebuie sa invete pozitia de hard block prin efectuarea unei curse complete pana la inchiderea geamului. În timpul calibrării, WL învaţă curentul necesar efectuarii mişcării pentru

fiecare din cele 32 de zone ale geamului. După calibrare, în funcție de curentul consumat de motor, se poate afla și poziția geamului.

Functionalitati ale Window Lifterului:

- Anti-pinch

Aceasta este una din cele mai importante functionalitati, intrucat este prevazut de lege ca un geam nu poate efectua miscare de auto-up fara un algoritm de anti-pinch. În cadrul calibrarii se invata curentii specifici fiecarui segment din cursa geamului. La executia unei comenzi de auto-up, se compara curentul consumat de motor pentru fiecare segment si se compara cu cel de referinta. Daca diferența dintre acestia depaseste un threshold prestabilit, atunci geamul va face reverse.

- Shortstroke

Aceasta functionalitate se regaseste pe masinile decapotabile si Coupé-uri deoarece acestea nu au cheder în partea superioara a portierei, si în cazul unei inchideri complete a geamului, acesta intra direct în plafonul masinii. Din acest motiv, atunci cand geamul este inchis complet si soferul vrea sa deschida portie, geamul ar putea fi deteriorat.

Algoritmul de shortstroke misca geamul în jos pentru a nu se agata în cheder.

- Thermal protection

Există mai mulți algoritmi de thermal protection în funcție de componenta vizată ca de exemplu: Fet Thermal Protection (bazat pe temperatura de pe FET-uri), Motor Thermal Protection (bazat pe temperatura motorului), Dynamic Thermal Protection (bazat tot pe temperatura motorului, însă se ține cont și de temperatura ambientală). În funcție de valoarea temperaturii, se poate intra în una din starile de warning sau alert. În primul caz, se va termina comanda care se află în desfasurare, dar nu se mai poate executa alta, iar în cel de al doilea, se taie și requestul curent.

- SoftStop

Aceasta este o functionalitate foarte importantă care are scopul de a menține într-o stare bună de funcționare mecanica geamului. În cazul în care acesta ajunge în poziția de hard block, geamul va face un reverse, pentru detensionarea sufelor.

Moduri de funcționare:

- Oprit -> dacă șoferul nu este prezent în mașină motorul geamului nu va putea fi actuat.
- Auto Up/Down -> în cazul în care se va primi o astfel de comandă, motorul va fi actuat până ce geamul va fi închis/deschis complet
- Manual Up/Down -> în acest caz, geamul va executa comanda pentru o perioadă prestabilită de timp.

Ciorna de solutii:

- Aplicația va fi activată la intrarea în laboratorul de SI. RPI își va trimite credențialele către un RPI master aflat în laborator pentru a fi înregistrat în cluster și a autoriza ce alte noduri să pot realiza acțiunea.
- Motorul geamului va fi actuat prin intermediul unei aplicații web, care va contine patru butoane, corespunzătoare mișcărilor de auto și respectiv manual
- Pentru a stabili poziția geamului, vom plasa în partea superioară a acestuia un senzor de distanță ultrasonic, care ne va preciza distanța rămasă până la inchidere
- În cazul unei comenzi de auto up, dacă geamul ajunge în hard block, vom opri motorul

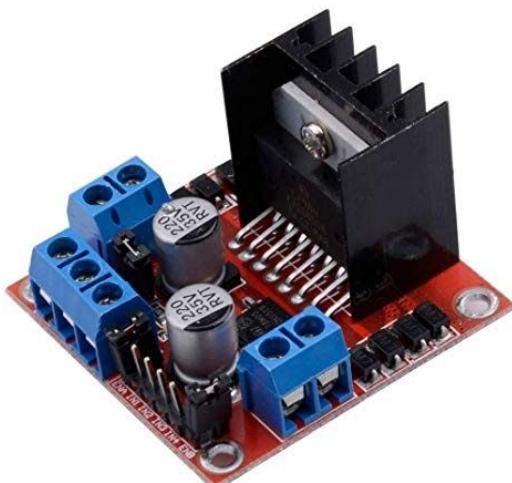
- În cazul unei mișcări de up (auto sau manual), dacă geamul este deja închis, comanda nu va fi executată.

Materiale necesare:

- Motor de curent continuu



- Motor driver



- Fire conexiune
- Raspberry Pi



Nume si prenume:

Gaspar Mona Gabriela

Tema proiectului:

- Vom monitoriza presiunea din roti, iar cand aceasta scade sub o anumita valoare, se va aprinde un martor in bord

Istoric:

Utilizarea senzorilor de presiune roti a inceput sa fie reglementata initial in SUA incepand cu anul 2000. Atunci, Administratia Pentru Siguranta Traficului (NHTSA) a elaborat documentul TREAD prin care se reglementeaza normele si implementarea sistemelor de monitorizare a presiunii aerului din anvelope de catre producatorii masinilor.

In Uniunea Europeana si implicit Romania, obligativitatea echiparii autovehiculelor cu acest tip de senzori a intrat in vigoarea incepand cu data de 01 noiembrie 2014 (aliniatul 5, art. 13 din Regulamentul (CE) nr.661/2009 .

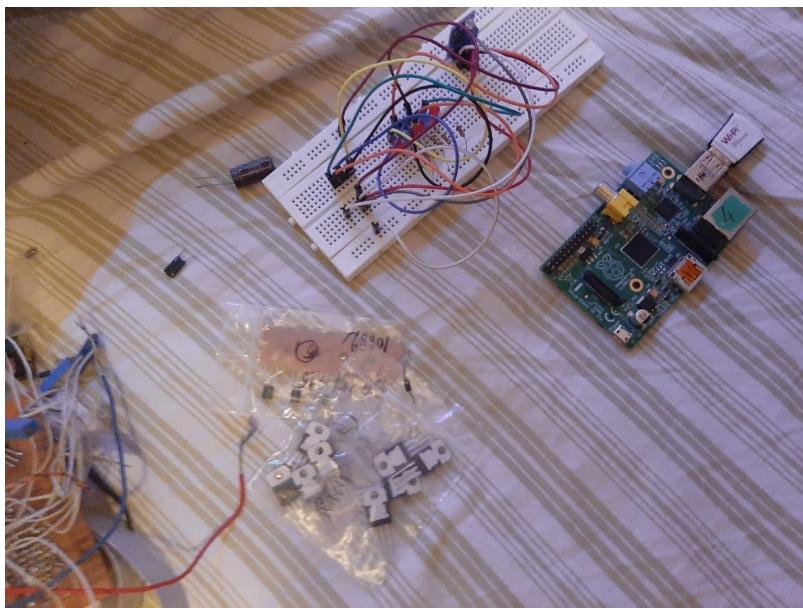
Echiparea rotilor cu senzori de presiune, pe langa faptul ca este obligatorie, are multiple avantaje, printre care cel mai important este reducerea riscului de accidente.

Rezumat:

- Vom folosi o placuta Raspberry PI 3 Model B si 4 x Modul senzor presiune atmosferica BMP280
Vom conecta senzorul de presiune in felul urmator:
VCC -> +3.3V
GND -> GND
SCL -> Pin 5 analogic (A5)
SDA -> Pin 4 analogic (A4)
SDO -> GND (adresa slave 0x76)
CSB -> +3.3V
*Vom folosi libraria Adafruit BMP280

Mod de functionare:

Atat timp cat contactul la masina este pus, sistemul va functiona.
Cand masina este oprită, la fel va fi si sistemul.





Mod de implementare:

```
package si
import (
    "encoding/binary"
)

type Sensor struct {
    Id          int
    Address     ble.Addr
    Prespascal  int
    TempCelsius int
}

func (this *Sensor) ParseData(b []byte) {
    this.Prespascal = int(binary.LittleEndian.Uint32(b[8:]))
    this.TempCelsius = int(binary.LittleEndian.Uint32(b[12:])) / 100
}
```

```

package si

func TestSensor(t *test.T) {
    var sensor *Sensor
    BeforeEach(func() {
        sensor = &Sensor{}
    })

    AfterEach(func() {
    })
    Describe("SI()", func() {
        It("should return a Sensor object", func() {
            AssertEqual(reflect.TypeOf(sensor).String(), "*si.Sensor")
        })
        It("should ", func() {
            file, err := os.Open("./fixtures/cold")
            if err != nil {
                log.Fatal(err)
            }
            data := make([]byte, 1872)
            file.Read(data)
            for i := 0; i < 1872; i = i + 18 {
                sensor.ParseData(data[i:])
                fmt.Printf("Presiune: %v, Temperatura: %v\n", sensor.Prespascal, sensor.TempCelsius)
            }
        })
    })
    Report(t)
}

package main

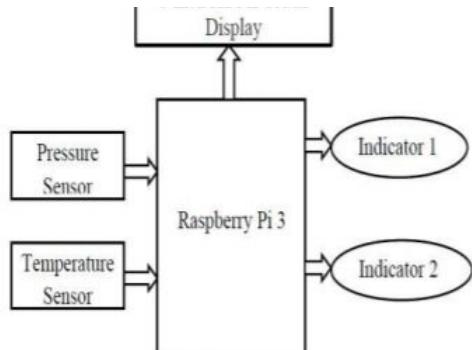
import (
    "flag"
    "si"
    "log"
    "time"
)

var {
    duration = flag.Duration("duration", 0, "monitoring duration Xs, 0 for indefinitely")
}

func main() {
    flag.Parse()
    tires, err := tpms.NewTpms()
    if err != nil {
        log.Fatal(err)
    }
    tires.Log("./log")
    tires.StartMonitoring()
    start := time.Now()
    defer tires.StopMonitoring()
    for *duration == 0 || time.Now().Sub(start) < *duration {
        for _, sensor := range tires.Read() {
            if sensor != nil {
                fmt.Printf("Sensor: %d, Pa: %d, °C: %d\n", sensor.Id, sensor.Prespascal, sensor.Tempcelsius)
            }
        }
        time.Sleep(5 * time.Second)
    }
}

```

Ciorne de solutie:



Proiectul constă în folosirea unei placute Raspberry Pi ce va măsura presiunea și temperatura din roți. În momentul în care sistemul va fi pus pe mașină va fi salvată valoarea de prag sub care presiunea nu trebuie să scada. În momentul în care în una din roți presiunea va scădea sub aceea valoare se va aprinde un martor în bord ce indică roata la care trebuie reglată presiunea.

Presiunea va fi convertită din PSI în kPa (1PSI ≈ 6.89 kPa). Aceasta va fi citită periodic și va fi transmisă mai departe informația pentru a fi prelucrată.

În funcție de datele pe care le primește, programul va anunța șoferul dacă și unde mai exact trebuie să se uite, ce roată nu se încadrează cu valorile în ceea ce am stabilit inițial.

Nume, prenume:

- Popa Angelica Gabriela



Tema de proiect:

- Control ștergere geamuri din față

Rezumat:

Modulul va controla când și cum se vor porni ștergătoarele de geamuri din față. Șoferul va putea seta intervale de timp sau le va activa la nevoie, în două moduri: doar cu apă sau și cu soluție de curățat



Explorare documentară asupra temei (raport sintetic alternative soluție):

Soluția constă în realizarea unui sistem de control a ștergătoarelor de geamuri, folosind un Raspberry PI 3 Model B, cu ajutorul unor servomotoare sau motoare pas cu pas.

Servomotoarele sunt motoare electrice speciale, de curent continuu sau curent alternativ cu viteză de rotație reglabilă într-o gamă largă în ambele sensuri având ca scop deplasarea într-un timp prescris a unui sistem mecanic de-a lungul unei traectorii date, realizând totodată și poziționarea acestuia la sfârșitul cursei cu o anumită precizie.

Sistemele de reglare automată moderne impun servomotoarelor următoarele performanțe:

- gamă largă de modificare a vitezei în ambele sensuri
- funcționare stabilă la viteză foarte mică
- constante de timp cât mai reduse
- fiabilitate și robustețe ridicate
- raport cuplu/moment de inerție cât mai mare
- suprasarcină dinamică admisibilă mare
- caracteristici de reglare liniare

Servomotoarele electrice se folosesc în cele mai diverse aplicații cum ar fi acționarea roboților industriali universali, a mașinilor unelte cu comandă numerică, a perifericelor de calculator, în acționarea imprimantelor rapide, în tehnica aerospatială, instalații medicale etc



O definiție simplă a motorului pas cu pas este: „*un dispozitiv electromecanic care convertește impulsurile electrice în mișcări mecanice discrete*”. [3,17,22]

Axul motorului pas cu pas execută o mișcare de rotație în pași incrementali discreți când este aplicată în secvență corectă o comandă electrică în pulsuri. Rotația motorului este strâns legată de caracteristicile acestor impulsuri electrice. Astfel direcția de rotație a motorului este direct legată de secvența în care sunt aplicate pulsurile electrice, de asemenea și viteza de rotație este direct dependentă de frecvența impulsurilor electrice iar deplasarea unghiulară este direct dependentă de numărul de pulsuri electrice aplicate.[3,17]

În comparație cu alte tipuri de motoare (motoarele de curent continuu sau motoarele de curent alternativ asincrone și sincrone) motorul pas cu pas are o serie de avantaje:

- Rotația unghiulară a motorului este proporțională cu pulsul electric aplicat;
- Motorul are moment maxim în poziția oprit dacă bobinele sunt alimentate;
- Poziționare precisă, cu o eroare de 3-5% la un pas, care nu se cumulează de la un pas la altul;
- Răspunsuri excelente la pornit/oprit/schimbarea direcției de rotație;
- Fiabilitate excelentă deoarece nu există perii de contact la motor, deci durata de funcționare depinde de rulment;
- Posibilitatea de a obține viteze foarte mici cu sarcina legată direct pe axul motorului;
- O gamă foarte largă de viteze de rotație;

dar există și unele dezavantaje:

- Rezonanța poate apărea în cazul unui control deficitar;
- Controlul greoi la viteze foarte mari.[1,16,24,25]

Moduri de funcționare:

- Oprit -> ștergătoarele nu se vor activa în absența șoferului.
- Auto On Water/Mixed -> la intervale regulate, ștergătoarele se vor activa
- Manual On Water/Mixed -> la o apăsare de buton pentru fiecare opțiune, ștergătoarele se vor activa

Ciornă de soluție:

Soluția constă în realizarea unui sistem de control a ștergătoarelor de geamuri, folosind un Raspberry PI 3 Model B, cu ajutorul unor servomotoare sau motoare pas cu pas. Cele două motoare vor fi amplasate de o parte și de alta a parbrizului și se vor mișca/vor fi activate în concordanță cu modul de funcționare. Acestea vor fi blocate atunci când motorul este oprit. În caz contrar, vor fi activate la apăsare

de buton, conform modului dorit. De asemenea, pentru modul Mixed, vor exista motoare care eliberează substanțe de curățare.

Mod de implementare:

GPIO Numbers

Raspberry Pi B
Rev 1 P1 GPIO Header

Pin No.		
3.3V	1	2
GPIO0	3	4
GPIO1	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO21	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26
5V		
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26

Raspberry Pi A/B
Rev 2 P1 GPIO Header

Pin No.		
3.3V	1	2
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26
5V		
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26

Raspberry Pi B+
J8 GPIO Header

Pin No.		
3.3V	1	2
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26
DNC	27	28
GPIO5	29	30
GPIO6	31	32
GPIO13	33	34
GPIO19	35	36
GPIO26	37	38
GND	39	40
5V		
GPIO2	3	4
GPIO3	5	6
GPIO4	7	8
GND	9	10
GPIO17	11	12
GPIO27	13	14
GPIO22	15	16
3.3V	17	18
GPIO10	19	20
GPIO9	21	22
GPIO11	23	24
GND	25	26
DNC	27	28
GPIO5	29	30
GPIO6	31	32
GPIO13	33	34
GPIO19	35	36
GPIO26	37	38
GND	39	40

Cod-exemplu:

```

import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(03, GPIO.OUT)
pwm=GPIO.PWM(03, 50)
pwm.start(0)
SetAngle(90)
pwm.stop()
GPIO.cleanup()

def SetAngle(angle):
    duty = angle / 18 + 2
    GPIO.output(03, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
    GPIO.output(03, False)
    pwm.ChangeDutyCycle(0)

```

Materiale:

- Raspberry PI Model 3B
- Fire
- Servomotor/Motor pas cu pas
- Breadboard



**Nume, prenume:**

- Bostan Maria Georgiana

Tema de proiect:

- Sistem de control a poziției scaunului

Rezumat:

Modulul va controla cum va fi poziționat scaunul șoferului. Sistemul a fost gândit, în principiu, pentru poziționarea scaunului pentru buna statură a șoferului. Acesta va putea să memoreze poziția cea mai frecventă, însă în cazul apariției unor schimbări sau a unor șoferi noi, sistemul va permite și schimabarea manuală a poziției.

Explorare documentară asupra temei (raport sintetic alternative soluție):

Soluția constă în realizarea unui sistem de poziționare a scaunului, folosind un Raspberry PI 3 Model B, cu ajutorul unor servomotoare.

Servomotorul este un element component al unui sistem care funcțional implică poziții relative reglabile între anumite elemente componente ale sale.

- servomotorul este elementul component care acționează direct sau indirect asupra elementelor componente cu poziții relative reglabile

- servomotorul poate avea poziție fixă, blocat pe sistem, în imediata lui apropiere sau poate fi conținut în subsistemul unui element cu poziție reglabilă

- puterea motorului servomotorului determină viteza de modificare a poziției relative și frecvența de modificare a poziției relative

- puterea motorului servomotorului este invers proporțională cu nivelul de precizie al servomotorului

- soluția tehnică care definește servomotorul, implică soluții constructive simple, care funcțional, impun un consum redus de energie, o cinematică definită de mișcări lineare, circulare sau combinări ale acestora

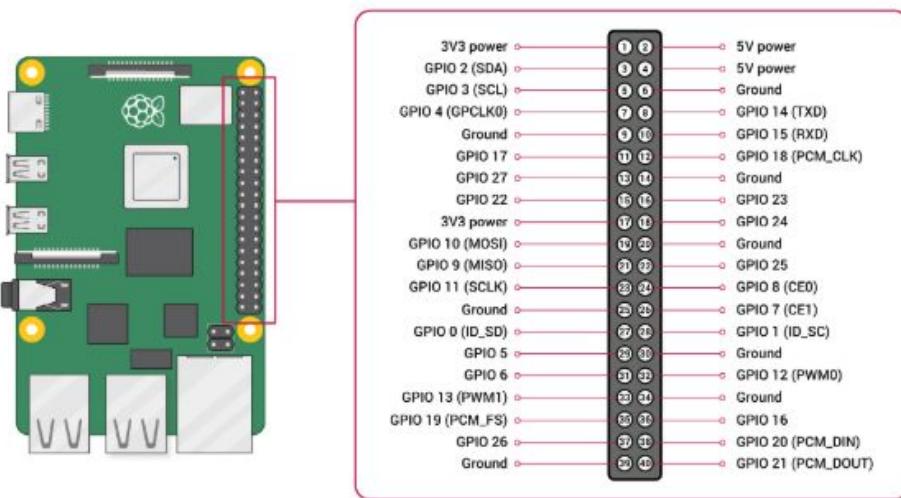
- soluția tehnică care definește servomotorul are o arie largă de aplicabilitate, fiind concepută pentru o multitudine de sisteme, prin aceasta inducând soluțiile tehnice și constructive pentru sistemele în care este agreat funcțional, rezultatul global fiind soluții constructive compacte, modulate, interschimbabile, standardizate pentru servomotoare.

Servomotorul are trei pini de conectare, Vcc, masă și semnal. Necesită o putere de 5V 1A, care este furnizată de la o sursă de alimentare adecvată.

Pe placa Raspberry Pi, pinul GPIO 22 este selectat ca pinul de semnal. Pinul GND al sursei de alimentare este conectat la pinul de masă al servo și, de asemenea, la al șaselea pin de pe Raspberry Pi.

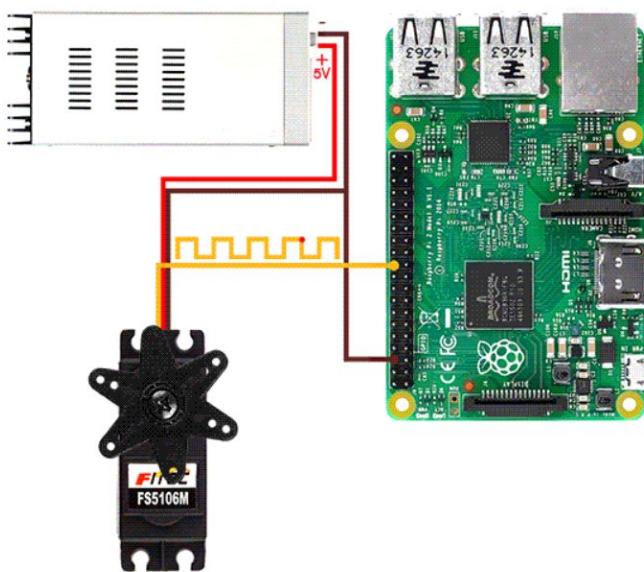
Moduri de funcționare:

- Automat: sistemul va putea memora poziția de bază a șoferului care utilizează cel mai mult mașina
- Manual: în cazul unui nou șofer, acesta își va customiza poziția scaunului



Ciornă de soluție:

Soluția constă în realizarea unui sistem de poziționare a scaunului, folosind un Raspberry PI 3 Model B, cu ajutorul unor servomotoare. Servomotoarele vor fi poziționate de o parte și de alta a scaunului, pentru controlul spătarului, și unghiul la care se vor mișca acestea va fi stabilit de șofer.



Materiale necesare:

-Raspberry PI Model 3B

- Servomotor SG90
- Breadboard
- Fire



Exemplu de implementare

```

import RPi
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(22, GPIO.OUT)

pwm=GPIO.PWM(22,100)
pwm.start(5)

angle1=10
duty1= float(angle1)/10 + 2.5

angle2=45
duty2= float(angle2)/10 + 2.5

ck=0
while ck<=5:
    pwm.ChangeDutyCycle(duty1)
    time.sleep(0.8)
    pwm.ChangeDutyCycle(duty2)
    time.sleep(0.8)
    ck=ck+1
time.sleep(1)
GPIO.cleanup()

```

Nume și prenume:

- Livadaru Cristi
- Pipirig Mihai

**Tema:**

Retrocomputing - IBM 6360 Diskette Unit

Prezentare generală, istoric:

Experientul își propune crearea unui proiect de tip **retrocomputing** folosind unitatea de dischete (**IBM 6360 Diskette Unit**) a microcomputerului dedicat pentru procesarea textului (**IBM Displaywriter System 6580**) scos pe piață de către IBM în anul 1980. Ne propunem să comandăm prin voce pornirea sistemului.

Retrocomputingul reprezintă utilizarea hardware-ului și software-ului computerizat mai vechi în timpurile moderne. Retrocomputingul este clasificat de obicei ca hobby și recreere, mai degrabă decât o aplicare practică a tehnologiei; pasionații adună adesea hardware și software rar și valoros din motive sentimentale.

IBM Displaywriter System 6580

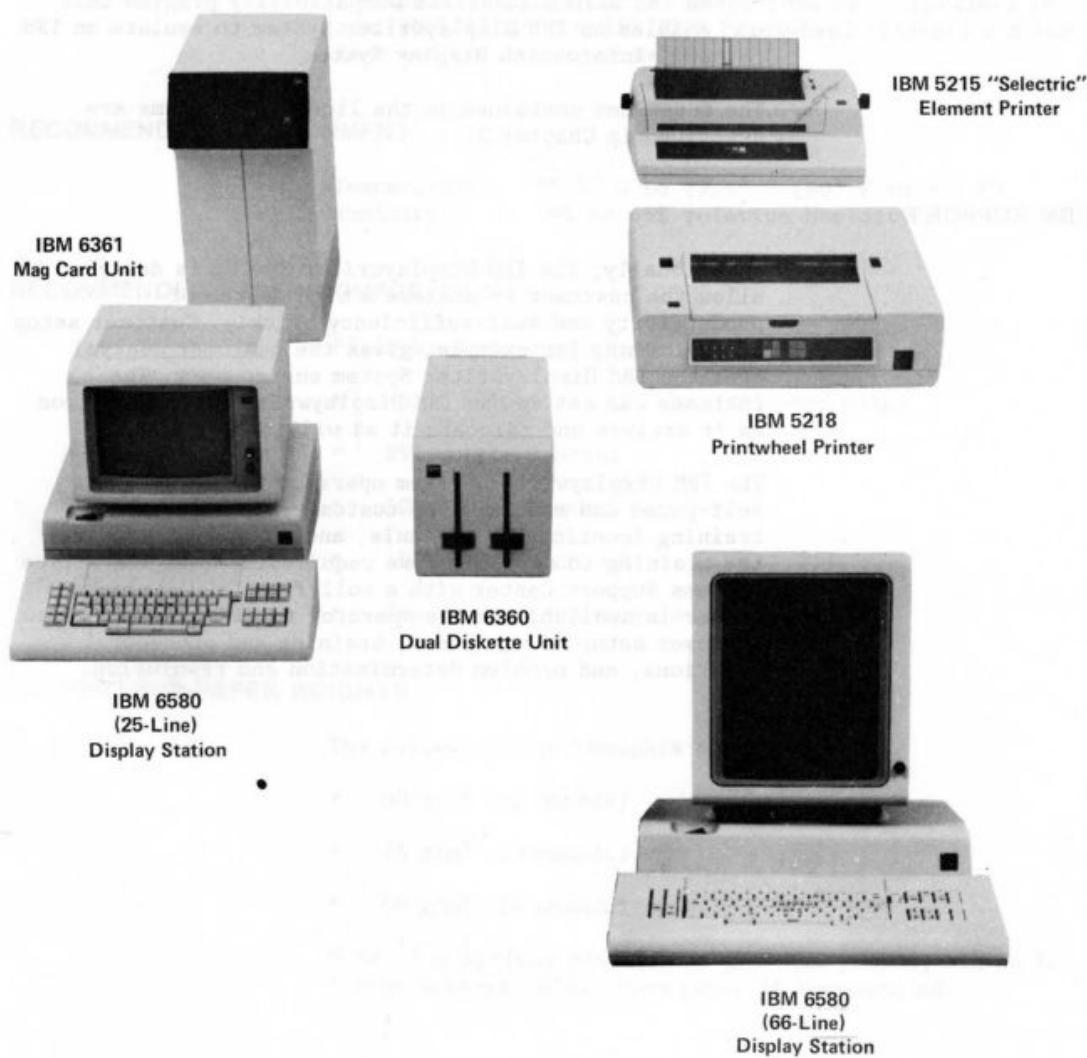
"IBM's Office Products Division announced the Displaywriter in June 1980 as an easy-to-use, low-cost desktop text processing system. The Displaywriter System enabled operators to produce high quality documents while keying at rough draft speed. Users could automatically indent text, justify right margins, center and underscore. They could also store a document and recall it for review or revision, and could check the spelling of approximately 50,000 commonly used words. While these features are taken for granted in the post-PC era, they were novel for a time when most documents were created, formatted and revised on manual or electric typewriters."

The Displaywriter's "intelligence" came in 160K, 192K or 224K bytes of memory. Single diskette drive diskette units with a capacity for approximately 284,000 characters of information were available. As requirements increased, customers could upgrade to a dual drive diskette unit. Optional communications features enabled the Displaywriter to distribute information quickly over ordinary telephone lines.

A basic system — consisting of a display with a typewriter-like keyboard and a logic unit, a printer and a device to record and read diskettes capable of storing more than 100 pages of average text — cost \$7,895 and leased for \$275 a month. A system of three

displays sharing a single higher speed printer and a paper handler sold for \$26,185 and leased for \$845 a month. The Displaywriter was not your father's Selectric." (IBM Archives - ibm.com/ibm/history/exhibits/pc/pc_8.html)





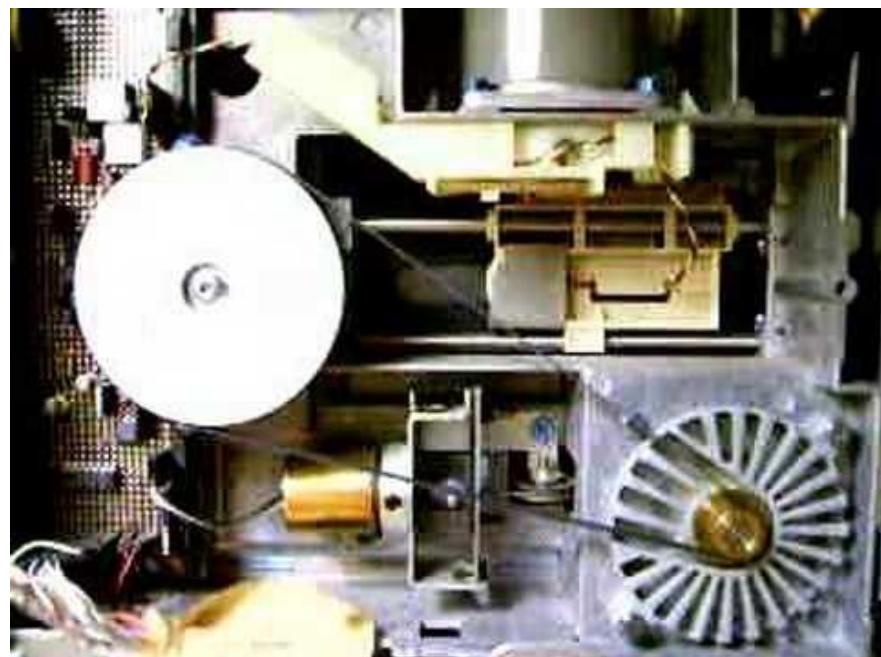
IBM 6360 Diskette Unit

*"The **IBM 6360 Diskette Unit** can read information stored on the magnetic diskettes and record information created at the keyboard onto the diskette. The operator can move the cable-connected diskette unit to either the left or right of the display station, whichever is more convenient.*

A **diskette** is a magnetic disk approximately 203 mm (8 inches) in diameter, enclosed in a protective sleeve. The storage capacity of the diskette depends upon the type of diskette used. An IBM Diskette 1 (one-sided diskette) has a storage capacity of approximately 284,000 bytes of customer usable characters and controls. An IBM Diskette 2D, a double-density, two-sided diskette, can have information stored on both sides of the diskette with more information stored in the same amount of space. An IBM Diskette 2D has a storage capacity of approximately 985,000 bytes of customer usable characters and controls. The licensed programs, which control text and communications functions, are stored on program diskettes. The operator uses work diskettes to store operator-created information.

The IBM 6360 Single Diskette Unit (Figure 2-4) has one usable slot (the right-hand slot is blocked) and operates with one diskette inserted at a time. Any tasks which require the use of more than one diskette, such as copying a diskette, are done by the operator inserting each diskette as it is needed.

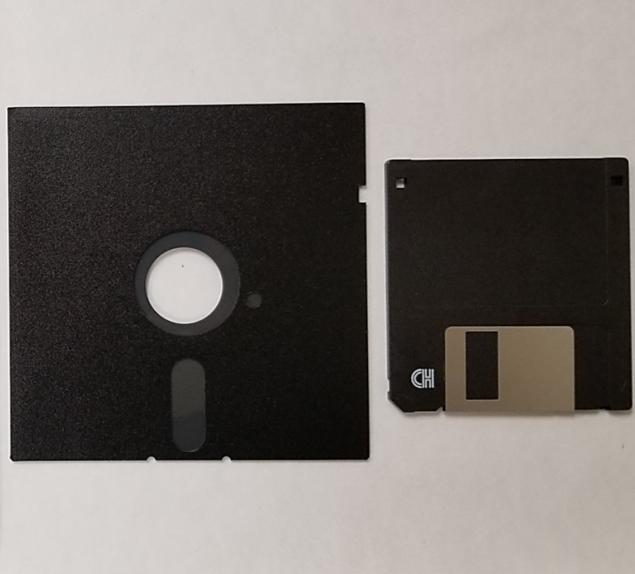
The IBM 6360 Dual Diskette Unit is identical to the single diskette unit in appearance and function, except that the dual diskette unit has two usable diskette slots. Printing can be done from one diskette while the operator works with the other diskette. Tasks such as copying documents from one diskette to another can be done without operator intervention" (General Information Manual for the IBM Displaywriter System Fifth Edition (May 1982))



8-inch

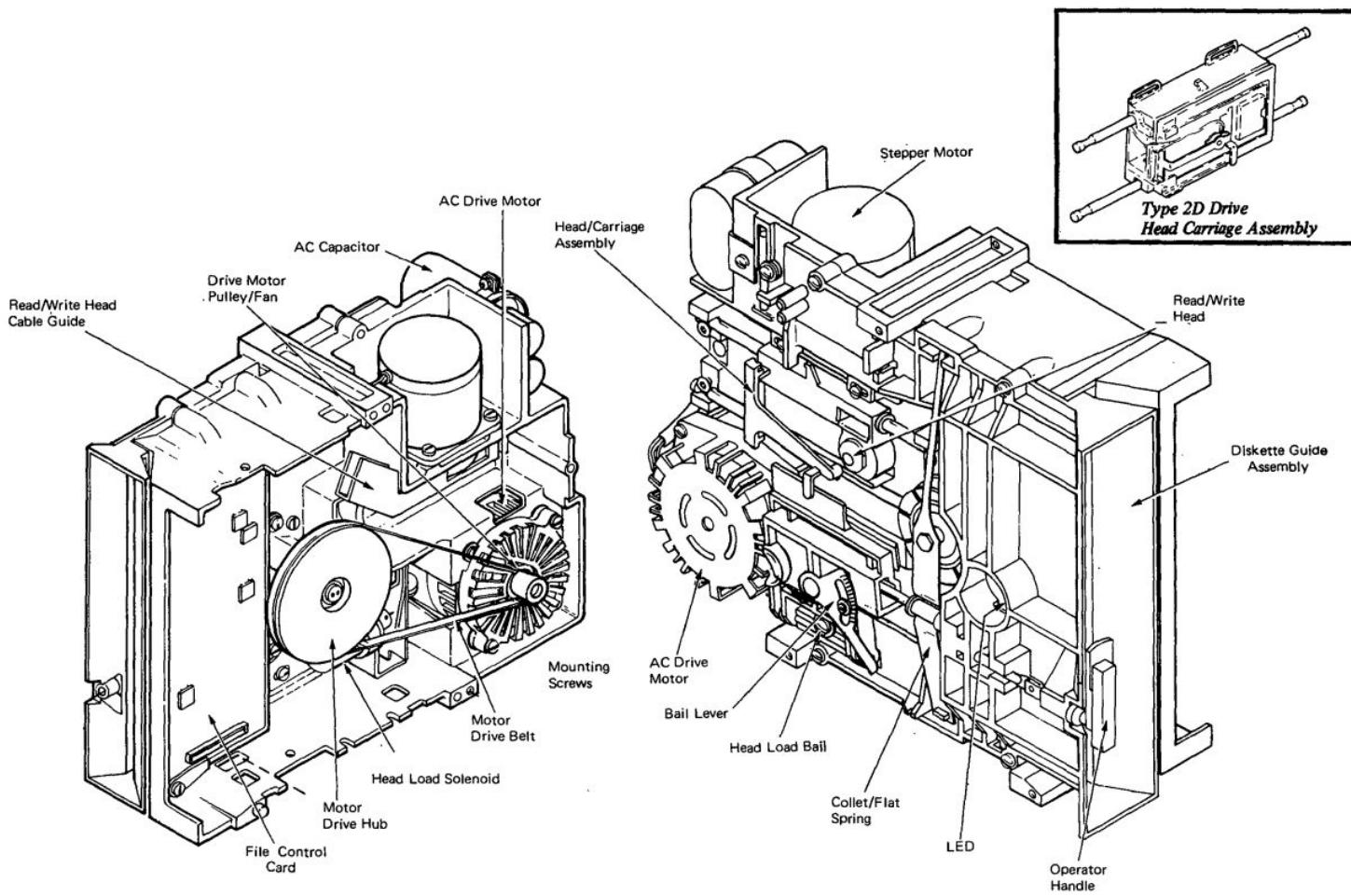


5.25-inch



3.50-inch

Diskette Unit Locations - Internal View



Data is written on the diskette by magnetically changing an area of the surface into a pattern of eight bits known as a "byte". Each byte represents one data character, either alpha/numeric or a code.

An operator can create text in system memory and record it on the diskette. The operator can move text from the diskette to the memory area, and return revised material to the diskette. After checking for a correct diskette write operation, the system clears the memory and makes it available for other jobs.

Diskette partitioning

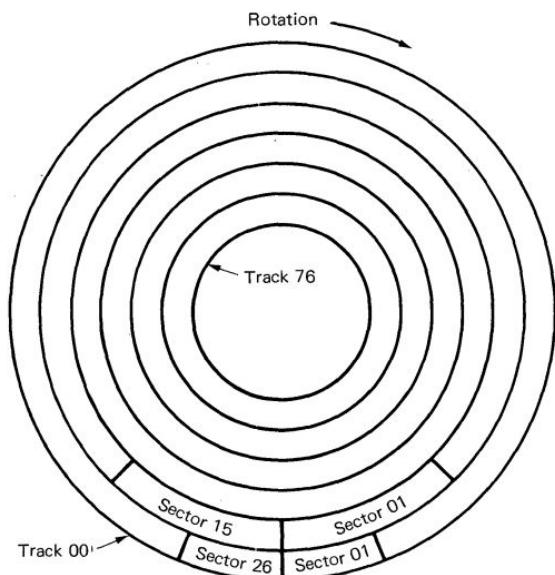


Figure 7-3. Diskette Tracks and Sectors

The IBM Diskette is divided into 77 circular tracks numbered 00 through 76 (Figure 7-3). As the diskette turns, data is magnetically recorded on these tracks by the read/write head. Track 00 is the index or address track and is divided into 26 sectors having 128 bytes of information each. This track contains the information necessary to identify the diskette and its contents. The index track cannot be used for data storage.

Tracks 01 through 76 are data tracks and are divided into 15 sectors for the Type 2D diskette." Track numbers 75 and 76 are reserved to take the place of any primary tracks (01-74) that have failed.

Each sector of each data track starts with its own address and contains 256 bytes of data. Each time the diskette does a read or write operation, a complete sector on the selected track is either read into or written from the main storage area.

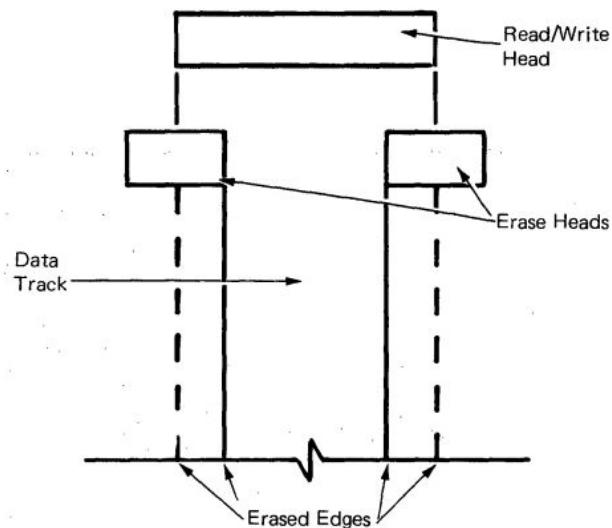
The IBM Diskette is permanently contained in an envelope for protection. The inner surface of the envelope is coated with a material which cleans the diskette as it turns inside the envelope.

Rotation feedback mechanism

A Light Emitting Diode (LED) and a Phototransistor (PTX) are used to sense diskette rotation. The diskette has a small hole in its inner edge which passes the LED and is sensed by the PTX with each revolution. As the diskette continues to turn, the light to the PTX is cut off. This causes a light pulse six times per second or every 166.7 milliseconds. These pulses indicate to the file control card that the diskette is turning and are reference points for the diskette adapter to check the speed of the diskette (360 Revolutions per Minute (RPM)).

Operation of the diskette drive will continue normally if the pulse period does not change from 166.7 ms by more than 4.2 ms (162.5 ms to 170.9 ms).

Read/Write Head



The read/write head is divided into three sections: A read/write section and two erase sections. The erase sections are located on either side of the data track and erase the edges of the data track during a write operation.

Figure 7-13. Read/Write Data Track

File Control Card

The File Control Card, attached to the right side of each diskette drive frame, contains circuits for the stepper motor, the head load solenoid, and the read/write functions. The card also contains the amplifier circuits for the read/write head and the LED/phototransistor. This card is controlled by signals from the diskette adapter card, located behind the diskette drive unit.

Stepping Signals

Two signals, access 0 and access 1, activate the stepper motor. These two signals are sent through an Access Degate logic block through an AND (A) block where they are combined, then through an amplifier to increase the signal before going to the motor control line. At this point, motor control line becomes an electrical detent. The stepper motor armature will turn 1.8 to align with the detent made in the winding. The stepper motor will remain in this detented position until the system electronics signal the access lines again.

The next access signal through the access degate logic selects stepper motor line 1. The stepper motor armature turns another 1.8 to the next track. The access lines remain active holding the motor and head carriage in this track until the next signal is received on the access lines to step the head either forward or back.

Fifteen separate step signals are needed to move the head carriage assembly from track 50 to track 65. The signals step the head carriage through a four condition sequence and repeat the sequence until fifteen steps have been completed. Seven step signals are needed to move the read/write head from track 61 back to track 54. If the system electronics cannot determine the track position, it recovers by sending 77 reverse step signals. This ensures the read/write head returns to track 00. A pin located in the

stepper pulley stops against the casting to physically stop the read/write head at track 00. When the system electronics completes the 77 step signals, it will start a new count sequence by reading an address field on track 00.

Track Location	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
Access 0	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+
Access 1	+	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-

- + Indicates an active state.
- Indicates an inactive state.

Figure 7-19. Stepping Signal Sequence

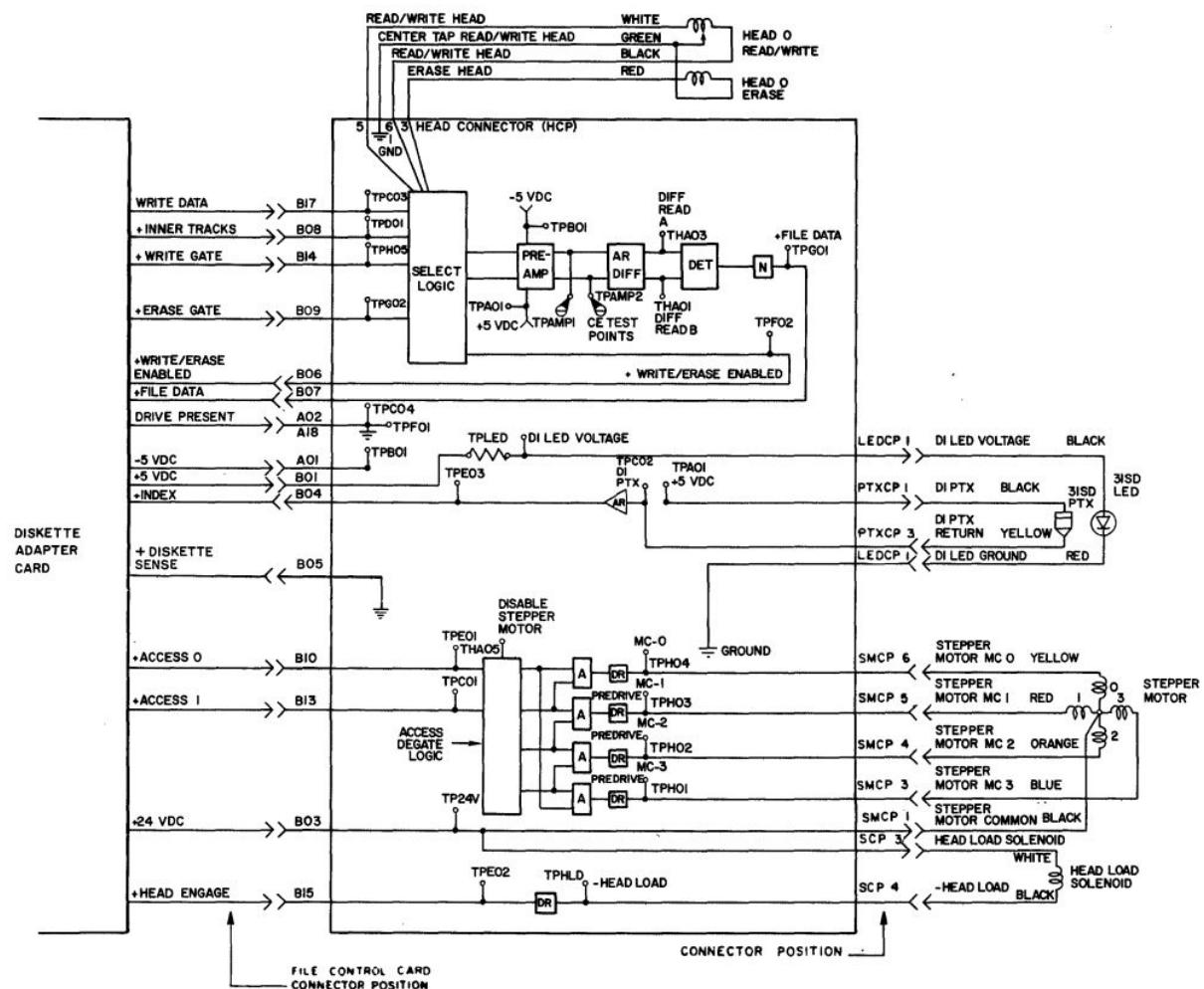


Figure 7-15. File Control Card - Type 1 Drive

Resurse:

- IBM 6360 Diskette Unit
- Platforma Raspberry Pi
- Releu 2 canale (Pololu Basic 2-Channel SPDT Relay Carrier with 5VDC Relays)
- iPhone

Surse programe prototip:

Pornire:

```
import RPi.GPIO as GPIO
import time

#This is name of the module - it can be anything you want
moduleName = "discoon"

#These are the words you must say for this module to be executed
commandWords = ["disco", "on"]

#This is the main function which will be execute when the above command words are said
def execute(command):
    # print("\n")
    print("-----DISCO ON-----")
    # print("\n")

    GPIO.setmode(GPIO.BOARD) # mod numerotare conector
    GPIO.setup(3,GPIO.OUT)  # iesire
    GPIO.output(3,GPIO.HIGH) # Sistem pornit
```

Oprire:

```
import RPi.GPIO as GPIO
import time

#This is name of the module - it can be anything you want
moduleName = "discooff"

#These are the words you must say for this module to be executed
```

```

commandWords = ["disco","off"]

#This is the main function which will be execute when the above command words are said

def execute(command):

    #  print("\n")

    print("-----DISCO OFF-----")

    #  print("\n")

    GPIO.setmode(GPIO.BOARD) # mod numerotare conector

    GPIO.setup(3,GPIO.OUT)  # iesire

    GPIO.output(3,GPIO.LOW) # Sistem oprit

```

Referinte documentare:

- [General Information Manual for the IBM Displaywriter System Fifth Edition \(May 1982\)](#)
- [IBM Archives](#)
- [Displaywriter 6360/6580 Product Support Manual \(February, 1983\)](#)
- [IBM Displaywriter System](#)
- [Retrocomputing](#)
- [Pololu Basic 2-Channel SPDT Relay Carrier with 5VDC Relays](#)

Toma Răzvan(1405A), Scutar Ioan-Alexandru(1406A)



Tema proiect:

Calculator pentru distante folosind un senzor ultrasonic

Rezumat:

Masurarea distantei dintre sursa semnalului și tinta utilizând un senzor cu ultrasunet.

Explorare documentara asupra temei:

Soluția constă în realizarea unui sistem pentru masurarea distantei cu ajutorul senzorului utilizat. Acest proiect se poate utiliza împreună cu un vehicul de test având rol de asistentă în trafic.

Sonarul ultrasonic: HC-SR04 este un modul utilizat frecvent pentru măsurarea distanței fără contact pentru distanțe de la 2cm la 400cm. Folosește sonar (precum liliacii și delfinii) pentru a măsura distanța cu precizie ridicată și lecturi stabile. Este format dintr-un emițător, receptor și circuit de control cu ultrasunete. Transmițătorul transmite rafale scurte care sunt reflectate de țintă și sunt preluate de receptor. Diferența de timp dintre transmiterea și recepția semnalelor ultrasonice este calculată. Folosind viteza sunetului și ecuația „Viteză = distanță / timp”, distanța dintre sursă și țintă poate fi calculată cu ușurință.

Calcul distanță: Timpul luat de puls este de fapt pentru și de la deplasarea semnalelor ultrasonice, în timp ce avem nevoie de doar jumătate din aceasta. Prin urmare, timpul este luat ca $\text{timp} / 2$.

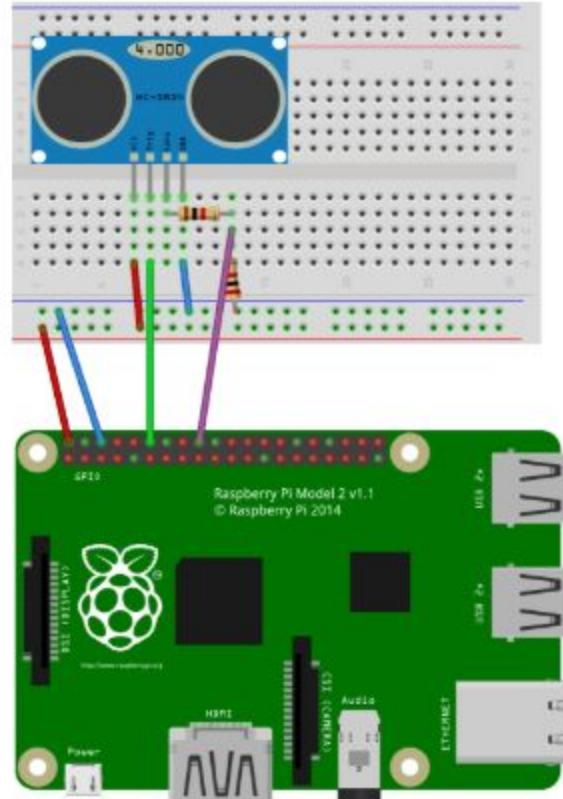
$$\text{Distanța} = \text{Viteza} * \text{Timp} / 2$$

Moduri de funcționare:

- Oprit -> nu afisează nimic;
- Pornit -> afisează distanța măsurată;

Materiale necesare:

- Raspberry Pi 3 Model B
- rezistor 1Ω
- rezistor $2 k\Omega$
- breadboard
- senzor ultrasonic HC-SR04
- cabluri
- microSD card cu distribuția Raspbian GNU/Linux



Cod py:

```
1 import RPi.GPIO as GPIO
2 import time
3 import signal
4 import sys
5
6 # use Raspberry Pi board pin numbers
7 GPIO.setmode(GPIO.BCM)
8 # set GPIO Pins
9 pinTrigger = 18
10 pinEcho = 24
11 def close(signal, frame):
12     print("\nTurning off ultrasonic distance detection...\n")
13     GPIO.cleanup()
14     sys.exit(0)
15 signal.signal(signal.SIGINT, close)
16 # set GPIO input and output channels
17 GPIO.setup(pinTrigger, GPIO.OUT)
18 GPIO.setup(pinEcho, GPIO.IN)
19 while True:
20     # set Trigger to HIGH
21     GPIO.output(pinTrigger, True)
22     # set Trigger after 0.01ms to LOW
23     time.sleep(0.00001)
24     GPIO.output(pinTrigger, False)
25     startTime = time.time()
26     stopTime = time.time()
27     # save start time
28     while 0 == GPIO.input(pinEcho):
29         startTime = time.time()
30     # save time of arrival
31     while 1 == GPIO.input(pinEcho):
32         stopTime = time.time()
33     # time difference between start and arrival
34     TimeElapsed = stopTime - startTime
35     # multiply with the sonic speed (34300 cm/s)
36     # and divide by 2, because there and back
37     distance = (TimeElapsed * 34300) / 2
38
39     print ("Distance: %.1f cm" % distance)
40     time.sleep(1)
```

Nume si prenume:

Focsaneanu Anda-Georgiana
Foia Adina-Teodora
Grupa 1405A

TEMA PROIECTULUI

Statie meteo mobila

INTRODUCERE. REZUMAT

În lucrare se propune măsurarea la o stație meteo a temperaturii și umidității, utilizând 3 mijloace: utilizând Arduino Uno, un modul LCD și un senzor digital inteligent (tip DHT11), utilizând Arduino Uno, un modul LCD, cu doi senzori analogici, unul pentru măsurarea temperaturii (LM35DH) și umidității (HIH-3602- A), utilizând Arduino Mega, cu LCD și un senzor digital inteligent (tip DHT11). Cu ajutorul acestor senzori se poate face conversia liniara temperatură-tensiune. Arduino UNO este o platformă de procesare open-source, bazată pe software și hardware flexibil și simplu de folosit. Conține într-o platformă de mici dimensiuni (6,8 cm / 5,3 cm) construită în jurul unui microcontroler și este capabilă de a prelua date din mediul înconjurător printr-o serie de senzori și de a efectua acțiuni asupra mediului prin intermediul luminilor, motoarelor, servomotoare, și alte tipuri de dispozitive mecanice. DHT11 este un senzor digital de temperatură și umiditate, care are încorporat un senzor de umiditate capacativ și un termistor, pentru a măsura aerul din jur și să dă un semnal digital pe pinul de date. Aceasta măsoară umiditatea relativă analizând vaporii de apă prin măsurarea rezistenței electrice dintre doi electrozi. Pin-ul VCC al DHT11 a fost conectat la pinul de pe placă Arduino de 5V, pinul GND al DHT11 a fost conectat la pinul GND de pe placă Arduino, pinul DATA al DHT11 a fost conectat la pinul digital 4 al plăcii Arduino.

EXPLORARE DOCUMENTARE ASUPRA TEMEI

Această lucrare urmărește facilitarea măsurării temperaturii și a umidității, în cadrul unei stații meteorologice, cu ajutorul unor senzori foarte sensibili și, totodată, robusti, dar și prin folosirea unor plăci de dezvoltare Arduino UNO și MEGA. Microcontrolerele constituie în această perioadă un domeniu deosebit de dinamic. Acestea se impun din ce în ce mai mult în aplicațiile industriale. Acestea pot fi găsite în componența oricărui tip de echipamente electrice. Orice aparat care măsoară, stochează, comandă, calculează sau afișează informații este o potențială gazdă pentru un microcontroler. Datorită dimensiunilor reduse a componentelor, utilizarea acestora în practică este facilitată, spațiul de acționare a acestora nereprezentând o problemă, și totodată, împreună cu dimensiunile reduse intervine reducerea costurilor de fabricare a acestora, fiind o industrie în continuă dezvoltare, și cu o accesibilitate a prețurilor acceptabilă, și o varietate mare de produse.

PREZENTARE CIORNA SOLUTIE. ELEMENTE COMPOUNTE. IMPLEMENTARE SOLUTIE

Măsurarea realizată în continuare, se bazează pe senzorii analizați, citirile fiind eventual efectuate de către placă Arduino, și transmise către un LCD pentru afișarea măsurătorilor. Placa Arduino UNO Arduino UNO (fig.1) este o platformă de procesare open-source, bazată pe software și hardware flexibil și simplu

de folosit. Conține dintr-o platformă de mici dimensiuni (6,8 cm / 5,3 cm) construită în jurul unui microcontroler și este capabilă de a prelua date din mediul înconjurător printr-o serie de senzori și de a efectua acțiuni asupra mediului prin intermediul luminilor, motoarelor, servomotoare, și alte tipuri de dispozitive mecanice.

Programele Arduino pot fi scrise în orice limbaj de programare cu un compilator capabil să producă un cod mașină binar. Atmel oferă un mediu de dezvoltare pentru microcontrolerele sale, AVR Studio și mai nou, Atmel Studio. Proiectul Arduino oferă un mediu integrat de dezvoltare (IDE), care este o aplicație cross-platform, scrisă în Java. Aceasta își are originile în mediul de dezvoltare pentru limbajul de programare Processing și în proiectul Wiring. Este proiectat pentru a introduce programarea în lumea artiștilor și a celor nefamiliarizați cu dezvoltarea software. Include un editor de cod cu funcții ca evidențierea sintaxelor, potrivirea acoladelor și spațierea automată și oferă mecanisme simple cu un singur click, pentru a compila și a încărca programele în placă Arduino. Un program scris în IDE pentru Arduino se numește sketch.

Senzor de umiditate și temperatură

DHT11 este un senzor digital de temperatură și umiditate, care are încorporat un senzor de umiditate capacativ și un termistor, pentru a măsura aerul din jur și dă un semnal digital pe pinul de date.

Acesta măsoară umiditatea relativă analizând vaporii de apă prin măsurarea rezistenței electrice dintre doi electrozi.

- Pin-ul VCC al DHT11 a fost conectat la pin-ul de pe placă Arduino de 5V;
- Pin-ul GND al DHT11 a fost conectat la pin-ul GND de pe placă Arduino;
- Pin-ul DATA al DHT11 a fost conectat la pin-ul digital 4 al placii Arduino.

Fiecare senzor DHT11 este strict calibrat în laborator și este extrem de precis în calibrarea umidității. Coeficienții de calibrare sunt stocați ca programe în memoria OTP, care sunt utilizate de procesul de detectare a semnalului intern al senzorului. Interfața serială cu un singur fir face integrarea rapidă și ușoară a sistemului. Dimensiuni mici, consum redus de energie și până la 20 de metri transmisia semnalului, ceea ce îl face cea mai bună alegere pentru diverse aplicații, inclusiv pentru cele mai exigente. Componenta este un pachet cu 4 pini pentru un singur rând. Este foarte convenabil ca și conectare și pachetele speciale pot fi furnizate în funcție de solicitarea utilizatorilor.



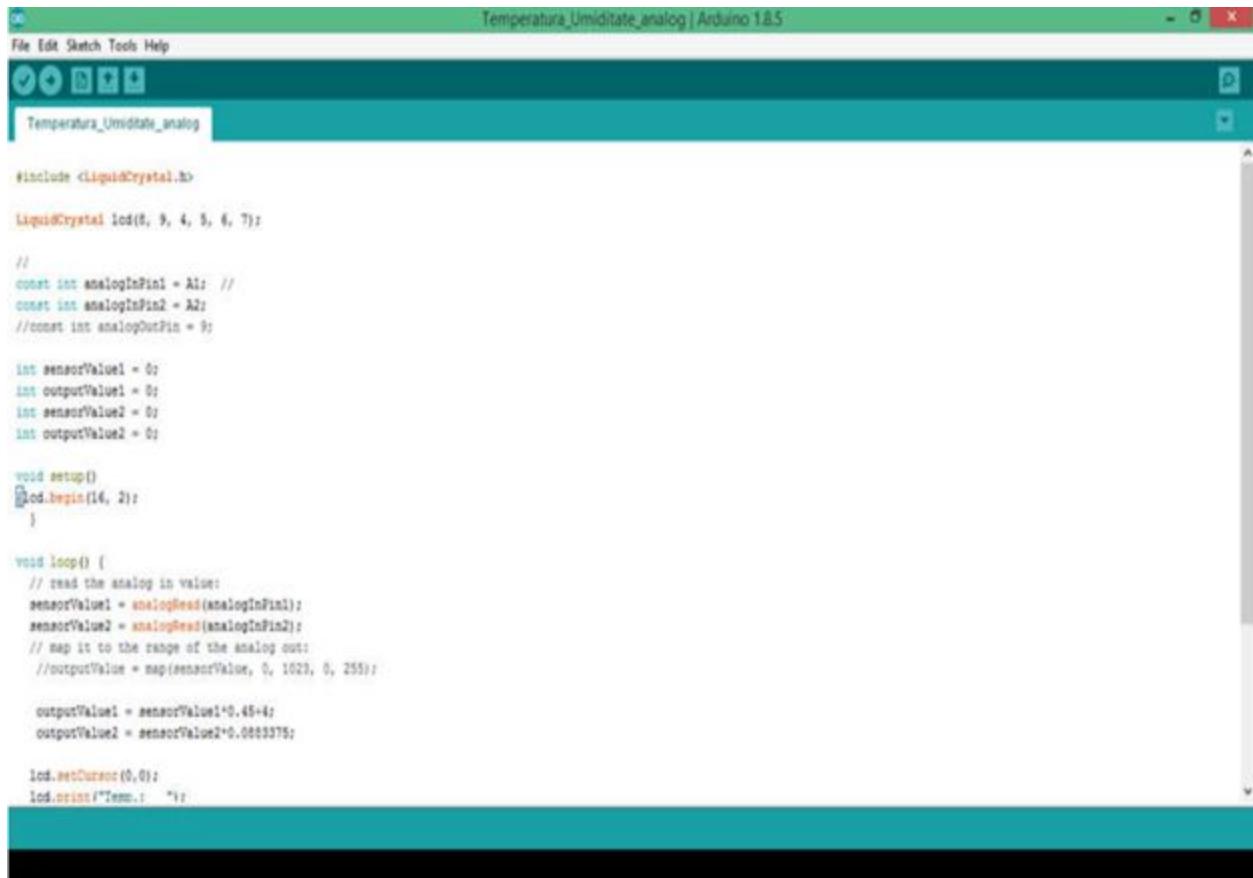
Liquid Crystal Display

Ecranul LCD utilizat este dedicat afișării textului. Poate afișa un text format din 32 de caractere, câte 16 pe fiecare dintre cele două rânduri disponibile. Fiecare caracter este de fapt o matrice de 5x8 puncte. Ecranul se poate monta direct pe placă Arduino Uno sau Mega.

LCD-urile nu emite lumină directă, ci folosesc o lumină de fundal sau un reflector pentru a produce imagini color sau monocrom. Ecranele LCD sunt disponibile pentru a afișa imagini arbitrară (ca pe un afișaj general al computerului) sau imagini fixe cu conținut redus de informații, care pot fi afișate sau ascunse, cum ar fi cuvintele presetate, cifre și afișaje pe șapte segmente, ca într-un ceas digital. Aceștia utilizează aceeași tehnologie de bază, cu excepția faptului că imaginile arbitrară sunt alcătuite dintr-un număr mare de pixeli mici, în timp ce alte afișaje au elemente mai mari. Ecranele LCD pot fi în mod normal on (pozitiv) sau pe off (negativ), în funcție de aranjamentul polarizatorului. De exemplu, un LCD cu caractere pozitive cu iluminare din spate va avea inscripții negre pe un fundal care este culoarea luminii de fundal, iar un LCD negativ de caractere va avea un fundal negru, literele fiind de aceeași culoare ca luminile de fundal. Filtrele optice sunt adăugate la alb pe ecranele albastre pentru a le oferi aspectul lor caracteristic.



Programare



The screenshot shows the Arduino IDE interface with the sketch titled "Temperatura_Umiditate_analog". The code uses the LiquidCrystal library to interact with an LCD display. It reads analog values from pins A1 and A2, maps them to a range of 0-255, and then converts them to percentages (51.00% and 21.00°C) to be displayed on the LCD.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

//const int analogInPin1 = A1;
//const int analogInPin2 = A2;
//const int analogOutPin = 9;

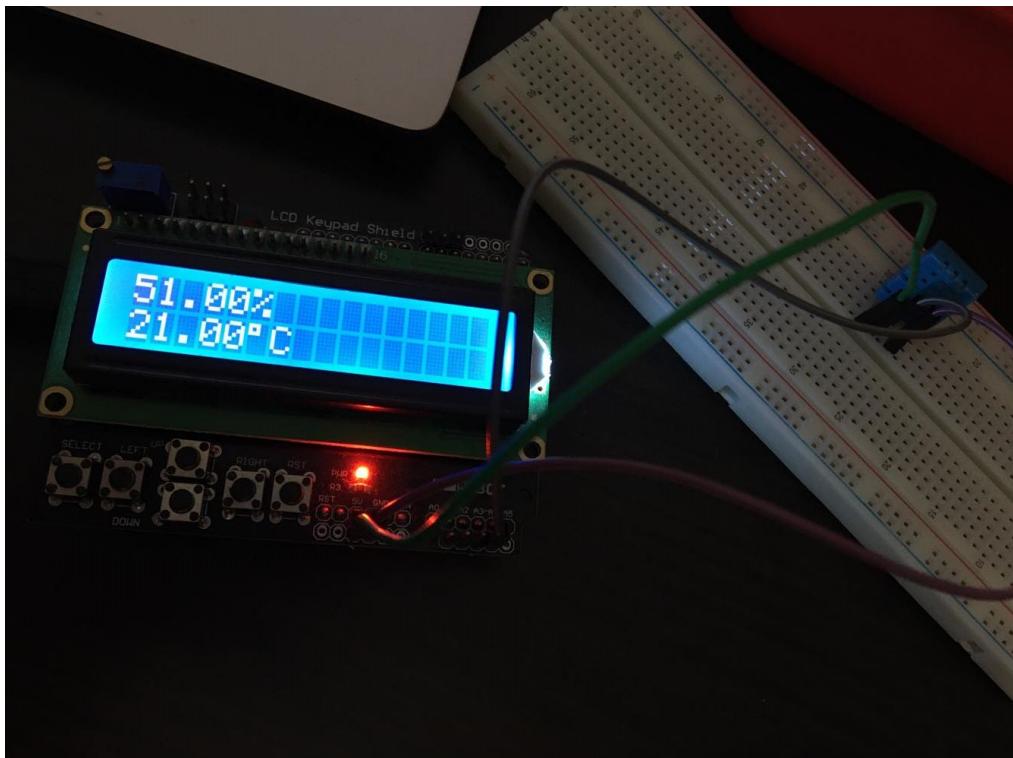
int sensorValue1 = 0;
int outputValue1 = 0;
int sensorValue2 = 0;
int outputValue2 = 0;

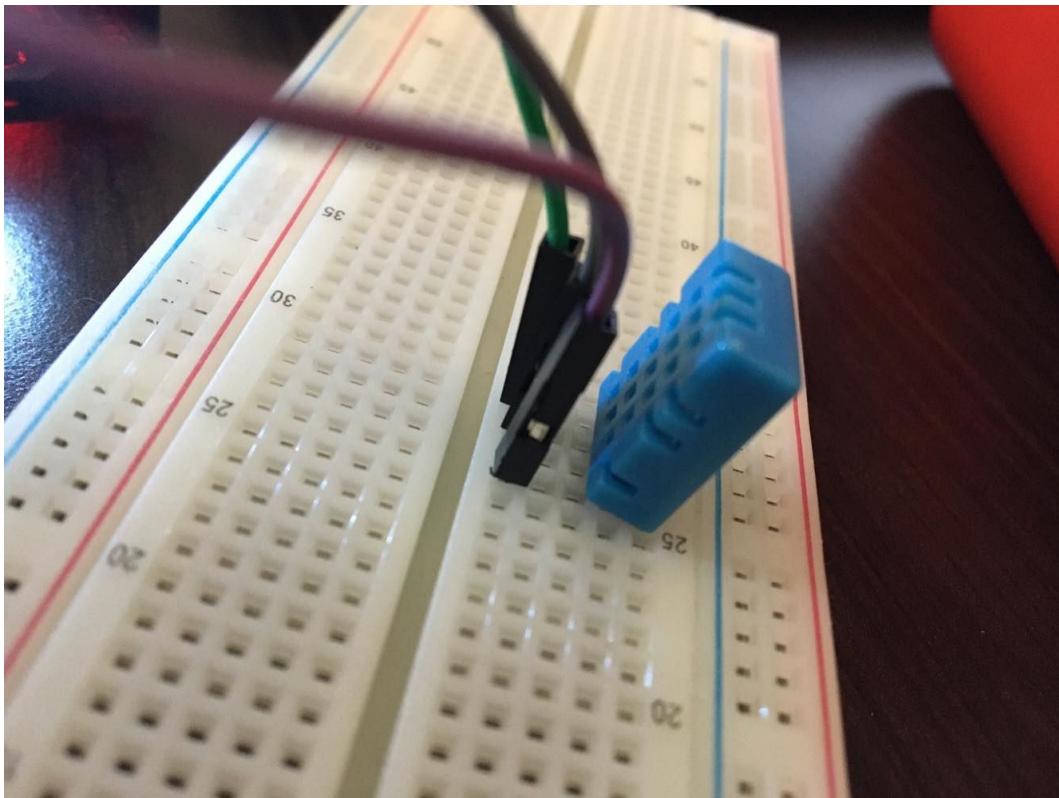
void setup()
{
  lcd.begin(16, 2);
}

void loop() {
  // read the analog in value:
  sensorValue1 = analogRead(analogInPin1);
  sensorValue2 = analogRead(analogInPin2);
  // map it to the range of the analog out:
  //outputValue = map(sensorValue, 0, 1023, 0, 255);

  outputValue1 = sensorValue1*0.45+4;
  outputValue2 = sensorValue2*0.0853375;

  lcd.setCursor(0,0);
  lcd.print("Temp.: ");
  lcd.print(outputValue1);
  lcd.print("%");
  lcd.setCursor(0,1);
  lcd.print("Umiditate:");
  lcd.print(outputValue2);
  lcd.print("°C");
}
```





CONCLUZII

Un microcontroler este un microcircuit programabil care încorporează o unitate centrală (CPU) și o memorie împreună cu resurse care-i permit interacțiunea cu mediul exterior. Sistemele cu microcontroler sunt din ce în ce mai mult folosite în procesele de control automat, atât datorită costurilor mici pe care le oferă această tehnologie în momentul de față, cât mai ales înaltei flexibilități aplicative. Prin utilizarea senzorilor analogici cu ajutorul unor formule de calcul se pot măsura corect unele mărimi neelectrice (temperatură și umiditate). Senzorii digitali, care trimit informația în format binar, permit realizarea unor programe mai mici. În aplicațiile practice este mult mai ușor să se utilizeze module (Shield) LCD care se pot conecta direct pe modulele Arduino, comparativ cu display-urile LCD simple care necesită mai multe conductoare de legătură.

BIBLIOGRAFIE

https://en.wikipedia.org/wiki/Arduino_Unc

<https://www.sparkfun.com/products/11061>

<http://www.ti.com/lit/ds/symlink/lm35.pdf>

<https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>

Nume, prenume:

Rusu Bianca-Elena, Timofte Teodora-Andreea

Grupa: 1406B

Tema de proiect:

Telegram Bot folosind Raspberry Pi, ce realizeaza aprinderea unui led simultan cu un mesaj (ON/OFF) pe display.

Rezumat:

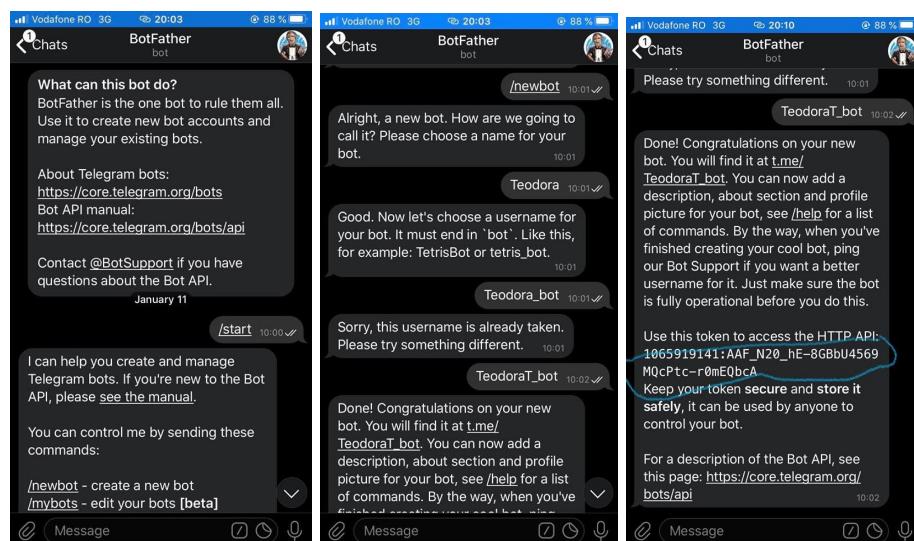
Proiectul consta in instalarea aplicatiei de mesagerie Telegram, disponibila atat pe Android cat si iOS, pe un device si crearea unui bot ce va comunica cu placuta, in scopul aprinderii unui led in acelasi timp cu afisarea pe un display a mesajului corespunzator starii ledului.

Explorarea documentara asupra temei:

Telegram este un serviciu de mesagerie instantă și VoIP dezvoltată de Telegram Messenger LLP, companie înregistrată în Londra, Regatul Unit, și înființată de antreprenorul rus Pavel Durov. Versiuni ale aplicației sunt disponibile pentru Android, iOS, Windows Phone, Windows NT, macOS și Linux.



Dupa instalarea aplicatiei Telegram pe un smartphone, se deschide o conversatie noua cu BotFather(ce reprezinta bot-ul principal care le controleaza pe celelalte) si se trimit mesajul: "/start", apoi se creaza un nou bot folosind comanda: "/newbot", careia i se va trebui un nume la alegere cu terminatia "_bot". Din ultimul mesaj primit de la BotFather se extrage cheia necesara accesarii HTTP API.

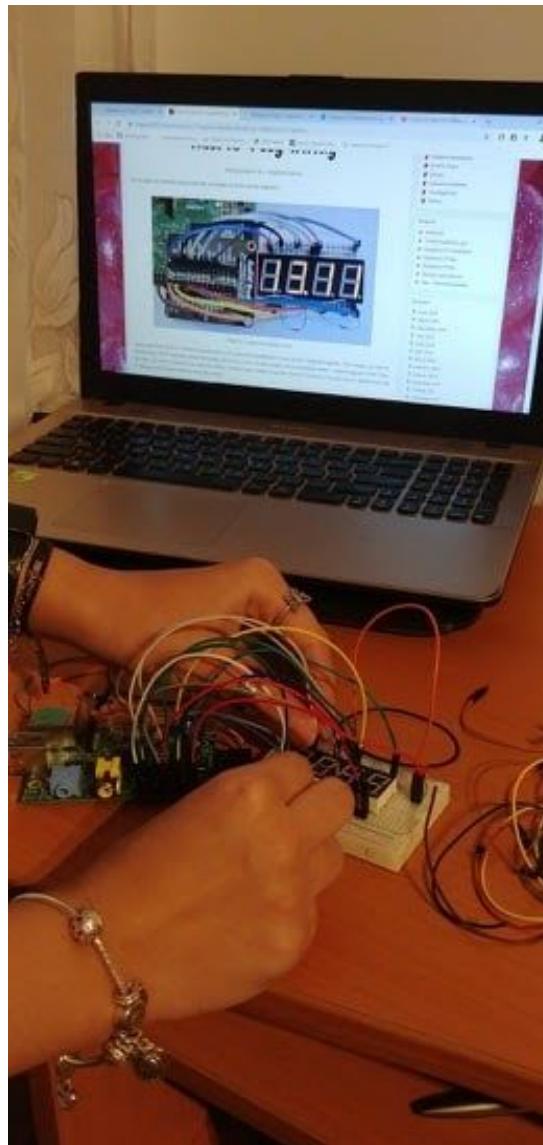


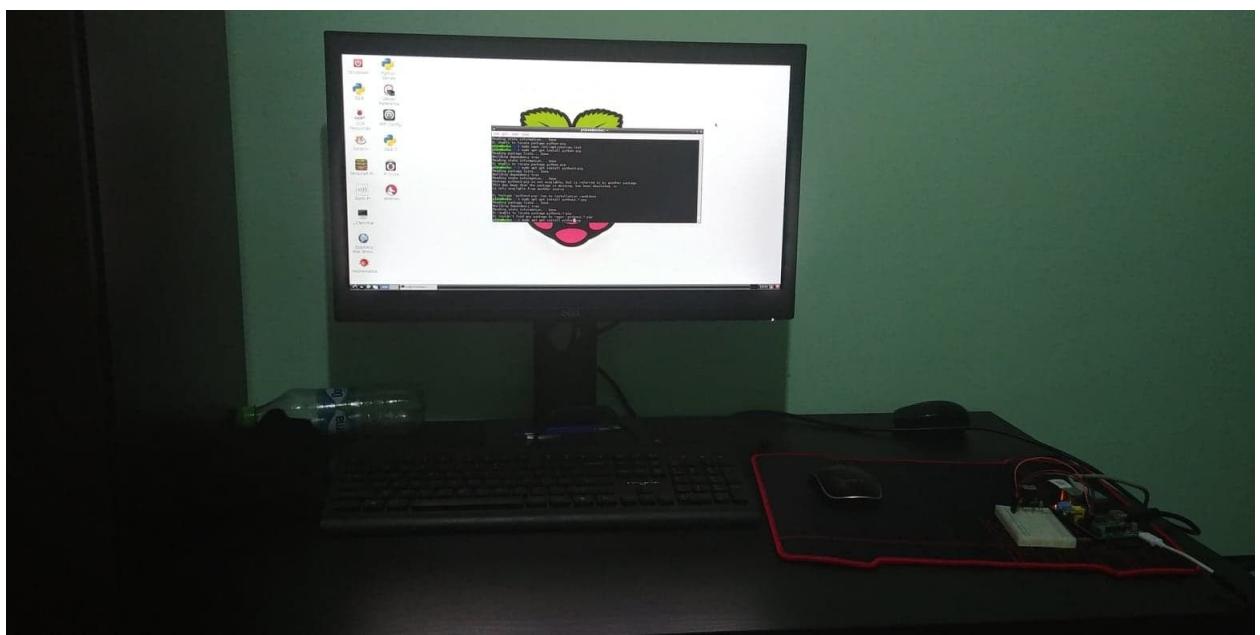
Pentru configurarea placutei Raspberry Pi 3 Model B, s-au realizat urmatorii pasi:

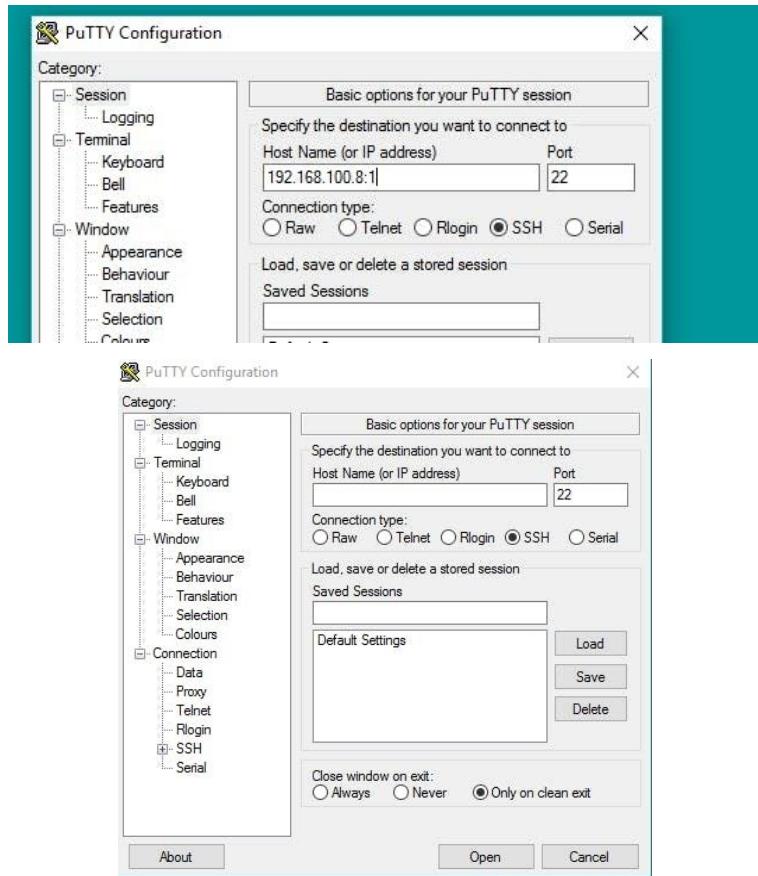
- Conectarea cablului de alimentare, a cablului de internet, cablul HDMI pentru conectarea monitorului, dar si mouse-ului si al tastaturii prin porturile USB.
- S-a instalat programul Raspbian.
- SSH este preinstalat pe Raspberry Pi.

Pentru rularea Telegram bot-ului pe Raspberry Pi am folosit PuTTY, conectandu-ne prin SSH. In linia de comanda ne logam cu user-ul: pi si parola: raspberry.

Se instaleaza telepot pentru a ne ajuta sa facem conexiunea intre interfata telefonului si placa folosita.





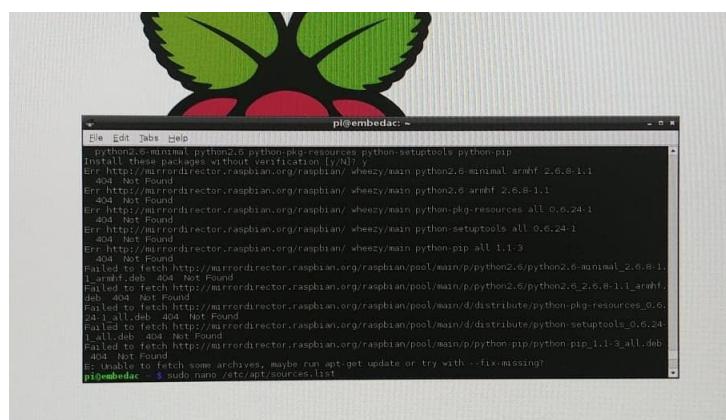


Se ruleaza codul in Python:

```
bot = telepot.Bot('Bot Token') unde Bot Token este cheia obtinuta de la BotFather;
python telegrambot.py
```

Se conecteaza ledul si display-ul pe breadboard folosind fire si 8 rezistente.

Revenim la aplicatia Telegram, se ruleaza comanda "/start" urmata de "On" sau "Off".



Mod de functionare:

On - ledul este activ, iar pe display se afiseaza mesajul "On";

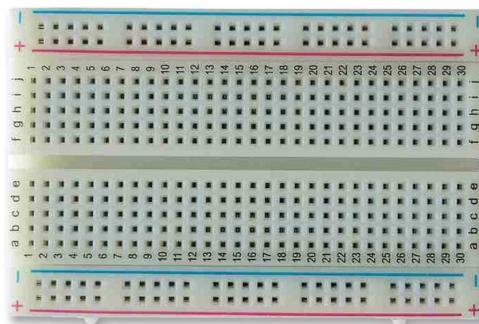
Off - ledul este stins, pe display apare mesajul "Off";

Materiale necesare:

- Raspberry Pi 3 Model B



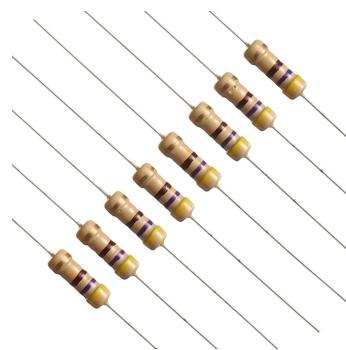
- Breadboard



- Led



- 8 Rezistori



- Display 12 pini, 4 digits



- Fire



Codul sursa:

```
import sys
import time
import random
import datetime
import telepot
import RPi.GPIO as GPIO

O = [1,1,1,1,1,1,0]
F = [1,0,0,0,1,1,1]
n = [0,0,1,0,1,0,1]
}

segments = (12, 13, 19, 16, 26, 20, 21)
```

```
#Digit 1
GPIO.setup(7, GPIO.OUT)
GPIO.output(7, 0) #Off initially
#Digit 2
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, 0) #Off initially
#Digit 3
GPIO.setup(25, GPIO.OUT)
GPIO.output(25, 0) #Off initially
```

```

#Digit 4
GPIO.setup(24, GPIO.OUT)
GPIO.output(24, 0) #Off initially

# GPIO ports for the 7seg pins
##### A B C D E F G
for segment in segments:
    GPIO.setup(segment, GPIO.out, initial=GPIO.LOW)
#LED

def print_segment(letter):
    if letter == 1:
        for i in range(7):
            GPIO.output(segments[i], O[i])

    if letter == 2:
        for i in range(7):
            GPIO.output(segments[i], F[i])

    if letter == 3:
        for i in range(7):
            GPIO.output(segments[i], n[i])
return;

def on(pin):
    GPIO.output(pin,GPIO.HIGH)
    return
def off(pin):
    GPIO.output(pin,GPIO.LOW)
    return
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)
# set up GPIO output channel
GPIO.setup(11, GPIO.OUT)

delay_time = 0.001

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print 'Got command: %s' % command

    if command == 'on':
        bot.sendMessage(chat_id, on(11))
        GPIO.output(7, 1) #Turn on Digit One
        print_segment(1)
        time.sleep(delay_time)

```

```
GPIO.output(8, 1) #Turn on Digit Two
print_segment(3)
time.sleep(delay_time)

elif command =='off':
    bot.sendMessage(chat_id, off(11))
    GPIO.output(7, 1) #Turn on Digit One
    print_segment(1)
    time.sleep(delay_time)
    GPIO.output(8, 1) #Turn on Digit Two
    print_segment(2)
    time.sleep(delay_time)
    GPIO.output(25, 1) #Turn on Digit Two
    print_segment(2)
    time.sleep(delay_time)

bot = telepot.Bot('Bot Token')
bot.message_loop(handle)
print 'I am listening...'

while 1:
    time.sleep(10)
```

Nume, prenume:

Ungureanu Gabriela

Grupa:

1405B

Tema de proiect:

Simularea unei pedale de acceleratie

Rezumat:

Proiectul consta in simularea unei pedale de acceleratie prin intermediul unui potentiometru. Valoarea preluata de pe acesta va fi transmisa in retea pentru a putea fi folosita ca semnal de comanda pentru un motor DC.

Mod de funcționare:

Utilizatorul va actiona potentiometrul pentru a simula cuplul dorit pentru pedala de acceleratie, iar valoarea de pe potentiometru va fi trimisa in retea la un interval de 0.5s.

Explorarea documentara asupra temei:

- <https://www.hackster.io/gawad/gas-pedal-for-simulator-game-fb73c7>
- http://blog.whaleygeek.co.uk/wp-content/uploads/2013/10/NetworkMessaging-Crisheet.pdf?fbclid=IwAR2Cabb2KgSYDsZkAk_VOdDsUqVn51kDb6dNQLw6R5T5TdTL0geV-luj6lk
- <https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/ads1015-slash-ads1115>
- <https://www.hackster.io/jonmendenhall/mini-gaming-wheel-with-gas-brake-pedals-92acc2>
- <https://cdn-shop.adafruit.com/datasheets/ads1015.pdf>

Ciornă de soluție:

Soluția constă în realizarea unul sistem care va simula pedala de acceleratie, utilizand un Raspberry Pi 3 B+. Pentru preluarea valorii de pe potentiometru se va folosi un ADC ADS1115 cu protocol de comunicatie I2C.

Semnalul preluat de pe potentiometru va fi scalat si va fi trimis in retea, iar valoarea acestuia reprezinta duty cycle-ul corespunzator pentru semnalul pwm ce va actiona

motorul DC. Pentru transmiterea valorii respective in retea, se va rula un script “client” care va transmite valorile catre un “server” care ar putea procesa informatiile primite si le va trimite pentru a actiona componenta corespunzatoare.

Resurse hardware:

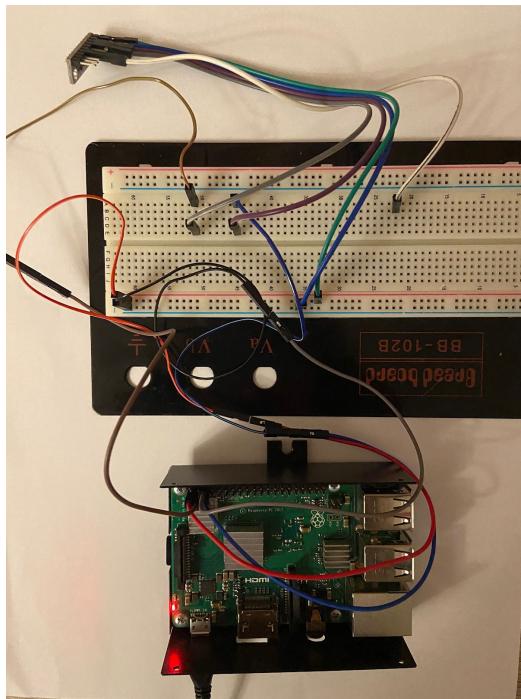
- Raspberry Pi 3 B+
- Fir de conexiune
- Breadboard
- ADC ADS1015 / ADS1115 pe 12 biti
- Potentiometru b10k

Resurse software:

- VNC Viewer
- Adafruit_Python_ADS1x15 library
- Network library
- Time library
- Python 3

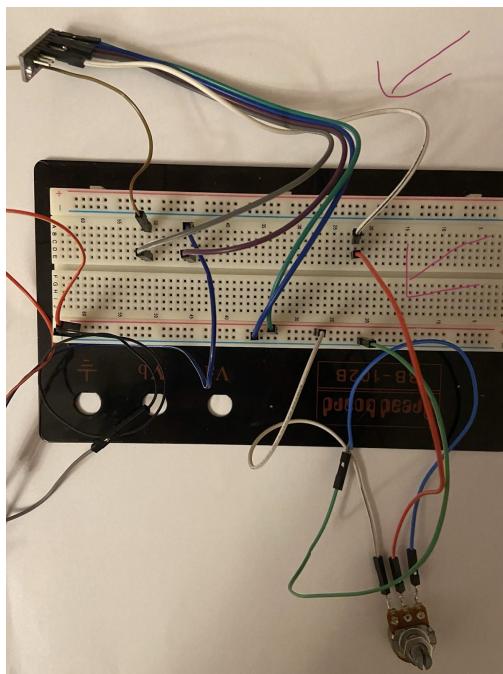
Interfatarea intre componente:

- Interfatare ADC cu Raspberry Pi:
 - ➔ Pin ADC VDD la Raspberry Pi 3.3V
 - ➔ Pin ADC GND la Raspberry Pi GND
 - ➔ Pin ADC SCL la Raspberry Pi SCL
 - ➔ Pin ADC SDA la Raspberry Pi SDA

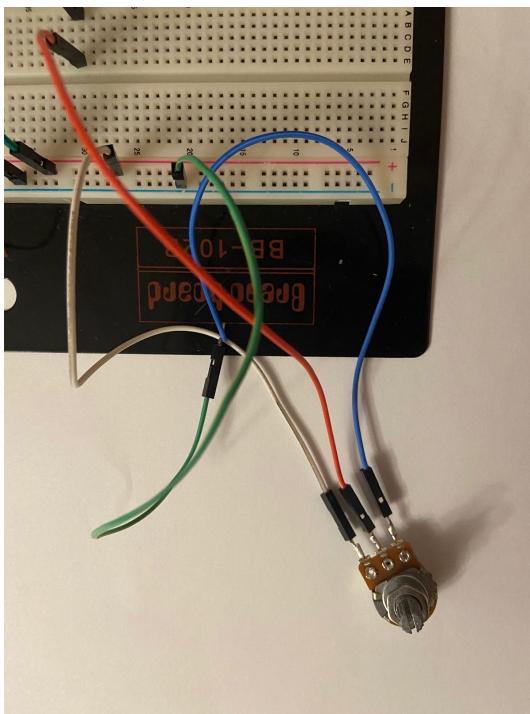


- Interfatare intre ADC si potentiometru (valorile de pe potentiometru vor fi preluate de canalul analogic A0 al ADC-ului, insa ar fi putut fi folosit oricare dintre cele 4 canale):

→ Pin ADC A0 la pinul de iesire (data) al potentiometrului



- Alimentare potentiometru (pentru cresterea valorii citite in sens orar):
 - Pinul din extremitatea stanga la Raspberry Pi GND
 - Pinul din extremitatea dreapta la Raspberry Pi 3.3V



Coduri sursa:

- Client

```
import network  
  
import sys  
  
import time  
  
import Adafruit_ADS1x15  
  
adc = Adafruit_ADS1x15.ADS1115()  
  
GAIN = 1
```

```
def heard(phrase):

    print ("them:" + phrase)

print ("Chat Program")

if (len(sys.argv) >= 2):

    network.call(sys.argv[1], whenHearCall=heard)

while True:

    values = adc.read_adc(0, gain=GAIN)

    values = values/260

    if (values > 100):

        values = 100

    values = round(values)

    print('{0:^6}'.format(values))

    phrase = str(values)

    print ("me:" + phrase)

    network.say(phrase)

    time.sleep(0.5)
```

- Server

```
import network

import sys

def heard(phrase):

    print ("them:" + phrase)

print ("Chat Program")

if (len(sys.argv) >= 2):

    network.call(sys.argv[1], whenHearCall=heard)

else:

    network.wait(whenHearCall=heard)

print ("Connected!")

while network.isConnected():

    a = 1
```

Nume:

Alexandru Iulian

Grupa:

1405B

Tema project:

Senzor de parcare

Rezumat:

Proiectul consta in folosirea unui senzor de proximitate cu ultrasunete care determina in mod continuu distanta pana la cel mai apropiat obstacol. Daca se gaseste un obiect la o distanta mai mica de 50 de centimetri, se va aprinde un led si se va trimite un mesaj in retea.

Explorarea documentă asupra temei:

- ❖ <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

- ❖ <http://blog.whaleygeek.co.uk/wp-content/uploads/2013/10/NetworkMessaging-Cribsheet.pdf?fbclid=IwAR0GztM8vxwT3MdgFSiemzmDyqu0Dq1P4loSKXix0v-UTIWu8QDYLQgRXuA>
- ❖ <https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/>
- ❖ <https://thephihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>
- ❖ <https://www.raspberrypi.org/documentation/usage/gpio/>
- ❖ <https://raspberrypihq.com/making-a-led-blink-using-the-raspberry-pi-and-python/>

Resurse hardware:

- ❖ Raspberry Pi Zero W
- ❖ Ultrasonic Ranging Module HC - SR04
- ❖ Breadboard
- ❖ LED
- ❖ Rezistente
- ❖ Fir de conexiune

Resurse software:

- ❖ PuTTY
- ❖ VNC Viewer
- ❖ Python 3.7.3

Cod sursa:

Aplicatia Client:

```
import RPi.GPIO as GPIO
import time
import network
import sys
#GPIO.setwarnings(False)

def heard(phrase):
    print ("them:" + phrase)
```

```
GPIO.setmode(GPIO.BOARD)
```

```
GPIO_TRIGGER = 18
GPIO_ECHO = 16
GPIO_LED = 40
```

```
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
```

```

GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(GPIO_LED, GPIO.OUT)

LED_ON = 1

def distance():
    GPIO.output(GPIO_TRIGGER, True)

    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2

    return distance

try:
    if (len(sys.argv) >= 2):
        network.call(sys.argv[1], whenHearCall=heard)
    else:
        print("IP address needed")
        exit()

    while network.isConnected():
        phrase = distance()
        if phrase <= 50:
            LED_ON = 0
        else:
            LED_ON = 1
        GPIO.output(GPIO_LED, LED_ON)

```

```
network.say(str(phrase))
time.sleep(1)
except KeyboardInterrupt:
    print("Process stopped by user")
    GPIO.cleanup()
    exit()
#####
```

Aplicatia Server:

```
import network
import sys
import time

def heard(phrase):
    print ("message: " + phrase)

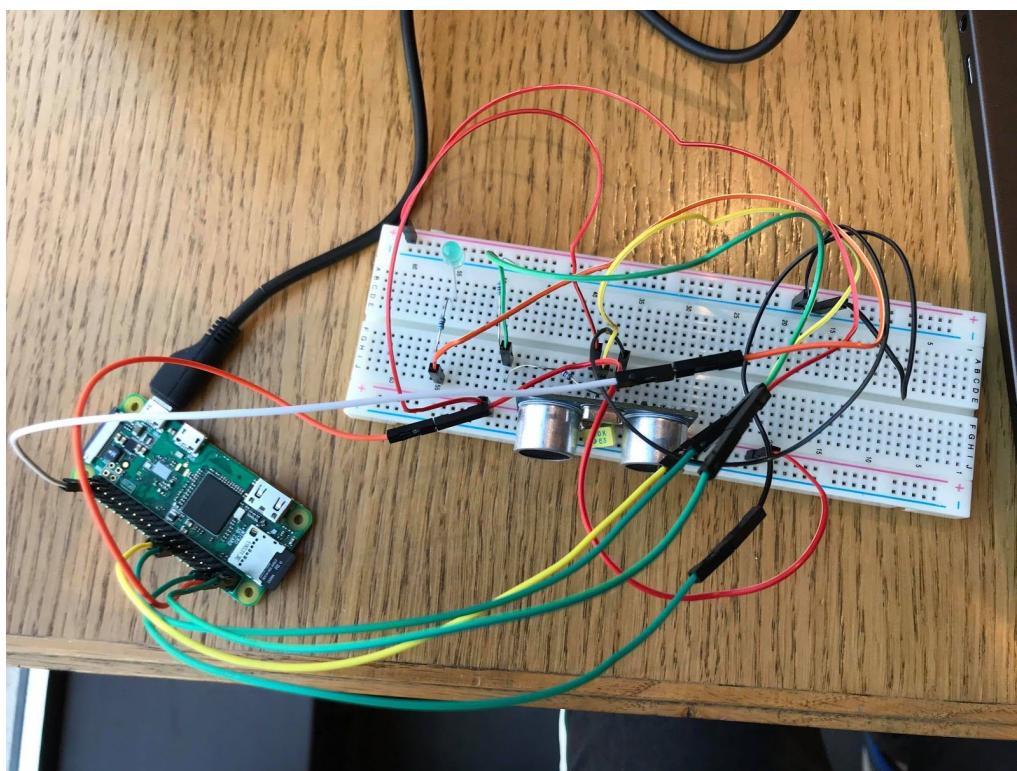
if (len(sys.argv) >= 2):
    network.call(sys.argv[1], whenHearCall=heard)
else:
    network.wait(whenHearCall=heard)

while network.isConnected():
    time.sleep(1)
```

Poze si capturi de ecran:

```
pi@raspberrypi:~/SI  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon Jan 13 13:09:08 2020 from fe80::8dd6:a344:5cc3:34bf%wlan0  
pi@raspberrypi:~ $ cd SI  
pi@raspberrypi:~/SI $ python3 server.py  
message: 36.93892955780029  
message: 37.625861167907715  
message: 38.308703899383545  
message: 36.992084980010986  
message: 38.54994773864746  
message: 40.59029817581177  
message: 37.51955032348633  
message: 39.53127861022949  
message: 39.80523347854614  
message: 77.36567258834839  
message: 43.30531358718872  
message: 77.97900438308716  
message: 41.09323024749756  
message: 40.71296453475952  
message: 40.66389799118042  
message: 41.060519218444824  
message: 41.50211811065674  
message: 41.07687473297119  
message: 42.107272148132324  
message: 43.09678077697754  
message: 40.267276763916016
```

```
pi@raspberrypi:~/SI $ python3 blink_led.py 192.168.1.194  
^CProcess stopped by user  
pi@raspberrypi:~/SI $
```



Tema proiectului: Aprinderea si stingerea luminii prin intermediul vocii
(Utilizand Alexa)

Studenti: Bors Mircea si Condurat Florin

Grupa: 1405A

Cuprinsul proiectului:

1. Descrierea functionala a sistemului
2. Componente necesare
3. Implementare
4. Surse Utile, Referinte

1. Descriere

Efectuarea unui sistem ce va aprinde/stinge lumina prin intermediul vocii.

Comenzile vocale sunt preluate de catre Amazon Alexa Echo Dot, si sunt redirectate catre Raspberry Pi 4. Becul se va aprinde printr-o comanda speciala lui. Exemplu: “ Alexa, ask raspberry pi to turn light on”

Becul va fi aprins/inchis de Raspberry Pi prin intermediul releului.

2. Componentele necesare sistemului:

Raspberry Pi 4 4gb



Releu Songle



Amazon Alexa Echo Dot versiunea 2



Un bec cu dulie si 3 fire de conexiune intre releu si Raspberry Pi cu mufa de tip mama.

3. Implementare

Primul pas din implementarea proiectului a fost instalarea sistemului de operare NOOBS Raspbian pe placuta Raspberry Pi 4 cu ajutorul unui SD card, iar apoi am ne-am conectat laptopul personal la placuta cu ajutorul IP-ului prin Putty SSH si VNC Viewer, astfel am putut sa lucram mai usor.

Amazon a creat o librarie python numita “flask-ask” care accepta apeluri si comenzi de la Alexa.

Al doilea pas din proiectul nostru este reprezentat de conexiunea placutei Raspberry Pi la un HTTPS endpoint prin crearea unui tunel cu ajutorul NGROK. NGROK creeaza instant un url public care ruleaza local. Astfel am creat conexiunea intre Alexa si Raspberry Pi.

Al treilea pas: Scriptul Python care importa modulul flask mentionat anterior si modulul GPIO (General Purpose Input/Output si descrie pinii placutei Raspberry Pi) folosit pentru pinul 3 ca un output. Atunci cand vom da comanda vocala “on” setam pinul pe HIGH, iar la “off” pe LOW. Apoi la final avem statementul “Turning {} car room light”.

Al patrulea pas: Avand conexiunea intre placuta si Alexa, scriptul python facut, nu ramane decat programarea Alexei utilizand Alexa Skill, implementand un skill nou. Astfel am reusit sa prelucram comanda vocala in acces API.

```
"interactionModel": {  
    "languageModel": {  
        "invocationName": "raspberry pi",  
        "intents": [  
            {  
                "name": "AMAZON.CancelIntent",  
                "samples": []  
            },  
            {  
                "name": "AMAZON.HelpIntent",  
                "samples": []  
            },  
            {  
                "name": "AMAZON.StopIntent",  
                "samples": []  
            },  
            {  
                "name": "AMAZON.NavigateHomeIntent",  
                "samples": []  
            },  
            {  
                "name": "TurnLight",  
                "slots": [  
                    {  
                        "name": "slot1",  
                        "type": "String"  
                    }  
                ]  
            }  
        ]  
    }  
}
```

```
{  
    "name": "status",  
    "type": "GPIO_CONTROL"  
}  
,  
"samples": [  
    " to turn {status} the lights",  
    " to turn lights {status}"  
]  
}  
,  
"types": [  
    {  
        "name": "GPIO_CONTROL",  
        "values": [  
            {  
                "name": {  
                    "value": "on"  
                }  
,  
                {  
                    "name": {  
                        "value": "off"  
                    }  
                }  
            }  
        ]  
    }  
]
```

Ultimul pas este reprezentat de conexiunea releului la placuta si diode.

Releu are doua parti, partea de control si partea switch. Pe partea de control avem DC+, DC- si IN si este partea conectata la RPi. DC+ este conectat la 5v, DC- la GND si IN la pinul GPIO mentionat mai sus, pinul 3. Pe partea switch-ului avem NO (Normally Open adica curentul nu circula decat atunci cand releul este activat), NC (Normally Closed opusul NO, adica curentul circula pana releul este activat, si COM care este inputul. Releul se activeaza la un semnal HIGH la pinul IN. In cazul de fata am folosit NO.

4. Surse Utile, Referinte

<http://www.circuitbasics.com/raspberry-pi-basics-setup-without-monitor-keyboard-headless-mode/>

<https://alexatutorial.com/flask-ask/>

<https://thisdavej.com/how-to-host-a-raspberry-pi-web-server-on-the-internet-with-ngrok/>

<https://developer.amazon.com/en-US/docs/alexa/smarthome/understand-the-smart-home-skill-api.html>

light.py - Notepad

```
File Edit Format View Help
from flask import Flask
from flask_ask import Ask, statement, convert_errors
import RPi.GPIO as GPIO
import logging
GPIO.setmode(GPIO.BCM)
app = Flask(__name__)
ask = Ask(app, '/')
logging.getLogger("flask_ask").setLevel(logging.DEBUG)

@ask.intent('TurnLight', mapping={'status':'status'})
def turn(status):
    GPIO.setup(3,GPIO.OUT) #configure for the specific pin
    if(status=='on'):
        GPIO.output(3,GPIO.HIGH)
    elif(status=='off'):
        GPIO.output(3,GPIO.LOW)
    return statement('Turning {} car room light'.format(status))

if __name__ == '__main__':
    port = 5000 #the custom port you want
    app.run(host='0.0.0.0', port=port)
```