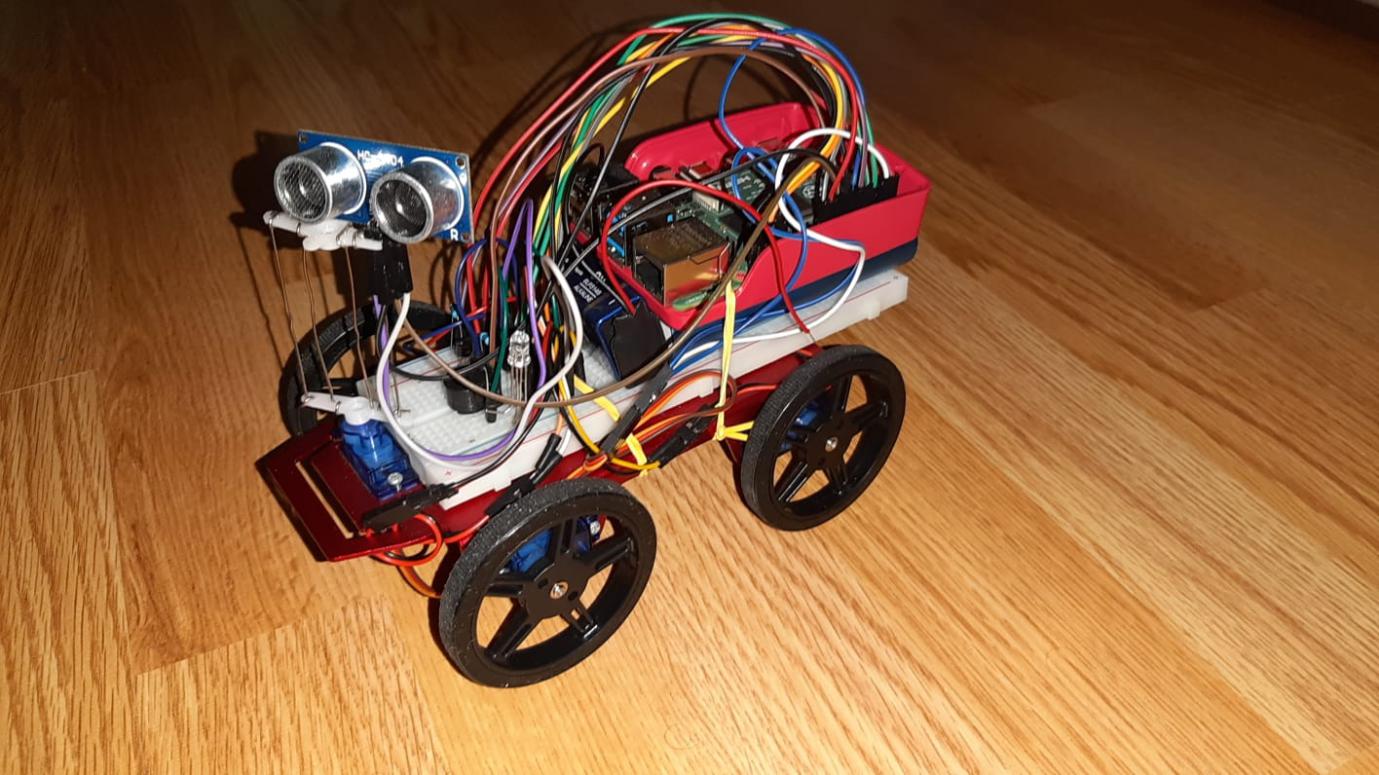
**Proiect Practică Anul 2**

**Robot Controlat de pe Telefon, cu Senzor de Distanță pentru Identificarea Obstacolelor**



Îndrumător proiect Student sg. Maftei Silviu

slt. ing. Popovici Cristina Grupa E113-B

-Galați-

Cuprins

1. Ideea proiectului. Rezumat general............................................................3
2. Componente folosite. Justificarea folosirii lor...........................................3
3. Calculele necesare proiectului....................................................................6
4. Raspberry Pi. Conectarea sa cu circuitul....................................................6
5. Explicarea codului în Python......................................................................8
6. Explicarea funcționării fișierului HTML aferent codului.........................10
7. Funcționare.Sistemul Linux.……............................................................12
8. Concluzii. Posibile îmbunătățiri...............................................................13
9. Bibiografie/Siteografie.............................................................................15
10. Ideea proiectului. Rezumat general
11. Scurtă prezentare

Proiectul realizat de mine este, după cum îi sugerează numele, un robot, având în principal scopul de a cerceta zona din fața sa, reprezentată printr-un câmp de 180 de grade, permițând scanarea acestui câmp, și avertizarea sonoră și auditivă asupra existenței vreunui obstacol.

El se bazează pe placa de dezvoltare Raspberry Pi, o plăcuță mică, dar foarte puternică. Codul programului care rulează (putem spune în buclă infinită, de la rulare până când îi este comandată oprirea) este scris în limbajul Python. O parte semnificativă a lui, controlarea, a fost scrisă într-un editor text simplu și rulată prin html, ceea ce permite oricărui utilizator care este conectat la rețeaua de internet să controleze robotul.

O mențiune specială trebuie făcută asupra faptului că proiectul a început ca un pat de testare pentru diferitele funcționalități ale plăcuței și ale componentelor aferente. Deși ele funcționează foarte bine individual, a fost o provocare să creez un program care să reușească să le coordoneze pentru ca stagiul final să fie un produs finit.

1. Rezumat general

Pe scurt, proiectul a început cu ideea atașării unui senzor de distanță pe un servomotor ce permite determinarea câmpului înainte-menționat. Ulterior, m-am decis să îl atașez unei platforme mobile, robotul. Una din provocări a constat în controlarea sa de la distanță. Odată rezolvată această problemă, i-am atașat avertizarea luminoasa și sonoră. Finalul a constat în unirea tuturor părților de cod, fiind implementate sub forma unor funcții în Python.

1. Componentele folosite. Justificarea folosirii lor.
2. Componentele folosite

În cadrul proiectului, se regăsesc următoarele componente :

-placa de dezvoltare Raspberry Pi

-baterie externă (pentru alimentarea plăcuței, conferindu-i independență față de cablu de alimentare conectat la priză)

-4 motoare de curent continuu cu rotație continuă

-L293D H-Bridge – modul de conducere al motoarelor (sub forma unui CI de 16 pini)

-servomotor, cu rotație de 180 de grade

-senzor de distanță ultrasonic HC-SR04

-LED RGB

-buzzer

-rezistoare (1kΩ și 100Ω)

-tranzistor (pentru controlarea buzzer-ului)

-diodă (deservește buzzer-ul)

-diverse componente mecanice, precum șasiul robotului, roți, șuruburi, piulițe, elastice flexibile, agrafe

1. Justificarea folosirii lor

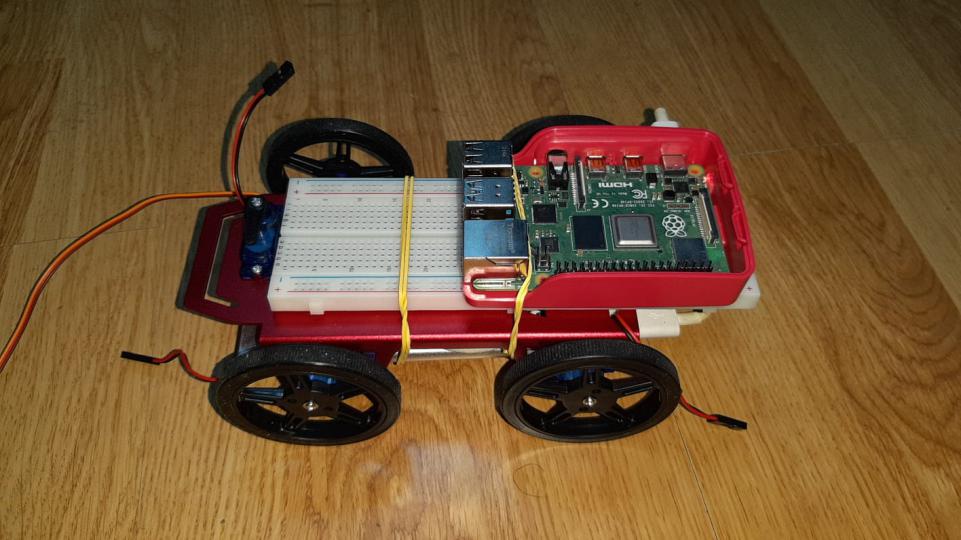
Putem separa proiectul fizic în 2 subcapitole - partea mecanică și partea electrică.

Partea mecanică înglobează tot ce nu ține de Raspberry Pi, și conferă mobilitatea mecanică pentru diferitele funcționalități ale proiectului. Tot aici, intervine și modalitatea de poziționare a elementelor pe șasiu.

Au fost atașate motoarele în spațiile destinate lor, câte 2 pe fiecare parte. Inițial, robotul a avut doar 2 roți și o roată mobilă pe centru însă, demonstrând mobilitate superioară în sistemul 4x4, așa am hotărât să arate robotul. De menționat este că, pentru simplificarea schemei, motoarele de pe aceeași parte sunt conectate între ele.

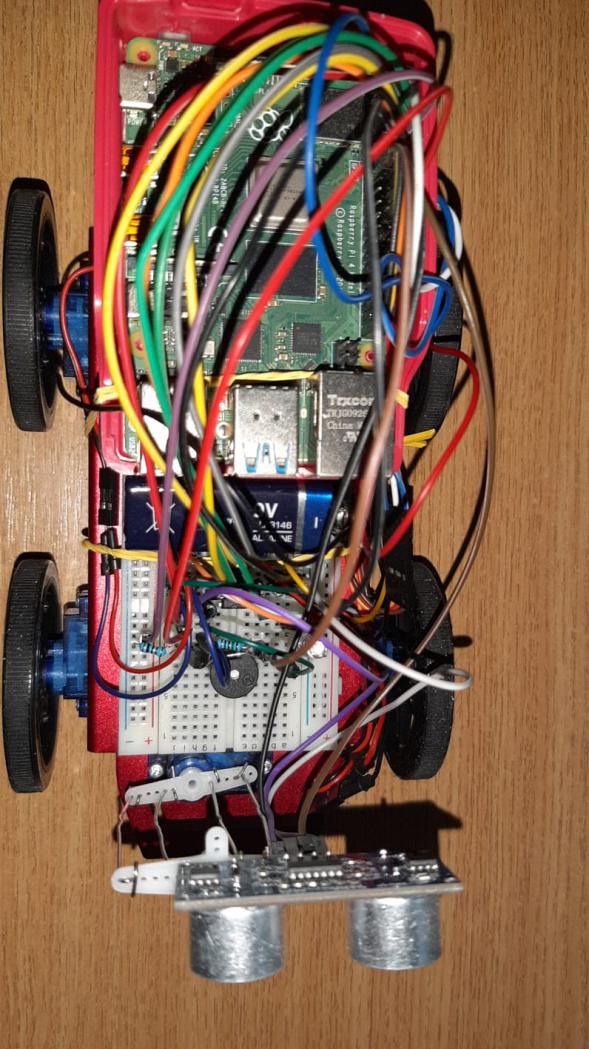
Din cele 5 motoare, 4 sunt de curent continuu și rotație continuă, fiind înșurubate de roți. Al 5lea este un servomotor, prins de caroserie vertical, fiind conectat de senzorul ultrasonic printr-o serie de agrafe desfășurate.

Benzile elastice sunt folosite pentru a imobiliza structura robotului la șocurile mecanice (inerția deplasării).



Imaginea 1 - vedere asupra părții mecanice

Partea electrică reprezintă tot ce se conectează direct la Raspberry Pi. Acestea sunt elementele de tip senzorul de distanță, modalitățile de avertizare audio și sonoră, controlarea motoarelor prin CI L293 și controlarea servomotorului.



Imaginea 2 - vedere asupra părților electrice și a conectivității

O importanță semnificativă o are circuitul integrat înainte-menționat, al cărui principiu de funcționare este reprezentat în figura 3.

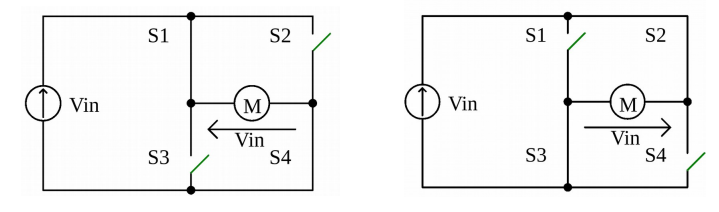


Figura 3 - Funcționarea IC293

Se observă cum, prin controlarea celor 2 întrerupătoare, putem controla sensul de rotație al motorului, notat cu M. Aceste întrerupătoare sunt comandate de pinii plăcuței de test, în așa fel încât motoarele se sincronizează, și robotul se deplasează în sensul dorit de utilizator.

1. Calculele necesare proiectului

Proiectul nu necesită multe calcule, însă cele prezente vor fi menționate aici.

Întâi de toate, trebuie aleși pinii de alimentare, dintre 3.3V sau 5V. Pe scurt, am ales pinul care convine cel mai mult situației necesare (3.3V pentru CI 293 și LED, 5V pentru buzzer, senzor, servomotor).

Ulterior, trebuie calculată valoarea rezistențelor care trebuie legate în serie cu LEDul RGB, pentru a evita arderea lui. Pentru a păstra schema de circuit simplă, am ales conectarea unui singur rezistor, la masa acestuia.

Rs = (Vs-Vd)/Id . – formula calculului rezistenței în serie cu LEDul

Pentru valorile ideale ale LED-ului roșu, adică Vd de 1.7V și Id de 15mA, regăsim valoarea rezistenței de 106 Ω.

Pentru valorile ideale ale LED-ului verde, adică Vd de 2V și Id de 15mA, regăsim valoarea rezistenței de 86 Ω.

Am ales valoarea medie de 100Ω.

Tot aici apare problema alimentării motoarelor cu rotație continuă. Am ales să le conectez la 9V, pentru a reuși să dezvolte un cuplu de forțe mai puternic, în special pentru a căra bateria externă care, deși oferă independență de o sursă statică de curent, adaugă o masă semnificativă robotului. Conform producătorului, la 5V oferă un cuplu de 1.3kg/cm, iar la 9V un cuplu de 2kg/cm (semnificativ mai mare). Acest calcul a fost obținut prin trasarea dreptei caracteristice tensiune-cuplu.

1. Raspberry Pi. Conectarea sa cu circuitul

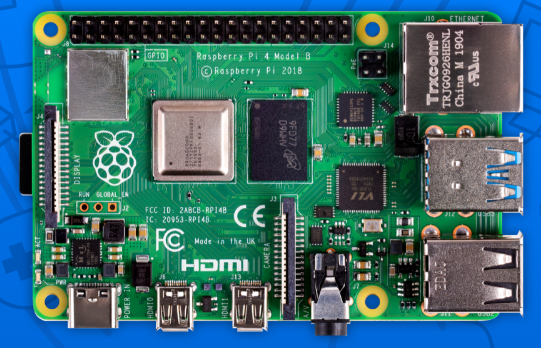


Figura 4 - Vedere superioară asupra plăcuței

Raspberry este o plăcuță de test, un computer de dimensiunea unui card de credit, care, la bootare, începe să ruleze un sistem de operare bazat pe Linux. În acest mediu, a fost scris codul care rulează indefinit în fundal, care reprezintă componenta soft a proiectului.

Parametrul său fundamental este frecvența la care poate executa instrucțiunile, care se ridică la 1.5 GHz

Figura 4 arată alocarea pinilor programabili pe placă. Conexiunile au fost realizate astfel :

Pinii 1, 2 și 4 -> Sursele de 5V și 3.3V ale circuitului

7, 8 -> Rutare Logica Motoare (stanga-dreapta)

10, 22, 35, 37 -> Direcție Motoare

11 -> Servomotor frontal (controlare puls)

12 -> Buzzer

13, 15 -> controlare LED RGB

14, 39 -> Masa circuitului

16, 18 -> Trig și Echo al senzorului de distanță

**NOTĂ**

Pinul 12 este singurul pin care permite o modulație de tip PWM de frecvență înaltă. Din acest motiv, este imperativ să i se asocieze buzzerul, altfel bubuie tranzistorul care face parte din schema de implementare a buzzer-ului. Acest lucru se datorează faptului că ceilalți pini nu pot menține un factor de umplere constant la fel de precis, introducând zgomot nedorit în tranzistor.

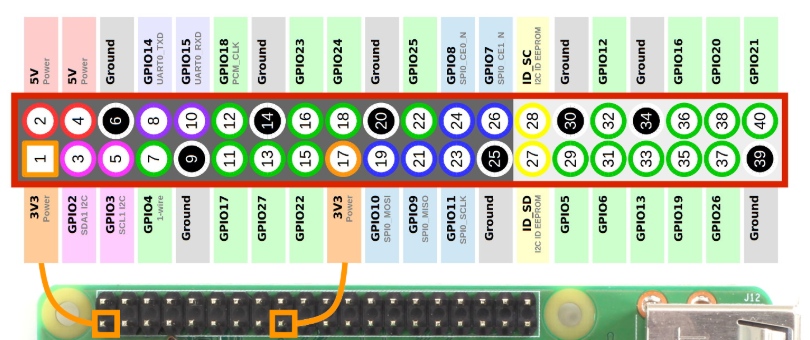


Figura 5

1. Explicarea codului Python

Întâi de toate, programul în care rulează codul este Python. În opinia mea, este relativ ușor de învățat pentru o funcționare de bază, precum aprinderea și stingerea unor LEDuri, dar devine mult mai complicat, de exemplu, pentru a încerca să derulăm un program care permite controlarea unui robot. De remarcat este că am început să programez în Python 2, dar apoi am ales să schimb abordarea și am realizat programul în Python 3. Cele 2 versiuni nu sunt complet compatibile una cu alta, și am considerat că, dacă se va realiza un update, prima abordare s-ar putea să nu mai funcționeze după un anumit timp.

Codul poate fi împărțit în mai multe etape. Inițial, se declară fie anumite funcții care vor fi folosite din anumite biblioteci (from ... import ...), fie bibliotecile întregi cu totul. (import ...). Ulterior, are loc partea de declarare a variabilelor locale sistemului principal. Exemple includ parametrii plăcii , adică rolul fiecărui pin folosit, relativ la biblioteca folosită și parametrii rezultați în urma interogării senzorului de distanță.

-> importarea bibliotecilor

-> declararea variabilelor legate de pinii plăcii

-> asignarea valorilor pinilor (input/output?)

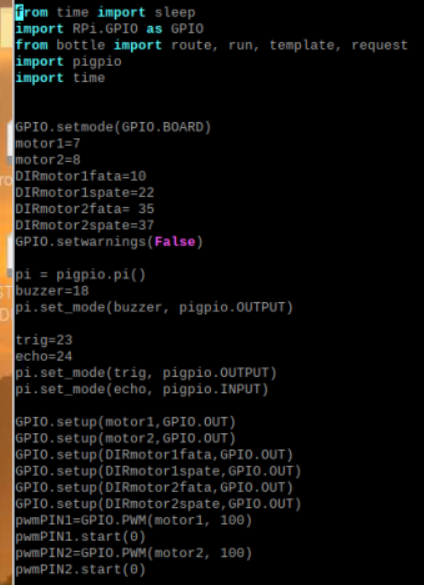
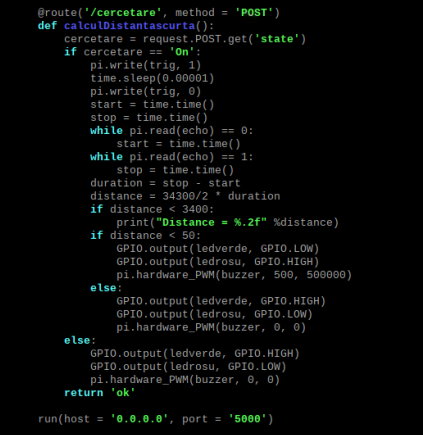


Figura 6 - Prima parte a codului

În a treia parte, se observă bucla în care se va găsi sistemul pentru mare parte din executare. Fiind partea centrală a programului, aici au loc toate testările de la intrare, din pagina web. Codul intră într-o buclă permanentă prin apelarea funcției run(host), așteptând imputuri prin pagina web astfel deschisă, tratată în capitolul următor. Este însă esențial ca în această buclă să se afle funcțiile care tratează diversele schimbări apărute pe pagina web, cu input de la utilizator. Pe scurt, prin html se găsesc aceste schimbări, iar programul în Python, odată ce i s-a pasat aceasta, intră într-o rutină prestabilită. Acestea comunică prin comanda @route(), care primește ca parametru rutina din html. Aici, se definește această funcție, care va fi apelată ori de câte ori i se ordonă din pagina web.



-> identificarea funcției ca fiind ‘/cercetare’ (se observă că e de tip POST, deci e un switch cu memorare de poziție -fiecare comutare postează rezultatul)

-> Body-ul funcției. În acest caz, ea se ocupă de senzorul de distanță și de interpretarea datelor de la el (se decide dacă să pornească avertizările sau nu)

-> run(host) - ne dă adresa ip a paginii web. 0.0.0.0 este aferentă adresei ip a RPI-ului în sine. Portul 5000 este un port nefolosit de altceva, de aceea l-am selectat pe el

Figura 7 - un exemplu de folosire al unei funcții în rutina de permanență

La acest nivel am conceput 3 tipuri de inputuri.

Comutatoare fără memorare de poziție (buton), care controlează deplasarea robotului și îi ordonă acestuia să claxoneze și să intre în modul cercetare. Acestea apelează rutinele doar cât timp este apăsat butonul, neavând niciun efect când el a fost eliberat.

Slider-ul funcționează în tandem cu comutatorul cu memorare de poziție, permițând robotului să scaneze rapid zona către care s-a îndreptat senzorul de distanță. Menționez aici aceste elemente, întrucât la acestea, care au memorare de poziție, este nevoie să facem o interogare continuă (metoda POST).

Ultima parte a codului constă în resetarea tuturor pinilor, stingerea LEDului și oprirea buzzerului, în caz că a mai rămas vreunul pornit.

De menționat este că fiecare dintre subrutine se regăsește și într-un program separat, care permite testarea fiecărei componente individuale în parte.

1. Explicarea funcționării fișierelui HTML aferent codului

Consider că unul dintre cele mai mari avantaje ale programului este cum cele 2 programe aferente conduc la interoperabilitate, și ajută orice utilizator în a-l folosi relativ ușor, fără a fi nevoit să înțeleagă întregul proces.

Programul scris în html are 2 părți, asemănător cu cel în Python.

În prima parte, este inițializat layout-ul paginii web. Noțiuni precum titlul, fontul, culori etc. pot fi definite aici. Multe din pictogramele interactive au fost luate de la cei de la fontawesome.com, care permit modificarea iconițelor și a dimensiunilor lor.

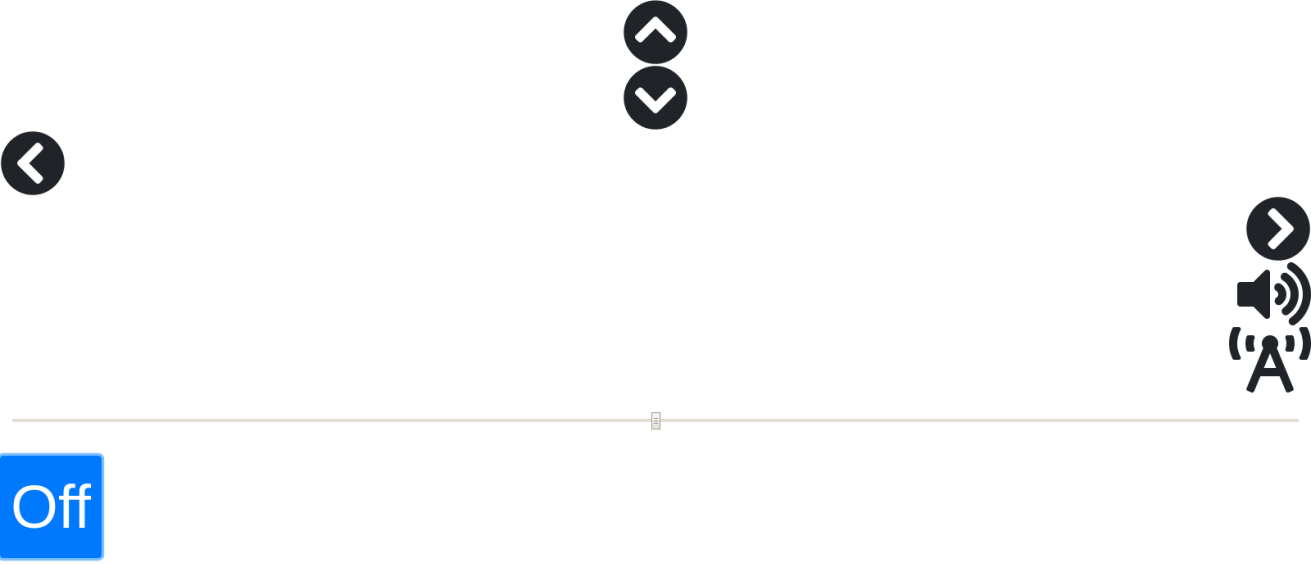
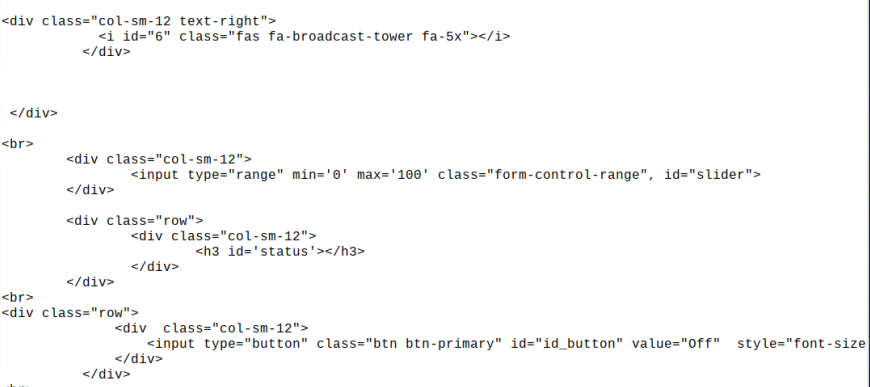


Figura 8 - Layout-ul paginii web



-> definirea butonului broadcast tower, cu id 6, mărit de 5 ori

-> definirea slider-ului

-> definirea butonului, cu valoarea implicită Off

Figura 9 - Declararea elementelor de pe pagina web

În a doua parte, sunt descrise funcționalitățile acestor inputuri, rutinele pe care le apelează și alte mici evenimente de acest gen. Aceasta este partea care se leagă propriu-zis de codul din Python.

-> Folosirea butonului #6

-> la mousedown (apăsare), se apelează funcția care a fost precizată la secția route(/sondare) în Python

-> la mouseup (eliberarea apăsării), se apelează rutina /stopsondare. Statusul se setează corespunzător

->

-> definirea butonului, cu valoarea implicită Off

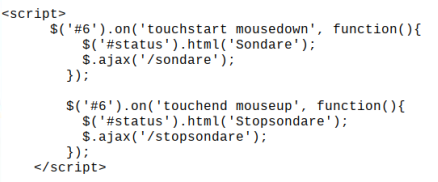


Figura 10 - Exemplu de buton în html

1. Funcționare. Sistemul Linux

Întâi de toate, programarea direct din terminal (bash), fiind dificilă și nefiind adecvată proiectului meu, am ales să lucrez cu limbajul Python, a cărui aplicație vine deja preinstalată. Lucrul în el se poate face prin două metode.

Prima, care este poate și mai interesantă, implică editarea direct din terminal a unui editor text (eu am folosit nano, de exemplu), și apoi rularea acestui fișier ca un program cu extenstia .py. Aceasta este o metodă preferată de mulți, întrucât nu este necesară o interfață grafică. A doua abordare este programarea direct în aplicația Python, ca și cum am scrie un cod în orice alt limbaj. Deși este evidentă nevoie de o interfață, acest lucru este compensat prin existența debuggerului, care permite vizualizarea variabilelor în timpul rulării programului, o unealtă indispensabilă pentru crearea unei aplicații cu suficient de multe necunoscute.

Sistemul de operare Linux permite o serie de funcționalități proiectului. Prima dintre ele ar fi în cazul în care plăcuța se folosește ca și calculatorul de bord al robotului, o putem seta ca să ruleze automat programul nostru la startup, prin salvarea comenzii de rulare in folderul bashrc, un folder ale cărui comenzi sunt mereu rulate la bootarea sistemului de calcul.

Poate cea mai importantă caracteristică a acestui sistem de operare este că el este gratis și are updateuri regulate, care sunt implemetate pentru vasta majoritate a distribuțiilor folosite astâzi, inclusiv Raspian, distrubuția folosită în cazul meu. Acesta este, evident, un mare avantaj, care reduce mult potențialele costuri de implementare ale acestui proiect.

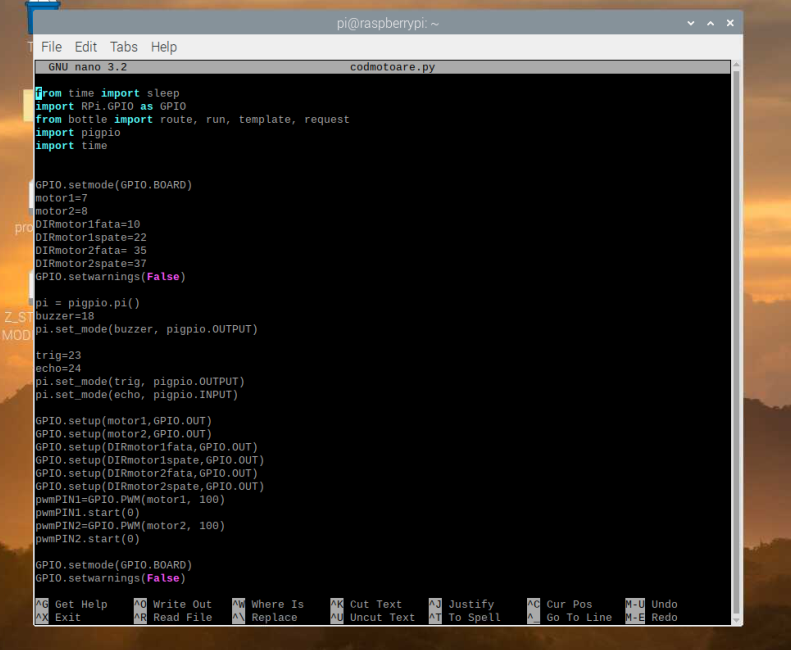


Figura 11 – Screenshot efectuat de pe sistemul de operare al plăcuței de test

Voi nota aici, de asemenea, cum m-am conectat la plăcuța de test. Aici se pot considera 3 variante, după cum urmează :

1. Conectarea unei tastaturi direct la plăcuța de test, nimic mai mult. Are avantuajul că sistemul rezultat este foarte mic în dimensiuni, însă, din cauza lipsei afișajului, poate fi o provocare pentru cei mai putin experimentați să lucreze astfel
2. Conectarea unei tastaturi, mouse și conectarea la un ecran. Astfel, plăcuța devine practic un sistem de calcul de sine stătător. Are dezavantajul că necesită un ecran, ceea ce nu toată lumea dispune de unul (de exemplu, studenții în cămin).
3. Conectarea, prin intermediul unei mașini virtuale, la un alt sistem de calcul (laptop). Această metodă este și cea preferată, întrucât nu avem nevoie de alte periferice pentru a o efectua. Dezavantajul este că necesită niște pași preliminari de conectare la aceeași rețea, de exemplu, la hotspot-ul făcut de un telefon mobil. Provocarea constă în setarea plăcuței pentru a se conecta automat la această rețea. Linkul va fi pus în siteografie.
4. Concluzii. Posibile îmbunătățiri

În opinia mea, proiectul conține în sine relativ multe inovații, pe care am fost forțat să le găsesc în mare parte singur, sau să le adaptez din anumite surse de inspirație, menționate în ultima parte. Probabil că un programator – mai specific unul care să cunoască html și Python mai bine – ar fi reușit să creeze un cod mai interesant sau cu mai multe funcționalități. O limitare observată asupra faptului că eu l-am scris este integrarea bibliotecilor în proiect. Sunt foarte multe biblioteci la liberă alegere, care, la prima vedere, conțin seturi de funcții cu aceeași aplicabilitate. Este evident că anumite dintre ele sunt recomandate în anumite situații, însă nu am încă cunoștințele să îmi spun părerea.

Din acest motiv, am început să compun programul cu ajutorul bibliotecii GPIO (General Purpose Input Output), însă i-am observat limitările și am început să folosesc biblioteca Pigpio. Cu toate astea, din cauză că nu am cunoștințele necesare, nu am renunțat pe deplin la prima bibliotecă.

Pe de altă parte, consider că, deși multora li se poate părea relativ inaccesibil, cu siguranță bazele Linux și Python se pot învăța repede. Bineînțeles, asta va asigura doar niște funcționări de bază, și dacă este necesar un algoritm mai complex, să spunem pentru un sistem care interconectează un document html cu un script în Python, atunci mai mult timp va trebui petrecut.

Multe din problemele apărute au fost din lipsa cunoștințelor de bază despre Linux. Chestiuni precum utilizatorul root, utilitarul sudo, editorul de text nano și multe altele au fost dificil de înțeles, cel puțin până le-am abordat la cursul dl Grivei, creditat în bibliografie.

O posibilă îmbunătățire ar fi legarea unui comutator la funcționarea întregului sistem. Cu alte cuvinte, pentru a evita anume posibile accidente, aș fi putut adăuga un comutator cu memorarea poziției la alimentarea întregii plăcuțe, și aș fi putut pune un buton suplimentat pe placuță, care să oprească rularea programului. Din păcate, nu am reușit să găsesc o soluție să le adaug pe placă, din cauza firelor dese care sunt puse pe deasupra, și din cauza existenței deja a unui număr semnificativ de componente.

Pe de altă parte, o idee de îmbunătățire a proiectului ar fi afișarea în timp real pe un cadran, folosindu-se coordonatele unghiulare (rază, fază) a distanței percepute de senzor, în mișcarea sa de 180 de grade. Este o idee la care lucrez, și va fi implementată într-un update ulterior.

După cum am menționat înainte, o mare îmbunătățire poate fi adusă prin simpla educare a programatorului(a mea) să lucreze mai eficient cu materialele pe care le are la îndemână. Nu am oferit cele mai elegante soluții la mai multe probleme apărute pe parcurs, soluții care cu siguranță ar fi fost găsite de un specialist în domeniu.

Ultima sugestie este cea legată de pinul 12 al plăcuței. Acest pin este de obicei folosit pentru generarea de semnale de frecvențe de ordinul kHz, însă nu poate fi folosit dacă se folosește, în cadrul aceluiași program, și un buzzer, și un servomotor la care este necesară o precizie ridicată. După ce am ars un tranzistor legat la buzzer, care generează alarma de obstacol în imediata vecinătate, a trebuit să renunț la ideea de a conecta servomotorul la acest pin. Drept urmare, l-am conectat la un pin obișnuit, ceea ce îi conferă o mică instabilitate. Funcționând pe același principiu, cu modulație PWM, servomotorul nu poate fi controlat la fel de precis de oricare alt pin, în afară de pinul 12, ceea ce duce la producerea unor mici zgomote de reajustare a motorului în funcționare. Sugestia mea ar fi aceea că ar fi ideal ca pinul 12 să nu fie unicul pin cu această capabilitate interesantă.

Ca o notă personală, de final, mi-a plăcut să asociez acest proiect cu cursul de microcontrolere al dl-ui colonel Popescu, întrucât se pot regăsi multe elemente comune, precum setarea pinilor de input output, generarea de diferite tipuri de semnale la ieșire pentru înaintemenționatul buzzer, operarea cu motoarele de curent continuu și altele.

Ultima concluzie trasă de mine este că acest tip de proiect este cel mai interesant, tip care îți permite să experimentezi cu orice noțiune, și să încerci constant să îmbunătățești constant un produs de o calitate redusă, prin inovație, descoperiri personale, idei date de colegi etc. Îi adresez mulțumiri îndrumătorului de proiect în acest sens.

1. Bibiografie/Siteografie

<https://www.instructables.com/id/How-to-Setup-Raspberry-Pi-Without-Monitor-and-Keyb/> Unul din ghidurile despre Raspberry Pi și conectarea facilă la el

<https://www.optimusdigital.ro/ro/> - Siteul de pe care am cumpărat mare parte din componente, inclusiv cele mecanice (roți, piulițe, șuruburi)

<https://www.raspberrypi.org> – Siteul celor de la Raspberry Pi, de pe care se și downloadează distribuția Linux de pe care am lucrat

<https://stackoverflow.com/questions/23013274/shutting-down-computer-linux-using-python> Acesta este doar un link al celor de la StackOverflow. Am folosit o sumedenie de informații de la ei, dar nu le-am salvat linkurile, așa că am să fac o referire generală la tot siteul. De aici am înțeles cam cum funcționează codul, și ulterior l-am adaptat nevoilor mele.

<https://www.amazon.de/-/en/Wireless-Programming-Electronic-Raspberry-Without/dp/B07W6N7YHB/ref=sr_1_42?dchild=1&keywords=roboter+chassis&qid=1606489152&sr=8-42>

Acesta este unul din linkurile care m-au ajutat mult în a înțelege o arhitectură de robot. Din poza aferentă linkului mi-a venit ideea să plasez un servomotor în fața robotului, și să scanez zona din fața lui.

Cursul dl cpt ing Grivei Alexandru de Sisteme de Operare – a inclus noțiuni fundamentale în ceea ce privește sistemul Linux, fiind, evident, de mare folos. O aplicație a fost, de exemplu, pornirea la startup a proiectului, fiind salvată comanda aferentă în fișierul .bashrc .

De asemenea, cursul dl-ui colonel Popescu a fost de mare folos în înțelegerea noțiunilor de bază de microcontrolere programabile.

Notă de final : țin să mulțumesc tuturor celor care au făcut acest proiect posibil. Aceștia sunt Ștefănescu George (baterie externă inițială, idee agrafe), Văduva George (set șuruburi), Gîra Gino (idee benzi elastice), Witowschi Eduard (cablu alimentare), Samoilă Alexandru (cablu alimentare baterie externă), Vîrtejaru Mihai (idee inițială alegere proiect).