

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAŞI  
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Robot auto-echilibrare**

propusă de

***Nemtoc Ciprian***

**Sesiunea:** iulie, 2018

Coordonator științific

***Lector Dr. Vârlan Cosmin-Nicolae***

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAŞI  
FACULTATEA DE INFORMATICĂ

# **Robot auto-echilibrare**

***Nemțoc Ciprian***

**Sesiunea:** iulie, 2018

Coordonator științific

***Lector Dr. Vârlan Cosmin-Nicolae***

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_

Data \_\_\_\_\_ Semnătura \_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemnatul(a) .....

domiciliul în .....

născut(ă) la data de ....., identificat prin CNP .....,  
absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de  
..... specializarea ....., promoția ....., declar pe propria răspundere, cunoscând consecințele falsului în  
declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr.  
1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul:  
\_\_\_\_\_

\_\_\_\_\_ elaborată sub îndrumarea dl. / d-na  
\_\_\_\_\_, pe care urmează să o susțină în fața  
comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin  
orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea  
conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări  
științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei  
lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere  
că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, .....

Semnătură student .....

## DECLARAȚIE DE CONSUMĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Robot auto-echilibrare*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însotesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, \_\_\_\_\_

Absolvent *Nemțoc Ciprian*

---

(semnătura în original)

## ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, Ciprian Nemțoc.

Încheierea acestui acord este necesară din următoarele motive:

*[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea produsului-program (personal, tehnologii, fonduri) și aportul adus de fiecare resursă.]*

Iași, \_\_\_\_\_

Decan *Adrian Iftene*

---

(semnatura în original)

Absolvent *Prenume Nume*

---

(semnatura în original)

## Cuprins

1	Introducere	7
2	Arhitectura hardware	10
2.1	Componente hardware	10
2.1.1	Microcontroler	12
2.1.2	Modul accelerometru cu giroscop	13
2.1.3	Punte H control motoare	14
2.1.4	Senzor infraroșu	16
2.1.5	Șasiu	17
2.1.6	Alimentare	18
3	Arhitectura software	19
3.1	Algoritmul PID	20
3.1.1	Valoarea P (proporțional)	22
3.1.2	Valoarea I (integrat)	22
3.1.3	Valoarea D (derivat)	23
3.1.4	Setarea valorilor PID	23
3.2	Inițializare	24
3.3	Bucla de actionare	25
4	Prezentarea aplicației	28
5	Concluzii	30
5.1	Contribuții personale	30
5.2	Direcții de viitor	30
6	Bibliografie	31
7	Tabelul figurilor	32

## 1 Introducere

Unul dintre cele mai importante aspecte din evoluția umană este folosirea uneltelor pentru a ne simplifica viața și efortul fizic depus. În cadrul uneltelor se încadrează și roboții care în ultimul timp au o desfacere tot mai mare în industrie. Prin construcția lor amplă și bine pusă la punct aceștia au ajuns chiar să preia în totalitate munca omului în anumite domenii.

Cea mai mare utilizare le este întrebuită în mediile în care nu avem acces sau sănătatea ne este pusă în pericol. În Fig. 1 se poate observa creșterea numărului de roboți din zona industrială în ultimii 6 ani. Această creștere accelerată este datorată revoluției informaticii care a determinat trecerea de la societatea industrializată la societatea avansată informatizată. (1)

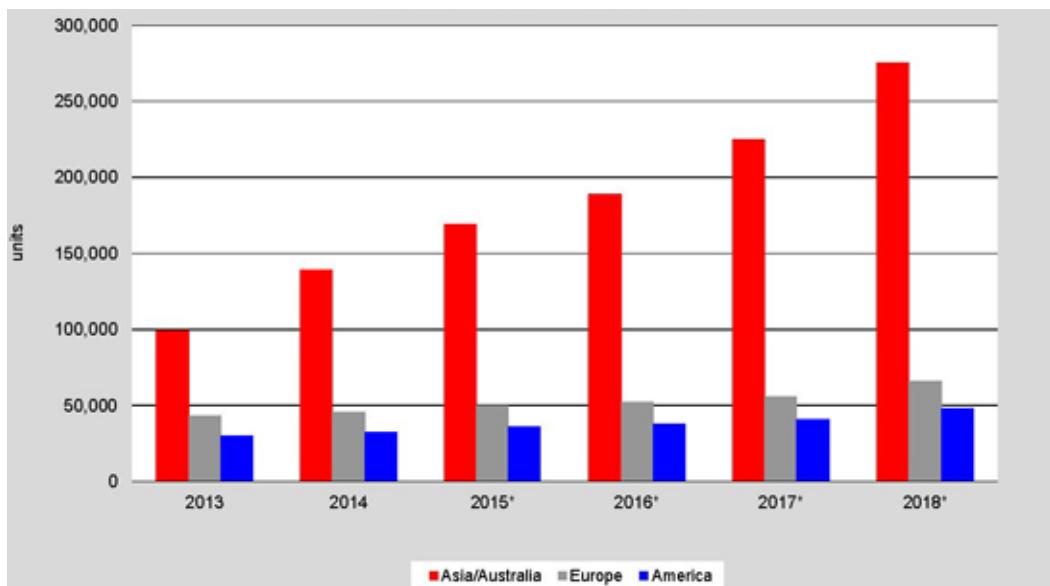


Fig. 1 Evoluția din industria robotică (2)

Lucrarea constă în construirea unui robot inteligent cu două roți, care să aibă capacitatea de a își menține echilibrul și de a urmări un traseu stabilit. În momentul actual există roboți care sunt capabili să își mențină echilibrul dar foarte puțini care să urmărească un traseu în același timp. Problema menținerii echilibrului trebuie rezolvată prin controlul motoarelor, iar problema urmăririi traseului trebuie rezolvată tot prin controlul motoarelor. Din această cauză apar conflicte deoarece procesul de menținere a echilibrului trebuie executat simultan cu urmărirea traseului. Principiul de bază în menținerea echilibrului este acționarea motoarelor în direcția de înclinare a robotului astfel unghiul de înclinare să fie îndreptat.

În Fig. 2 se poate observa principiul de bază al balansării prin controlul motoarelor în momentul în care unghiul de înclinare scade înspre partea din față. Procesul este identic și în sensul invers de înclinare a robotului.

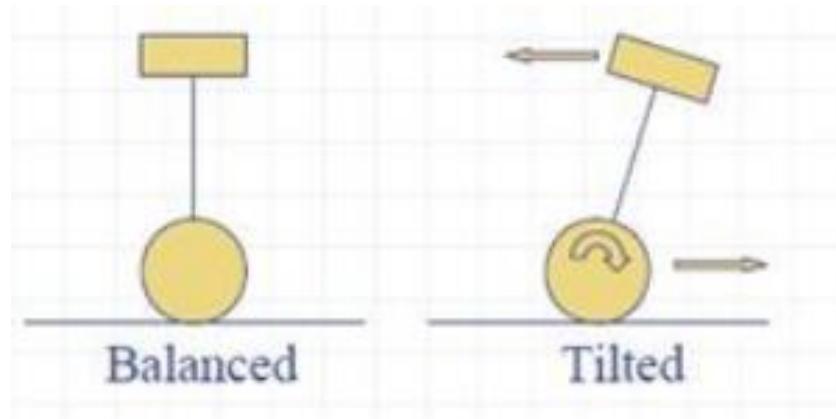


Fig. 2 Principiu de balansare robot

O aplicare practică a principiului de balansare care folosește o tehnologie similară este scuterul electric *hoverboard*. Acesta este compus din patru componente care lucrează împreună pentru a se asigura punerea în mișcare. Primul dintre acestea este senzorul amplasat pe fiecare dintre roți. Roțile în sine au motoarele electrice instalate împreună cu senzorii de viteză și înclinare. Cei doi lucrează cu giroscopul *hoverboard*-ului. Senzorii trimit informații despre viteza fiecărei roți la giroscop, giroscopul sortează informațiile trimise de pe roți și le transmite către placa logică principală.

Panoul logic servește drept creierul *hoverboard*-ului. În principiu, procesează toate informațiile care sunt transmise de la viteza *hoverboard*-ului la direcția și înclinarea fiecărei roți. Componenta finală în toate acestea este pachetul de baterii care ne asigură că avem suficientă putere pentru a avea o placă de bază funcțională. Procesul pornește de la motorul roților *hoverboard*-ului și se îndreaptă spre placa logică care controlează în principal viteza și mișcarea *hoverboard*-ului.

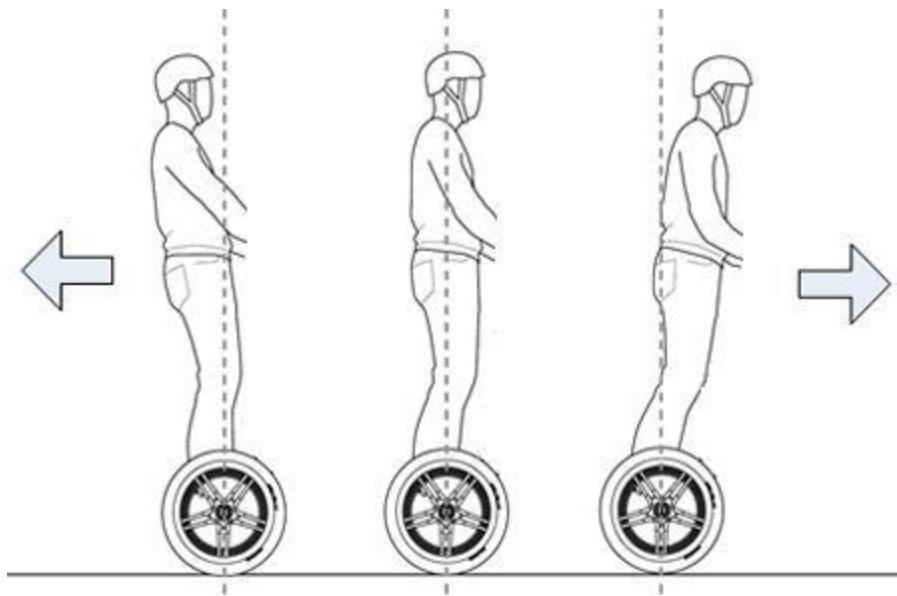


Fig. 3 Antrenarea motoarelor în momentul mișcării

Giroscopul ajută destul de mult la aspectul de echilibrare al *hoverboard*-ului. Cantitatea de presiune pusă pe tamponul de presiune determină cât de repede motorul se rotește. Pe măsură ce este pusă mai multă presiune asupra acesteia, giroscopul ajută la menținerea echilibrului și asigură faptul că centrul de greutate nu este afectat prea mult. Cu cât ne aplecăm mai mult, exercitând mai multă presiune asupra *hoverboard*-ului, cu atât avem mai multe șanse de a cădea și de a ne împinge înainte. Cu toate acestea, giroscopul funcționează pentru a ne asigura că nu facem acest lucru, deoarece funcționează pentru a echilibra placa corespunzător.

În capitolul 2 este prezentată arhitectura *hardware* a robotului, modul în care funcționează fiecare senzor folosit cât și modul în care aceștia comunică între ei.

Capitolul 3 pune accentul pe arhitectura *software* a robotului. Sunt prezentate concepte generale care au fost aplicate, algoritmi folosiți cât și principiul de funcționare.

Capitolul 4 descrie în detaliu toate funcționalitățile aplicației și problemele întâlnite în dezvoltarea acestora.

## 2 Arhitectura hardware

În cele ce urmează vor fi prezentate detaliile tehnice ale robotului, modul în care componentele comunică între ele. Pentru a putea îndeplini scopul final, arhitectura hardware și cea software au trebuit să fie foarte bine echilibrate. Nu putem avea un robot funcțional dacă avem doar una dintre componente foarte bine executate. Degeaba avem un cod eficient dacă construcția robotului nu este echilibrată și degeaba avem cele mai bune componente hardware dacă nu avem un cod care să ne ofere corect datele necesare într-un timp cât mai scurt.

### 2.1 Componente hardware

Pentru a realiza acest robot la nivel de hardware, ideal ar fi ca toate componentele să fie într-un ansamblu foarte bine balansat, cu o greutate și dimensiune medie. În Fig. 4 puteți observa construcția inițială a robotului, care nu a fost destul de optimă, fiind nevoie să construiesc altă versiune după cum puteți vedea în Fig. 5. Chiar și cu a doua versiune sunt sigur că ar mai fi loc de îmbunătățiri.

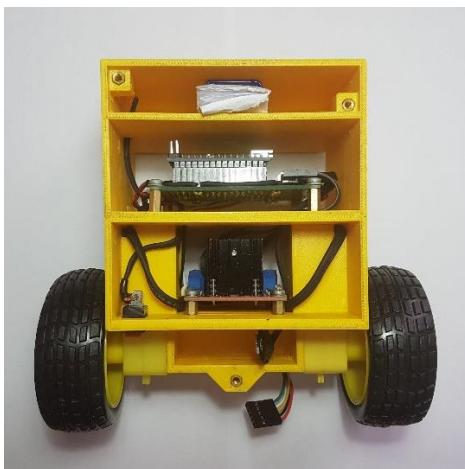


Fig. 4 Robot v1.0

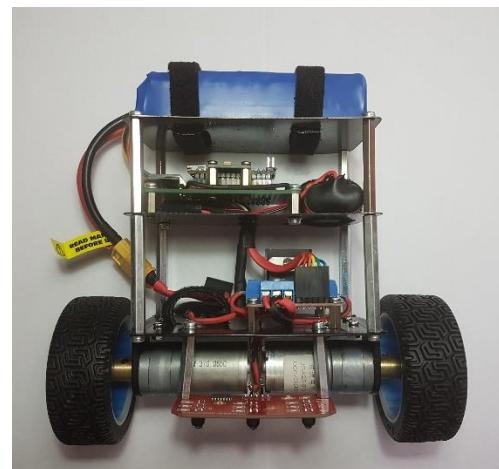
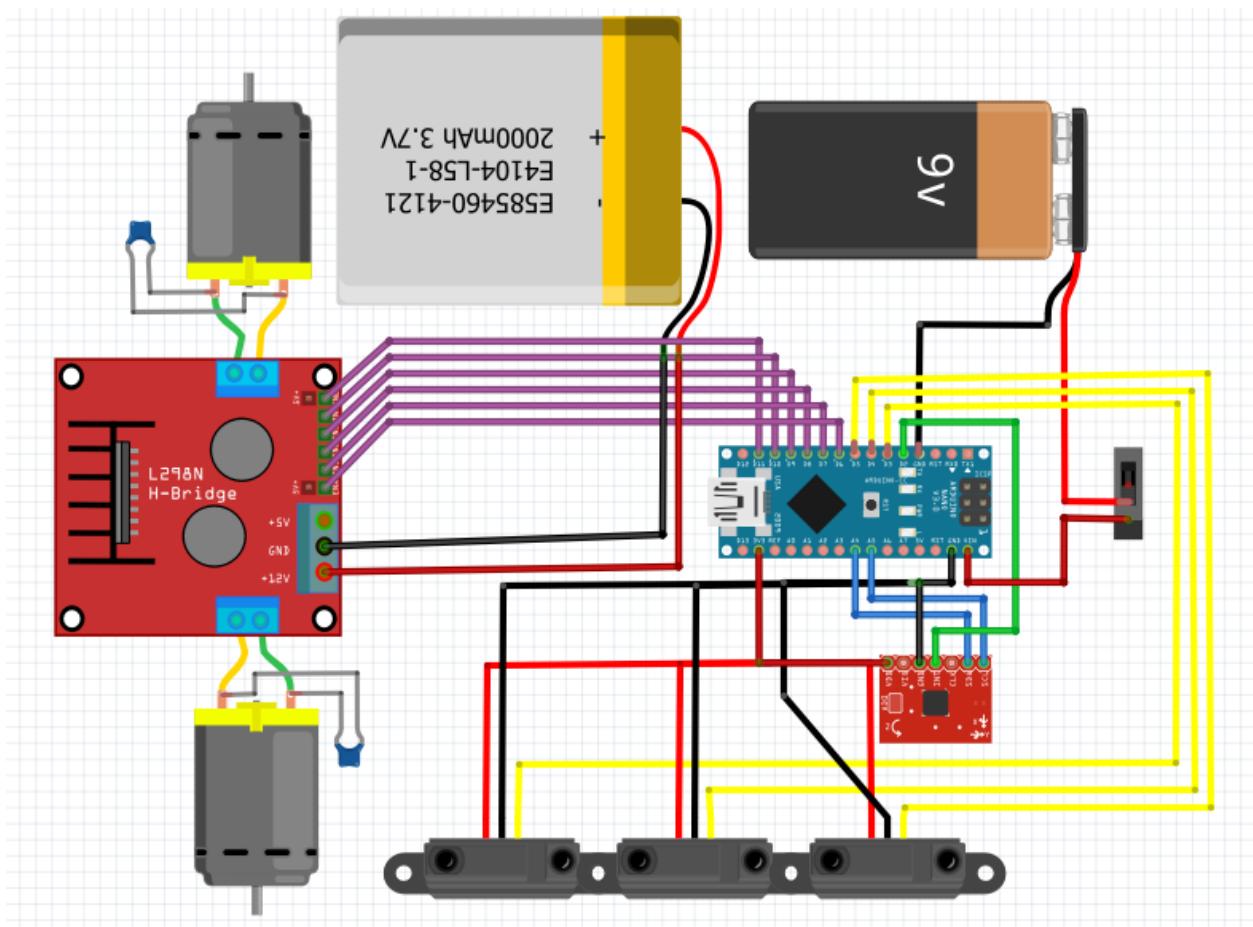


Fig. 5 Robot v2.0

În Fig. 6 se poate observa modul în care sunt conectate componente. Această schemă a fost realizată cu ajutorul unui soft de modelare a circuitelor dintre componente hardware, numit Fritzing.



*Fig. 6 Schema de conectare a componentelor*

În subcapitolele ce urmează este prezentată fiecare componentă hardware, și modul în care acestea comunică între ele.

### 2.1.1 Microcontroler



Fig. 7 Placă dezvoltare Arduino Nano

Arduino nano este o placă de dezvoltare bazată pe Atmega328 având 32 KB de memorie flash unde se poate stoca cod, din care 2 KB sunt folosiți pentru bootloader. Fiecare dintre cei 14 pini pot fi utilizati ca input sau output folosind funcțiile specifice, pinMode(), digitalWrite() sau digitalRead(). Unii dintre pini au funcții speciale. Dintre cei pe care i-am folosit sunt pinii 3, 5, 6, 9, 10, 11 care au funcția de PWM<sup>1</sup>, care ne ajuta să controlăm în cazul de față viteza de rotație a motoarelor, mai multe detalii în capitolul 2.1.3 de la pagina 14, și pinii 4(SDA), 5(SCL) folosiți pentru dispozitivul I2C în capitolul 2.1.2 de la pagina 13. Numărul de pini disponibil pe placă nu este mai mult decât suficient pentru a conecta senzorii de care avem nevoie.

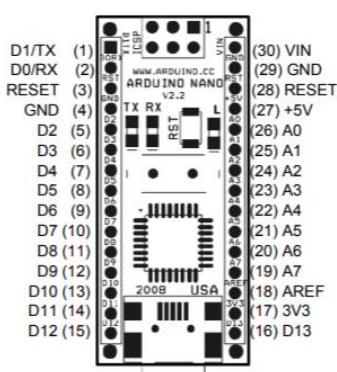


Fig. 8 Schemă pini Arduino Nano

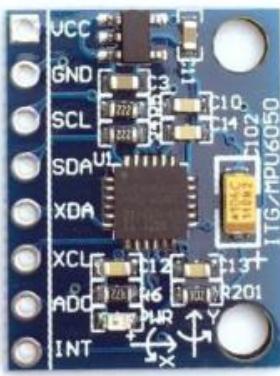
Pentru a realiza robotul am folosit o placă de dezvoltare Arduino Nano după cum puteți observa în Fig. 7. Fiind limitat de spațiu primul criteriu în alegerea controlerului a fost dimensiunea, acesta având o dimensiune de 4.3cm x 1.7cm. Odată cu dimensiunea mică a plăcii aceasta oferă și un consum redus de curent făcând astfel posibilă alimentarea.

Pentru robot, placă de dezvoltare Arduino Nano este centrul de comandă și control. Aici sosesc informațiile de la senzorii atașați prin intermediul pinilor digitali D2-D15 sau cei de tipul analog A0-A7 specificați în Fig. 8. Datele sunt prelucrate și în urma calculelor procesorul trimite semnale digitale către componente care trebuie să execute acțiuni.

Memoria oferită de Arduino Nano este suficientă, lăsând loc și de eventuale optimizări. Din cei 30Kb se folosesc 57%, aproximativ 17.7Kb, iar din memoria dinamică se folosesc 672 bytes pentru variabilele globale reprezentând 32% din aceasta.

<sup>1</sup> Pulse width modulation - Modularea lățimii prin pulsări

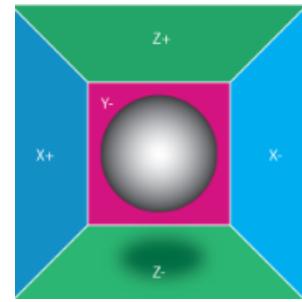
### 2.1.2 Modul accelerometru cu giroscop



*Fig. 9 Modul accelerometru cu giroscop*

Un accelerometru funcționează pe principiul efectului piezoelectric. Imagineați-vă o cutie cuboidală cu o mică minge în interiorul ei, ca în Fig. 10. Pereții acestei cutii sunt realizati cu cristale piezoelectrice. Ori de câte ori încărcați cutia, mingea este forțată să se deplaseze în direcția încinării datorită gravitației. Zidul pe care mingea se ciocnește creează mici curenți piezoelectrici. (3)

Pentru a putea balansa robotul avem nevoie de a determina în primul rând înclinația acestuia cât și viteza cu care unghiul de înclinație este modificat. În acest scop mă folosesc de un senzor MPU-6050, primul dispozitiv de urmărire a mișcărilor pe o axă x y z, construit astfel încât să consume un curent redus și să ofere o performanță optimă pentru consumatori. Acest senzor este folosit și pe unele dintre dispozitivele mobile.



*Fig. 10 Accelerometru*

Există trei perechi de pereți opuși într-un cuboid. Fiecare pereche corespunde unei axe în spațiul 3D: axe X, Y și Z. În funcție de curentul produs de pereții piezoelectrii, putem determina direcția de încinare și magnitudinea sa.

Modulul este conectat la Arduino după cum este arătat în Fig. 6 de la pagina 11, comunicând cu acesta prin protocolul I2C. Protocolul inter-integrat al circuitului (I2C) este un protocol destinat să permită comunicarea dintre un „slave” și un „master”, în cazul de față modulul MPU6050 fiind „slave”, iar Arduino Nano „master”. Comunicare se realizează prin intermediul pinilor SDA<sup>2</sup> respectiv SCL<sup>3</sup>. Biți sunt plasați pe linia de date SDA începând cu cel mai semnificativ bit, apoi linia de timp SCL este pulsată. Pentru fiecare 8 biți transferați, dispozitivul care recepționează datele trimite un bit de confirmare, deci există în realitate 9 impulsuri de ceas SCL pentru a transfera fiecare octet de date de 8 biți. Dacă bitul de confirmare trimis este scăzut atunci transferul a fost executat cu succes și este gata să accepte un alt octet. În caz contrar indică faptul că nu poate

<sup>2</sup> Linia de date

<sup>3</sup> Linia de timp

acceptă alte date. Viteza standard a liniei de timp SCL pentru protocolul de comunicare I2C este de 100KHz, care într-un regim de viteză ridicată poate ajunge până la 3.4MHz. În cazul de față linia de timp este la o viteză de 100KHz. Pentru o viteză mai ridicată ar trebui introduse întârzieri de timp după fiecare bit transferat, acest lucru fiind nefolositor.

Modulul MPU6050 este un dispozitiv de tipul I2C. Dispozitivele I2C sunt reprezentate printr-o adresă de 7 biți sau 10 biți, cea mai folosită fiind cea de 7 biți. La trimitera adresei de 7 biți, trimitem întotdeauna 8 biți. Bitul suplimentar fiind cel mai puțin semnificativ este folosit pentru a informa dispozitivul de tip "slave" dacă dispozitivul de tip "master" dorește să primească informații sau să trimită informații. Dacă bitul este zero, masterul scrie către slave. Dacă bitul este 1, masterul citește de la slave. Pentru a putea realiza comunicarea modulul MPU6050 are nevoie de un pin de intrerupere care ajută microprocesorul să știe că au fost introduse date noi în registrul care pot fi citite de către controler.

### 2.1.3 Punte H control motoare

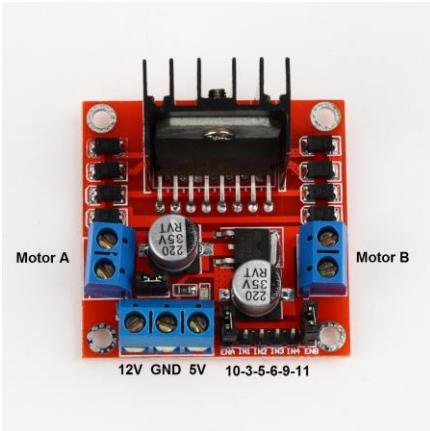


Fig. 11 Punte H

Puntea H utilizează driverul ST L298N cu punte dubla, un circuit monolitic integrat într-un pachet de 15 conduceri Multiwatt și PowerSO20. Acesta este un driver de înaltă tensiune, cu un curent dublu complet, conceput pentru a accepta niveluri logice standard TTL și pentru a actiona sarcini inductive, cum ar fi relee, solenoizi, motoare DC și motoare pas cu pas.

Sunt furnizate două intrări de activare pentru a activa sau a dezactiva dispozitivul independent de semnalele de intrare. Emițătorii tranzistorilor inferiori ai fiecărui pod sunt conectați împreună, iar terminalul extern corespunzător poate fi utilizat pentru conectarea unui rezistor de detecție extern. O sursă de alimentare suplimentară este furnizată astfel încât logica să funcționeze la o tensiune mai mică.

În cazul de față folosim puntea H, L298n prezentă în Fig. 11, pentru a controla separat viteza și direcția pentru fiecare motor. Puntea H este conectată la controler după cum puteți observa în schema de conectare a componentelor din Fig. 6 de la pagina 11. Pinii de control a motorului A

sunt ENA, IN1, IN2, iar pentru a controla motorul B ne folosim de pinii ENB, IN3, IN4. Pinii IN1, IN2, IN3 și IN4 sunt necesari pentru a stabili direcția motoarelor. Pentru a seta direcția motorului A pentru înainte, IN1 va avea valoarea 1, iar IN2 valoarea 0. În sens invers IN1 va avea valoarea 0, iar IN2 valoarea 1. Având direcția setată nu rămâne decât să mai definim viteza de rotație a motorului. Pentru a face acest lucru am conectat pinii ENA și ENB, de pe puntea H, la doi pini cu funcția de PWM de pe Arduino Nano. Modularea lățimii pulsului sau PWM este o tehnică pentru obținerea rezultatelor analogice cu mijloace digitale. Controlul digital este folosit pentru a crea un semnal între pornit și oprit. Acest model de pornire / oprire poate simula tensiunile între intrerupător (12 V) și oprit (0 V), prin schimbarea porțiunii din timpul petrecut pe durata semnalului față de momentul în care semnalul se stinge. Durata "de timp" se numește lățimea impulsului. Pentru a obține diferite valori analogice, trebuie să schimbăm această lățime a impulsului.

În Fig. 12 primul val are un timp de lucru care este de 10% din perioada totală de undă. În medie, energia conținută în impulsuri este de 10% din energia maximă disponibilă, linia roșie reprezentând media impulsurilor. De asemenea, al doilea și al treilea grafic arată 50%, respectiv 90% niveluri de energie. Un apel la analogWrite () se află pe o scară de la 0 la 255, astfel încât analogWrite (255) solicită un ciclu de funcționare 100% (întotdeauna activ), iar analogWrite (127) este un ciclu de funcționare de 50% exemplu.

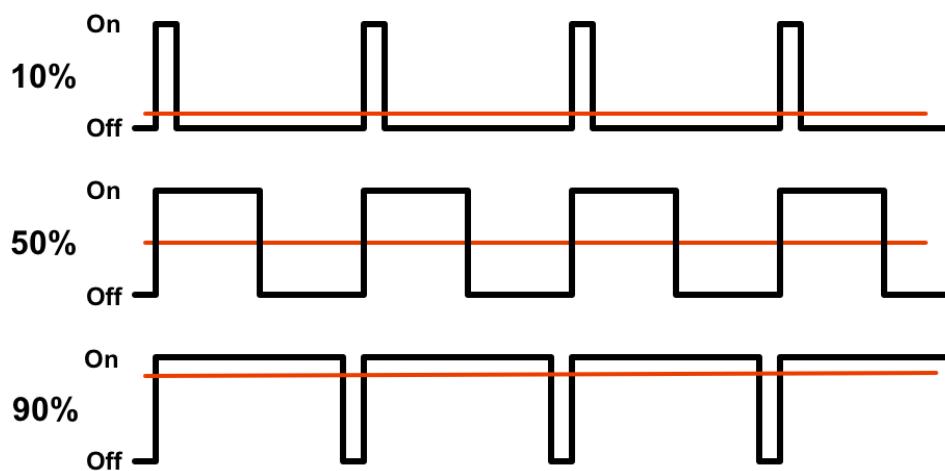


Fig. 12 PWM grafic



Fig. 13 Motor DC

Un motor DC este un mecanism electric care convertește energia electrică în energie mecanică. Motoarele folosite sunt atașate la puntea H după cum este notat în Fig. 11 de la pagina 14. Inițial am folosit pentru prima variantă două motoare DC cu angrenaj, care furnizează aproximativ 3000 de rotații pe minut. Din cauza angrenajului această putere este mult diminuată, ajungând astfel să fie ineficientă.

Pentru varianta finală a robotului am folosit motorul din Fig. 13. Funcționând la o tensiune de maxim 6V, în punctul de vârf acesta poate atinge o viteză de 280 rotații pe minut, transmise direct pe roata robotului, mărind astfel viteza de răspuns a mișcărilor.

#### 2.1.4 Senzor infraroșu

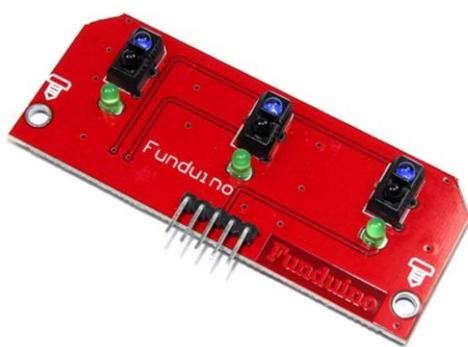


Fig. 14 Modul senzori infraroșu

Un senzor IR<sup>4</sup> este un instrument electronic care scană semnalele IR în anumite intervale de frecvențe definite de standarde și le transformă în semnale electrice prin pinul de ieșire, denumit în mod obișnuit pinul de semnal. Semnalele IR sunt utilizate în principal pentru transmiterea comenzi prin aer liber pe distanțe scurte, aproximativ de câțiva metri.

În cazul de față folosim senzorii infraroși din Fig. 14, pentru traseul pe care dorim să îl urmăram. Senzorii sunt conectați la controler după cum puteți observa în schema de conectare a componentelor din Fig. 6 de la pagina 11.

Fiind montat la o distanță mică față de suprafață pe care se deplasează, fascicolul IR emis de led loviște suprafață și se reflectă înapoi la fotodioda IR. Fotodioda oferă apoi o tensiune de ieșire proporțională cu reflecția suprafeței. Această placă oferă un semnal HIGH (5V) atunci când detectează o suprafață întunecată / neagră sau un semnal LOW (0V) atunci când detectează o suprafață albă. Ieșirea semnalului de la senzorul infraroșu este introdusă în placa principală care

---

<sup>4</sup> Infraroșu

are un comparator IC LM339 și face munca pentru a avea un semnal curat HIGH și LOW care mai apoi este transmis către controler prin intermediul pinilor digitali.

Pentru a stabili direcția de mers a robotului astfel încât acesta să urmărească linia, modulul prezentat în Fig. 14 dispune de o serie de trei senzori IR. Criteriul de bază este că linia neagră va avea o valoare de reflexie mai mică (negru absoarbe lumina) decât suprafața mai ușoară din jurul acestuia. Această valoare scăzută a reflecției este parametrul utilizat pentru a detecta poziția liniei de către robot. Valoarea mai mare a reflecției va fi suprafața din jurul liniei. Astfel, în această serie liniară de senzori IR, dacă senzorul IR de stânga / din dreapta prezintă valoarea scăzută a reflecției, atunci linia neagră este în mod corespunzător spre stânga / dreapta robotului. Controlerul compensează acest lucru semnalizând motorul să meargă în direcția opusă liniei.

### 2.1.5 Șasiu

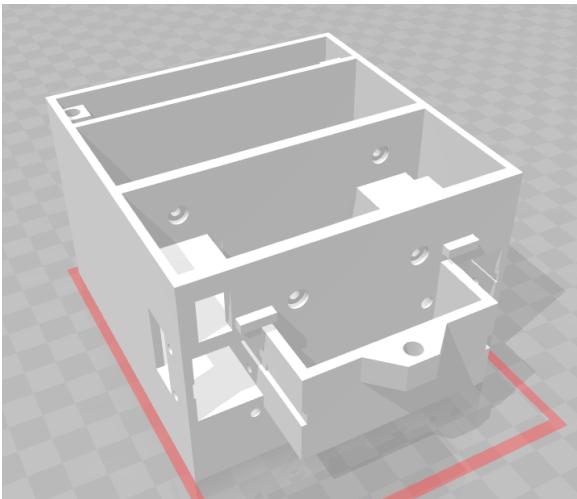


Fig. 15 Model 3D șasiu

Inițial pentru prima versiune a robotului am modelat 3D șasiul robotului, după cum se poate vedea în Fig. 15, respectând la milimetru toate dimensiunile componentelor și așezarea lor astfel încât construcția să fie cât mai bine balansată. Modelarea 3D a fost realizată într-un program de modelare 3D numit SketchUp care oferă și un mod de convertire a schiței pentru a facilita imprimarea.

Cu ajutorul profesorului coordonator am printat cu ABS<sup>5</sup> modelul la o imprimantă 3D la o temperatură de aproximativ 250 grade, tipărirea având o durată aproximativ de 19 ore. Din cauza timpului necesar pentru tipărire, șasiul robotului din varianta finală este realizată manual din diferite componente metalice după cum puteți vedea în Fig. 5 de la pagina 10. Fiind realizat manual acesta nu mai este atât de bine balansat în comparație cu modelul printat 3D cu o precizie la milimetru, dar are un avantaj mult mai ridicat la nivelul contragreutății.

<sup>5</sup> Acrilonitril butadien stiren (plastic)

### 2.1.6 Alimentare

Una dintre problemele majore în construcția robotului a fost alimentarea efectivă a acestuia. De-a lungul progresului am testat mai multe variante de alimentare. Pentru alimentarea controlerului și a senzorilor aferenți, o baterie de 9V este suficientă, problema apare la alimentarea motoarelor. Pentru a le alimenta am testat patru variante de alimentare, fiecare opțiune având nevoie de calibrări și ajustări diferite la nivel de soft. Încă de la început bateriile de 9V Li-on nu au avut puterea necesară pentru a facilita punerea în mișcare a motoarelor. O mică îmbunătățire a fost folosirea acumulatorilor de 9V Ni-mh dar nu pentru mult timp din pricina vitezei de descărcare la punerea în funcțiune a robotului. Opțiunea ideală, dar nu și cea optimă, a fost alimentarea printr-un încărcător de 9V conectat printr-un cablu. După o documentare mai profundă, pentru varianta finală am folosit un acumulator Li-Po de 11.1V, plasat în partea superioară a șasiului după cum se poate observa în Fig. 5 de la pagina 10, greutatea acestuia facilitând totodată procesul de stabilizare a robotului.

### 3 Arhitectura software

Pentru a programa placa Arduino Nano sau oricare alt controler trebuie să folosim un compilator care are capacitatea de a produce un cod mașină binar. Pentru a fi programat, Arduino oferă un mediu integrat de dezvoltare (IDE). Limbajul de programare suportat de Arduino este C sau C++, iar un program scris în IDE pentru Arduino este numit sketch. Un sketch poate conține mai multe tipuri de fișiere (ino, h, c, cpp). Pre-procesarea se face doar asupra fișierelor care au extensia .ino. În momentul pre-procesării toate fișierele .ino din sketch sunt concatenate, în ordine alfabetică, într-un fișier cu extensia .cpp. Înainte de a compila fiecarul.cpp, se face o încercare de a reutiliza fișierul .o compilat anterior, ceea ce accelerează procesul de construire. Un fișier .d (dependență) special oferă o listă a tuturor celorlalte fișiere incluse de sursă. Pasul de compilare este ignorat dacă fișierele .o și .d există și au timestamp-uri mai noi decât sursa și toate fișierele dependente.

Un fișier cu extensia .ino, care conține programul de bază trebuie să conțină cel puțin două funcții specifice. Funcția setup() care este rulată o singură dată, având rolul de a inițializa setările, și o funcție loop() care este apelată constant din momentul în care placa este în funcțiune. Mai jos se poate observa un exemplu unde ledul cu numărul 13 este inițializat în funcția setup(), iar funcția loop() acționează ledul în mod repetat.

```
void setup() {
    pinMode(13, OUTPUT);
}
void loop() {
    digitalWrite(13, HIGH);
    delay(100);
    digitalWrite(13, LOW);
    delay(100);
}
```

În diagrama din Fig. 16 de la pagina 20 este prezentată metoda de funcționare a programului pentru balansarea robotului cât și pentru urmărirea liniei. Senzorul MPU6050 cât și modulul IR trimit informații despre starea acestora către microprocesor, care la rândul lui procesează informațiile

primite. Datele calculate sunt trimise apoi mai departe către puntea H care le distribuie către motoarele specifice.

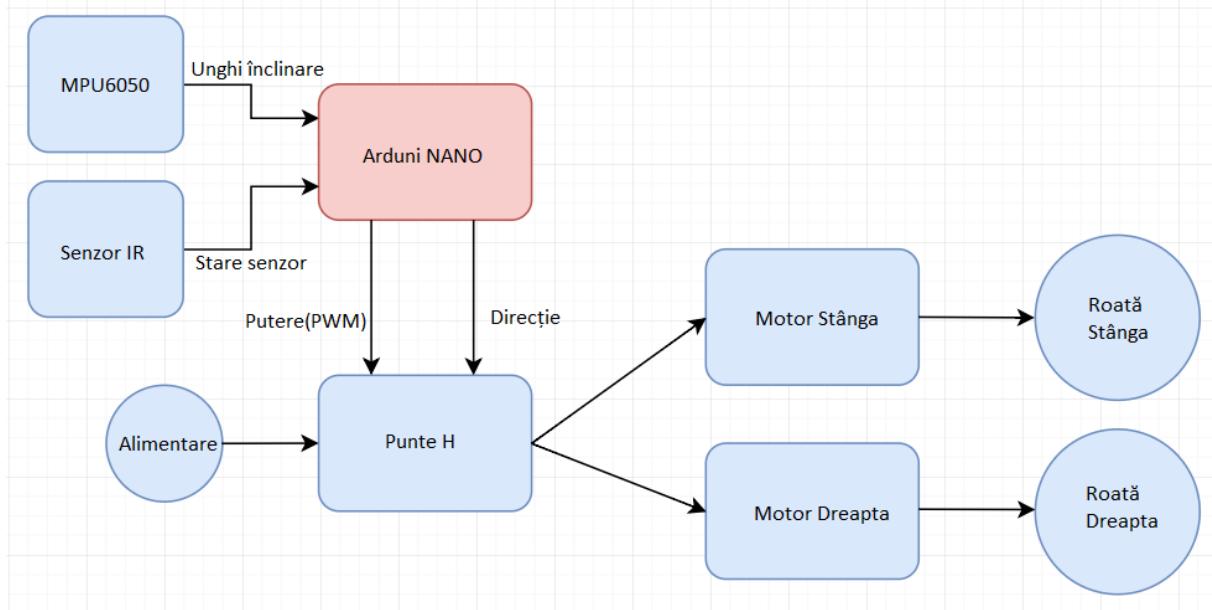


Fig. 16 Diagrama codului

### 3.1 Algoritmul PID

Pentru a menține echilibrul robotului am utilizat algoritmul PID. Controlul proporțional-integrat-derivat (PID) este unul dintre cei mai des întâlniți algoritmi de control, utilizați în industrie și este universal acceptat în controlul industrial. Popularitatea controlorilor PID poate fi atribuită parțial performanței lor robuste într-o gamă largă de condiții de funcționare și parțial simplificării lor funcționale, ceea ce le permite inginerilor să le opereze într-o manieră simplă și directă. După cum sugerează și numele, algoritmul PID are la bază trei coeficienți de bază, proporțional, integral și derivat care sunt setate pentru a obține un răspuns eficient. Ideea de bază din spatele unui controler PID este de a citi un senzor, în cazul de față senzorul care ne indică unghiul de înclinare a șasiului, apoi de a calcula ieșirea actuatorului dorit prin calcularea răspunsurilor proporționale, integrale și derive. Pentru a calcula valoarea de ieșire aceste valori sunt însumate.

Într-un sistem de control, variabila procesului este parametrul de sistem care trebuie controlat, unghiul de înclinare al robotului. Senzorul MPU6050 este utilizat pentru a măsura variabila de proces și de a oferi feedback sistemului de control. Punctul de setare este valoarea dorită, înclinație de 0%. În momentul în care avem o diferență dintre variabila de proces și valoarea setată, aceasta

este folosită de algoritm pentru a determina acțiunile necesare astfel încât să compenseze diferența de valoare.

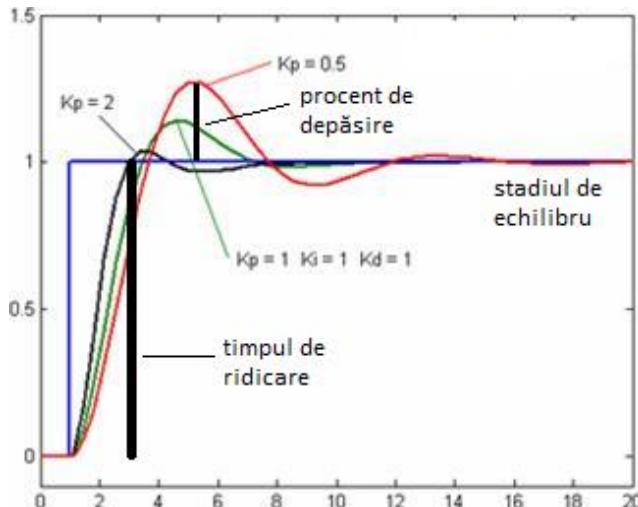


Fig. 17 Grafic valori PID

se regleze într-o marjă de eroare cât mai mică. Marja de eroare la starea de echilibru este diferența dintre valoarea setată și variabila de proces.

După utilizarea uneia sau a tuturor acestor cantități pentru a defini cerințele de performanță pentru un sistem de control, este util să se definească condițiile cele mai nefavorabile în care sistemul de control este de așteptat să îndeplinească aceste cerințe de proiectare. Adesea, există o perturbare a sistemului care afectează variabila procesului sau măsurarea variabilei de proces. Este important să se elaboreze un sistem de control care să funcționeze în mod satisfăcător în condițiile cele mai nefavorabile. Capabilitatea sistemului de control să depășească efectele tulburărilor este denumită respingerea perturbării sistemului de control.

În unele cazuri, răspunsul sistemului la o anumită ieșire de control se poate schimba în timp sau în legătură cu o anumită variabilă. Un sistem ne liniar este un sistem în care parametrii de control care produc un răspuns dorit la un punct de operare ar putea să nu producă un răspuns satisfăcător la un alt punct de operare. De exemplu, amplasarea senzorului MPU6050 care ne oferă date despre înclinarea șasiului va avea un răspuns mai rapid dacă acesta este amplasat cât mai sus pe șasiu decât atunci când este amplasat la baza acestuia. Capabilitatea sistemului de control de a tolera perturbațiile și neliniaritățile este denumită robustețea sistemului de control.

În Fig. 17 am reprezentat într-un grafic valorile oferite de algoritm PID în procesul de stabilizare a robotului. Timpul de ridicare reprezintă timpul necesar pentru a trece de la 10% la 90% din valoarea de echilibru. Procentajul de depășire este diferența cu care variabila de proces depășește valoarea finală de echilibru. Timpul de stabilire este timpul necesar astfel încât variabila procesului să

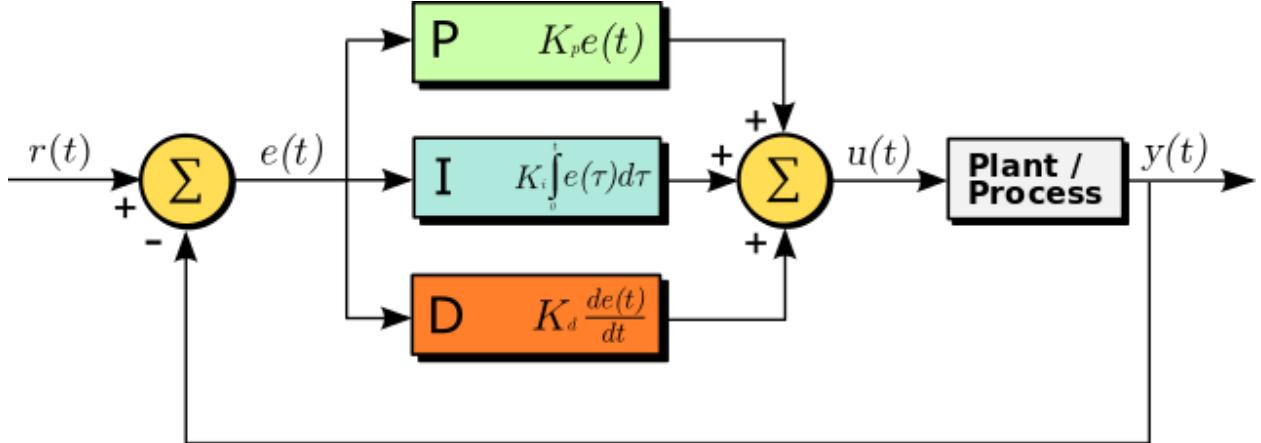


Fig. 18 Schema de calcul PID (4)

### 3.1.1 Valoarea P (proporțională)

Valoarea proporțională ia în considerare eroarea prezentă și face un efort proporțional cu măsura în care eroarea este de la punctul de referință. Cu toate acestea, pe măsură ce se apropiie de valoarea de echilibru, eroarea devine prea mică pentru a avea un efect asupra controlerului, prin urmare, va exista întotdeauna o eroare la starea de echilibru, care apare ca o deviere de la valoarea de referință.

Cu toate acestea, o valoare mare a termenului proporțional poate determina sistemul să atingă valoarea de echilibru, dar va face sistemul mai instabil cu oscilații și depășiri, o valoare mică a lui P va face robotul să cadă. Astfel se comportă mai mult ca un controler pornit sau oprit. Aceasta este motivul pentru care este nevoie de obicei și de setarea valorii I (integrat).

### 3.1.2 Valoarea I (integrat)

Valoarea I încearcă să echilibreze diferența de timp petrecută de valoarea proporțională pe ambele părți ale valorii de echilibru. De exemplu, dacă controlerul petrece ceva timp la 95% din valoarea setată, termenul I va împinge controlerul la 105% pentru aceeași perioadă de timp. Astfel, există o depășire în sistem. Aceasta va fi o problemă pentru sistemele sensibile, dar controlerul PI este mai bun datorită simplității sale. Controlerul PID nu poate depăși anumite limite, prin urmare limitele superioare și inferioare ale termenilor integrați trebuie să fie setați în program, astfel încât dincolo de aceste limite bucla PID să se satureze la valorile corespunzătoare. În robotul nostru de echilibrare, valoarea limită pentru controlul PWM al motorului este de la 0 la 255, prin urmare, ieșirea PID este limitată în limitele de la -255 la 255,

valorile negative reprezentând direcția opusă a motoarelor. O valoare bună a lui I va face robotul să se stabilizeze într-un timp cât mai scurt.

### 3.1.3 Valoarea D (derivat)

Valoarea D ia derivatul erorii sistemului în fiecare punct. Astfel, încearcă să prezică stadiul următor al robotului. Scopul acestui termen este de a verifica modul în care se mișcă variabila de proces fără a depăși valoarea de echilibru. Valoarea D acționează pe termenii PI prin contracararea acestora. În cazul robotului nostru de echilibrare, eroarea este unghiul și derivatul acestuia va da viteză unghiulară. Dar avem un senzor de giroscop care ne dă direct viteza unghiulară. Senzorul giroscop răspunde mai repede decât se poate calcula buclele PID. Prin urmare, putem trece peste calculul termenului derivat și îl putem înlocui cu rata giroscopului în jurul axei X. O valoare bună a valorii D va face robotul să stea în picioare chiar dacă acesta este împins.

### 3.1.4 Setarea valorilor PID

Procesul de stabilire a valorilor optime pentru P, I și D în scopul de a obține un răspuns ideal de la un sistem de control se numește tuning. Valorile unui controler PID pot fi obținute prin metoda de încercare și eroare. Odată ce sunt înțelese semnificațiile fiecărui parametru PID, această metodă devine relativ ușoară. În această metodă, termenii I și D sunt setați inițial la zero, iar valoarea lui P este mărit până la ieșirea buclei. Pe măsură ce crește valoarea lui P, robotul devine mai rapid, dar trebuie să fie luate măsuri care să nu facă robotul instabil. Odată ce valoarea lui P a fost setată pentru a obține un răspunsul rapid dorit, valoarea lui I este mărită pentru a opri oscilațiile. Valoarea lui I reduce starea de eroare la starea de echilibru, dar crește și depășirea. O anumită cantitate de depășire este întotdeauna necesară pentru un sistem rapid, astfel încât să poată răspunde imediat la schimbări. Valoarea lui I este optimizată pentru a obține o eroare minimă la starea de echilibru. Odată ce parametrii P și I au fost stabiliți pentru a obține sistemul de control rapid dorit cu o eroare minimă la starea de echilibru, valoarea lui D este mărită până când procesul de stabilizare este acceptabil de rapid. Creșterea valoiei D scade depășirea și obține un câștig mai mare cu stabilitate, dar ar determina sistemul să fie extrem de sensibil la zgromot. (5)

### 3.2 Inițializare

Pentru a putea accesa datele oferite de toți senzorii folosiți și pentru a trimite comenzi de acționare, toate componentele trebuie inițializate. Acest lucru se face prin metoda setup() care se execută o singură dată în momentul în care controlerul este pornit sau resetat. În procesul de inițializare setăm în primul rând conexiunile dintre module și micropresor. Fiecare pin primește o atribuție specifică necesității senzorului.

```
// definirea modulului cu senzori IR
pinMode(LEFT, INPUT);
pinMode(CENTER, INPUT);
pinMode(RIGHT, INPUT);

// definire senzor MPU6050 & inițializare
MPU6050 mpu;
mpu.initialize();
mpu.setXGyroOffset(118);
mpu.setYGyroOffset(280);
mpu.setZGyroOffset(13);
mpu.setZAccelOffset(532);

// inițializare PID
pid.SetMode(AUTOMATIC);
pid.SetSampleTime(10);
pid.SetOutputLimits(-255, 255);
```

Valorile de inițializare a senzorului MPU6050 sunt calculate cu ajutorul unui script de calibrare. Pentru a obține valorile optime, șasiul robotului trebuie așezat în poziția de echilibru iar apoi scriptul Arduino este rulat timp de 180 de secunde fără a fi mișcat din poziția de echilibru. Valorile astfel obținute ar trebui să reflecte limitele de echilibru ale șasiului în conformitate cu poziția senzorului giroscop.

Tot în procesul de inițializare se verifică dacă a fost efectuată cu succes stabilirea conexiunii dintre componente. În cazul în care este detectată o eroare în procesul de inițializare, robotul nu va face nici o acțiune în buclă. De cele mai multe ori eroarea poate apărea din pricina încărcării în memorie, moment în care placăa trebuie resetată.

### 3.3 Bucla de acționare

Instrucțiunile din funcția `loop()` sunt cele care oferă comenzi în principiu către componente, pentru a determina robotul să își mențină echilibrul și să urmărească traseul stabilit. Inițial se face o verificare a informațiilor primite de la senzorul MPU6050 iar dacă acesta indică o poziție orizontală a șasiului, robotul nu execută nici o acțiune, acesta nefiind capabil să se ridice singur în picioare. Această verificare ajută și în momentul în care robotul cade, oprind astfel acționarea motoarelor.

Datele primite de la senzorul giroscop sunt stocate într-o listă FIFO. Pentru a măsura unghiul de înclinare al robotului avem nevoie de valori de acceleratie de-a lungul axelor y și z. Funcția `mpu.getAccelerationZ()` va prelua datele de accelerare din registrele corespunzătoare, iar funcția `atan2(y, z)` dă unghiul în radiani între axa z pozitivă a unui plan și punctul dat de coordonatele (z, y) pe acel plan. Semnul este pozitiv pentru unghiurile în sens invers acelor de ceasornic și un semn negativ pentru unghiurile acelor de ceasornic. Pentru a echilibra robotul, nu avem nevoie doar de unghiul în care este robotul, ci trebuie să știm și rata la care acesta se încadrează. Giroscopul în 3 axe al lui MPU6050 măsoară viteza de rotație de-a lungul celor trei axe. Pentru a face acest lucru trebuie să cunoaștem timpul de buclă. Prin urmare, folosim `millis()` pentru a ne spune timpul necesar de la începutul programului. Timpul de buclă va fi, `currentTime - previousTime`, apoi vom folosi `mpu.getRotationX()` pentru a obține viteza unghiulară în jurul axei X. Acum, giroscopul nostru deține datele într-un registru de 16 biți și vom folosi `map()` pentru a le mapa într-un interval cuprins între -255 și 255.

În momentul în care senzorul giroscop este întrerupt, se calculează valoarea optimă de acționare a motoarelor cât și direcția acestora cu ajutorul algoritmului PID, pentru a stabiliza robotul. În procesul de stabilizare sunt verificate și senzorii de direcționare pentru a respecta traseul stabilit. Dacă robotul este într-un stadiu de echilibru, liniștit, atunci sunt acționate motoarele pentru deplasarea înainte respectând traseul stabilit până când factorul de echilibru este destabilizat. Factorul de echilibru, stabilitatea a robotului este stabilit cu ajutorul unei liste FIFO în care sunt introduse ultimele valori primite de la senzorul giroscop. Dacă aceste valori se află într-un interval cât mai apropiat atunci putem aprecia valoarea scăzută de stres a robotului.

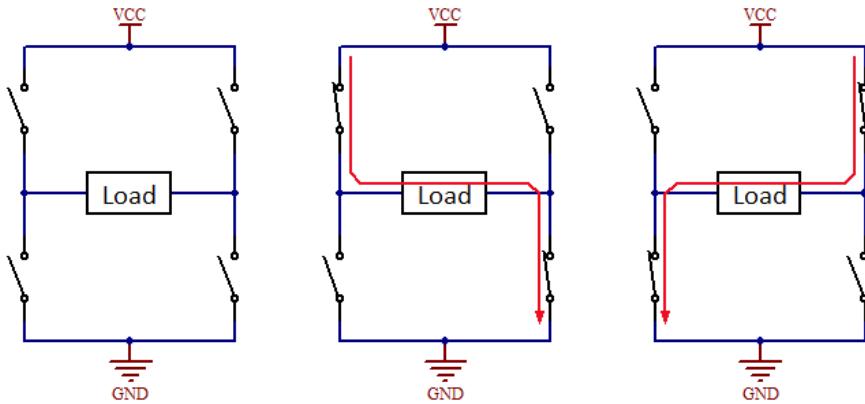
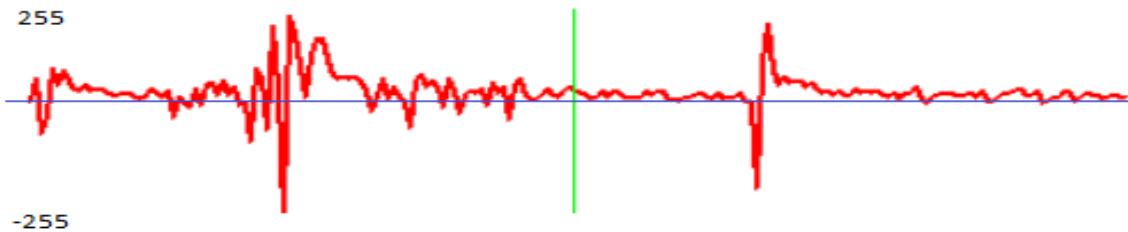


Fig. 19 Acționare punte H

Punerea în mișcare propriu zisă a robotului se face prin intermediul motoarelor, a căror direcție și sens sunt definite de puntea H la nivel logic. După cum se poate observa în Fig. 19, prima schiță reprezintă motorul în stare de standby. Direcția de rotație a motoarelor este schimbată prin definirea polarității. A doua reprezentare este specifică pentru rotația înainte a motorului, iar ultima este specifică pentru rotația înapoi.

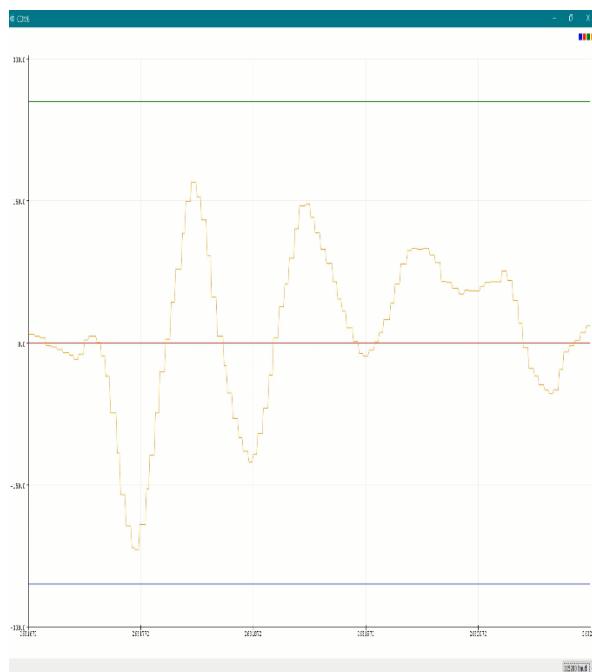
```
// Reprezentarea 2 din Fig. 19
analogWrite(ENA, 200); // viteza motor (PWM)
digitalWrite(IN1, LOW); // rotație înainte
digitalWrite(IN2, HIGH);
delay(1000);
// Reprezentarea 3 din Fig. 19
analogWrite(ENA, 200)
digitalWrite(IN1, HIGH); // rotație înapoi
digitalWrite(IN2, LOW);
delay(1000);
```

Dacă reprezentăm datele care ne sunt oferite de către algoritmul PID, într-un grafic corespunzător, în funcție de factorul de putere transmis motoarelor și timpul în care această putere este dispersată putem calcula și adapta mai ușor valorile PID în funcție de nevoile necesare. Un exemplu de oscilare a valorii de antrenare a motoarelor se poate observa în Fig. 20.

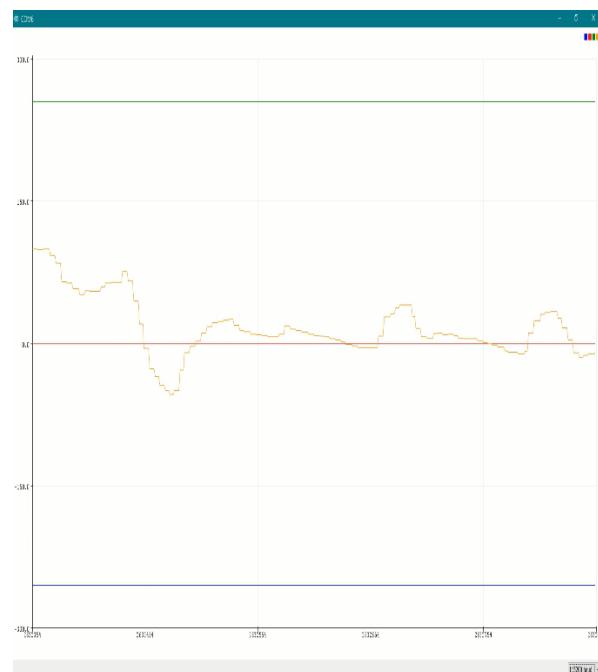


*Fig. 20 Reprezentare date de ieșire PID*

Arduino IDE oferă posibilitatea de a reprezenta grafic variația valorilor, prin terminalul “Serial plotter”. În reprezentările de mai jos este expusă valoarea întoarsă de algoritmul PID. Linia roșie prezintă la indicatorul zero, reprezintă punctul ideal de balans. În Fig. 21 este reprezentată variația valorii în momentul în care robotul este destabilizat și încearcă să atingă punctul de echilibru. Valorile prezintă o diferență substanțială fapt pentru care robotul reușește stabilizarea. Fig. 22, continuarea graficului din Fig. 21, reprezintă starea de echilibru și menținerea acesteia.



*Fig. 21 Grafic în procesul de stabilizare*



*Fig. 22 Grafic în momentul stabilizat*

## 4 Prezentarea aplicației

Această lucrare are ca obiectiv crearea unui robot cu doar două roți care să aibă capacitatea de a își menține singur echilibrul și de a parurge un traseu stabilit. Atingerea acestui scop este posibil prin îmbinarea noțiunilor teoretice cât și practice prezentate în capitolele 2 și 3.

Calitățile de bază a robotului sunt de a își menține echilibrul și de a urma traseul stabilit. În stadiul de menținere a echilibrului robotul își menține poziția verticală chiar dacă intervin factori perturbatori, precum împingerea intenționată a acestuia. Robotul este capabil de a se adapta pentru orice tip de taseu, acest lucru fiind posibil cu ajutorul senzorilor infraroși.

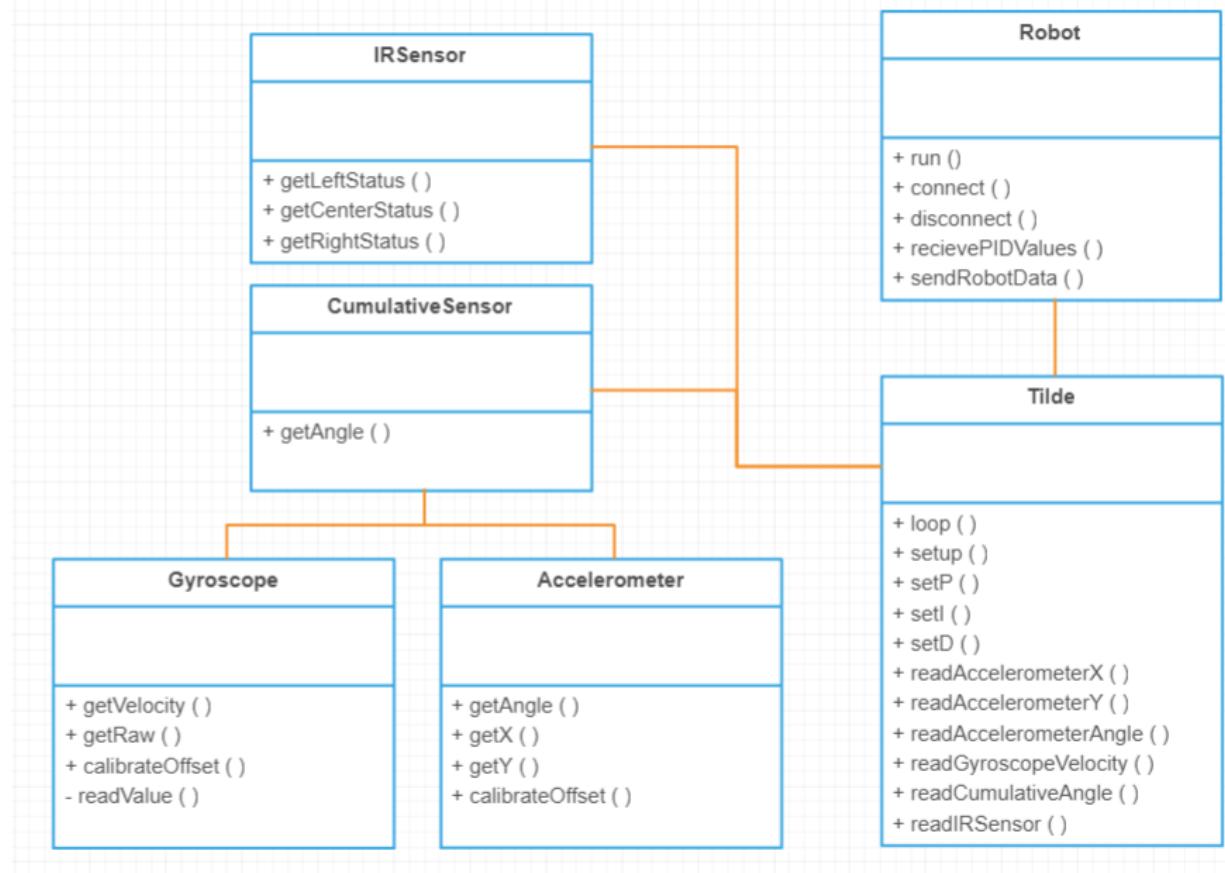


Fig. 23 Diagrama aplicației

În decursul realizării lucrării au fost întâmpinate mai multe probleme majore. Una dintre cele mai mari a fost diferența dintre componentele fizice care au trebuit compensate la nivel de soft. Din cauza unei erori la nivel de soft componente din prima versiune a robotului au devenit nefuncționale. În momentul înlocuirii, acestea au avut nevoie de noi teste, calibrări și abordări

diferite. O altă dificultate a fost în ansamblu de putere necesară cerută de soft, motoare și alimentare. Problema a fost rezolvată prin optimizarea fiecăreia din aceste componente. Astfel a fost stabilit un echilibru între aceste componente, motoarele ne fiind solicitate decât în momentele cheie pentru procesul de stabilizare a robotului.

Un aspect important de precizat la nivelul construcției în procesul de balansare este contragreutatea. Dacă inițial am încercat să construiesc un robot mai ușor la vâr, acest lucru ne fiind favorabil pentru algoritmul PID, în versiunea a doua am folosit acumulatorul ca contragreutate. În Fig. 24 se poate observa robotul final în starea de echilibru, urmărind traseul stabilit.

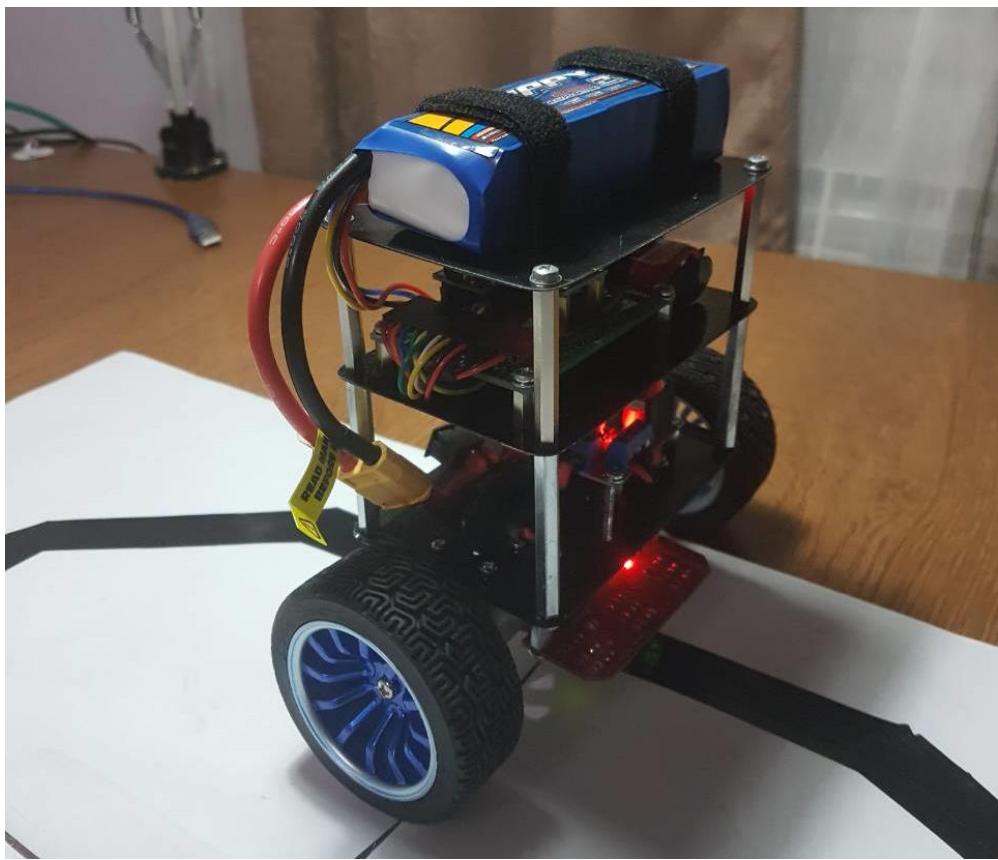


Fig. 24 Robot v2.0 în starea de echilibru, urmărind traseul stabilit

## 5 Concluzii

Pentru a avea un robot funcțional, arhitectura *hardware* și cea *software* trebuie să fie îmbinate proporțional și aduse la un standard optim. Componentele folosite au un rol esențial în menținerea stabilității robotului, dar acestea nu ar fi de folosință dacă avem un cod sursă mai puțin optimizat.

Codul sursă necesar pentru a îndeplini scopul propus nu este unul stulos, dar pentru a îndeplini scopul propus acesta trebuie să fie foarte eficient, o mică intrerupere sau eroare de calcul ar duce la pierderea totală a echilibrului. Pentru fiecare dintre roboți a fost nevoie de o adaptare diferită a codului și multe teste de calibrare pentru a obține un rezultat cât mai bun.

### 5.1 Contribuții personale

Am reușit prin îmbinarea elementelor *hardware* și *software* să realizez un robot funcțional care este capabil să decidă singur cum acționează în momentul în care trebuie să ajungă la o stare de echilibru, și momentul în care este posibilă urmarea traseului. Adaptările algoritmilor cât și implementarea metodelor pentru a realiza acest lucru sunt descrise pe larg în capitolele de mai sus.

Am reușit să realizez un robot compact, cu un minim necesar de resurse atât la nivel de *hardware* cât și la nivel de *software*.

### 5.2 Directii de viitor

Pentru viitor robotul poate fi dezvoltat și optimizat atât la nivel fizic cât și logic:

- Controlul robotului de la distanță prin *bluetooth* prin intermediul telefonului sau al unui controler.
- Aplicarea algoritmului PID de cel puțin două ori pentru a obține rezultate și mai bune
- Efectuarea calculelor mult mai repede cu ajutorul unui Raspberry pi care mai apoi să transmită rezultatele către Arduino
- Echiparea cu o serie de motoare *brushless*
- Memorarea traseului la prima parcurgere și posibilitatea de a îl reproduce fără traseul marcat
- Parcurea unor drumuri prestabilite, memorate.

## 6 Bibliografie

1. *White Collar: The American Middle Classes.* Mills, C. Wright. 1953.
2. **Nicolescu, Adrian Florin.** ROBOTICA prezent și perspective economice. [www.ttonline.ro](http://www.ttonline.ro). [Interactiv]
3. **Varum, Humberto.** *Accelerometers: Principles, Structure and Applications.* 2013.
4. **Urquiza, Arturo.** Schema de calcul PID. [Interactiv]
5. **Hagglund, K. Astrom and T.** *PID Controllers: Theory, Design and Tuning.*

## 7 Tabelul figurilor

Fig. 1 Evoluția din industria robotică (2).....	7
Fig. 2 Principiul de balansare robot .....	8
Fig. 3 Antrenarea motoarelor în momentul mișcării.....	9
Fig. 4 Robot v1.0 .....	10
Fig. 5 Robot v2.0 .....	10
Fig. 6 Schema de conectare a componentelor.....	11
Fig. 7 Placă dezvoltare Arduino Nano.....	12
Fig. 8 Schemă pini Arduino Nano .....	12
Fig. 9 Modul accelerometru cu giroscop .....	13
Fig. 10 Accelerometru .....	13
Fig. 11 Punte H .....	14
Fig. 12 PWM grafic .....	15
Fig. 13 Motor DC.....	16
Fig. 14 Modul senzori infraroșu.....	16
Fig. 15 Model 3D șasiu.....	17
Fig. 16 Diagrama codului .....	20
Fig. 17 Grafic valori PID .....	21
Fig. 18 Schema de calcul PID (4) .....	22
Fig. 19 Acționare puncte H.....	26
Fig. 20 Reprezentare date de ieșire PID.....	27
Fig. 21 Grafic în procesul de stabilizare .....	27
Fig. 22 Grafic în momentul stabilizat .....	27
Fig. 23 Diagrama aplicației.....	28
Fig. 24 Robot v2.0 în starea de echilibru, urmărind traseul stabilit.....	29