UNIVERSITATEA POLITEHNICA TIMIŞOARA

FACULTATEA DE ELECTRONICĂ ŞI TELECOMUNICAŢII

PROIECT DE DIPLOMĂ

Robot Autonom BB8

Conducător ştiinţific:

Profesor dr. ing. Aurel Gontean

Absolvent:

Cristian Paşcalău

Timişoara

2017

Sinteza lucrării

Lucrarea de diplomă este dezvoltată în cadrul Departamentului de Electronică Aplicată al Universității Politehnica din Timișoara. Lucrarea constituie rezultatul efortului depus pe parcursul mai multor luni, perioada noiembrie 2016 - iunie 2017. Această lucrare se incadrează intr-o tematică de interes pentru domeniul educațional si de noutate. Scopul lucrării este prezentarea teoretică și realizarea unui model experimental.

Capitolul 1 Introducere: comparație între diferite abordări

Capitolul 2 Proiectul mecanic: prezintă principalele componente mecanice ale robotului

Capitolul 3 Control: descrie partea de electronică și programare ce realizează funcțiile de control și comunicație cu robotul

Capitolul 4 Rezultate experimentale

Capitolul 5 Concluzii

Capitolul 6 Bibliografie

Cuprins

Cap. 1: Introducere

Domeniul roboticii a fost încă de la începuturi și este și în ziua de astăzi unul dintre cele mai interesante direcții de specializare pentru un electronist, dar totodată este și unul dintre cele mai complexe deoarece se află la intersecția mai multor științe de sine stătătoare precum mecanica, electronica, știința calculatoarelor, chimia prin știința materialelor, fizica și lista se lungește pe zi ce trece! Condițiile care au dus la apariția roboților sunt economice, de mediu, iar tehnologia din ziua de azi împinge robotica și spre domeniul mircotehnologiei și nanotehnologiei [1].

Divertismentul este și el un domeniu ce dă naștere unor roboți, existând ligi de lupte între roboți incluzând faimoșii sumo-boți, ligi de fotbal pentru roboți iar în următorii ani se va lansa și competiția de mașini electrice autonome de curse, în mărime naturală, Roborace [2].

Sursa: roborace.com

Figura 1: Mașina oficială Roborace

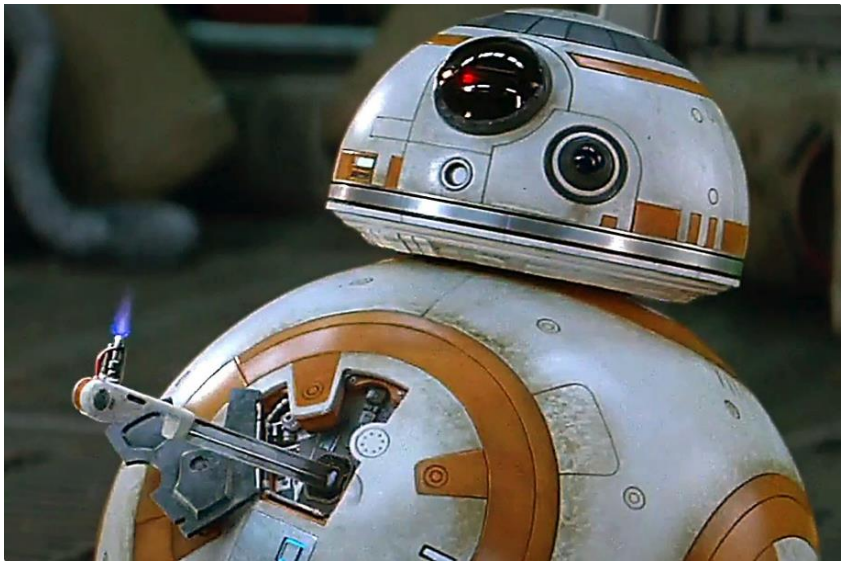
La sfârșitul anului 2015 a fost lansat un nou fim science-fiction din seria Star Wars, cu denumirea „The Force Awakens” în care apare un nou robot, cu denumirea consacrată de „droid”: BB8. Acesta se prezintă sub forma unei sfere ce se poate deplasa în orice direcție, iar deasupra sferei ce constituie corpul robotului se află capul robotului ce pare să plutească la mică distanță desupra corpului.

Ceea ce este specific acestui robot este că nu pare să existe o legătură între cap și corp, acesta plutind deasupra corpului indiferent de direcția de deplasare.

Sursa: gamedots.mx

Figura 2: BB8 în mișcare

În ciuda faptului că pentru realizarea filmului au fost folosite păpuși mânuite de oameni, o divizie specială a companiei „Disney” ce a produs filmul, numită „Disney Research” a venit cu o versiune reală, funcțională și aparent autonomă a robotului din film. Ceea ce nu se regăsesc la versiunea făcută de Josh Lee și Matt Denton pentru „Disney” sunt proiectorul de holograme și accesoriile din corp: un aparat cu electroșocuri, un sistem automat de fixare prin cabluri, un dispozitiv de prehensiune cu două degete și o brichetă.



Sursa: twitter.com

Figura 3: BB8 și bricheta

Între timp compania „Sphero” a lansat pe piață o versiune de jucărie, la scară mică, a robotului BB8 ce include și un proiector de holograme pentru realitatea virtuală iar pe internet s-a înființat o

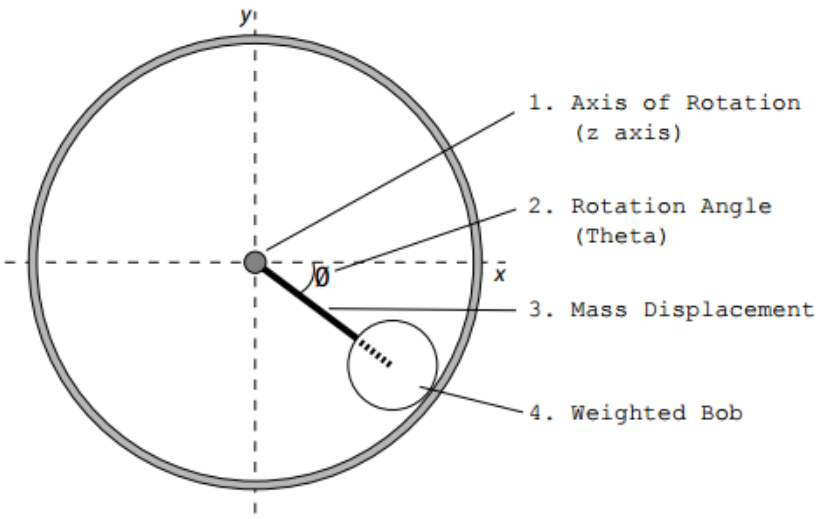
adevărată comunitate de constructori pasionați de acest personaj, sub forma unui forum numit bb8builders.club. Deși robotul BB8 are o formă simplă și familiară, acesta ridică mai multe probleme interesante din punct de vedere ingineresc cum ar fi: modul și mecanismul de conducere al robotului, rezistența la uzură, modalitatea de cuplare a capului, interfața cu utilizatorul uman și având în considerare că un model la scară poate depăși masa de 10kg, trebuie luat în calcul și aspectul de siguranță în interacțiunea cu oamenii.

Problema roboților sferici s-a mai pus în trecut în incercarea de realizare de roboți subacvatici [3] și roboți de explorare pentru securitate [4],[5].

Acum că tehnologiile de fabricație aditivă 3D și de prelucrare cu comandă numerică CNC devin din ce în ce mai accesibile publicului larg, întreaga comunitate online de costructori BB8 a avut capacitatea de a implementa și chiar dezvolta mai multe soluții la problema realizării unui astfel de robot. Toate au în comun cel puțin un aspect: capul are rotițe și este ținut desupra corpului printr-un cuplaj magnetic. În continuare vom analiza principalele abordări pentru construcția unui robot BB8 și vom stabili avantajele și dezavantajele fiecăreia:

1.1) Abordarea cu schimbare a centrului de greutate (Barycenter Offset)

Fiind vorba de un robot sferic, dacă acesta se află într-o stare de echilibru iar apoi mecanismul intern execută o mișcare ce duce la schimbarea centrului de greutate, atunci întregul robot se va rostogoli spre noua poziție de echilibru (poziția în care robotul are cea mai mică energie potențială gravitațională). Cu un control adecvat, robotul poate executa mișcări line în direcția dorită. Cea mai mare limitare a acestor tipuri de roboți este cuplul maxim ce se poate obține datorită imposibilității de mutare a centrului de greutate în exteriorul sferei [5]. Dacă considerăm un pendul în plan în interiorul unei sfere ca în figura următoare:



Sursa: mdpi.com/journal/robotics

Fig 4: Pendul în interiorul unei sfere

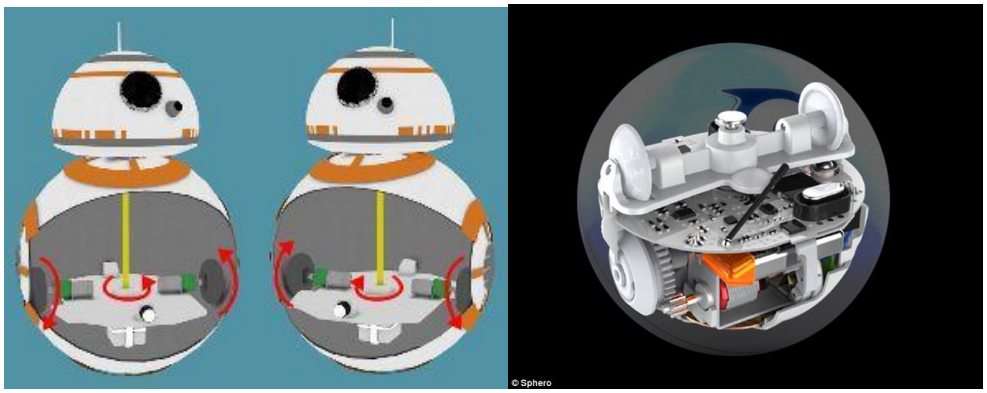
Atunci momentul de răsturnare în jurul axei Z (axa ce iese din desen) ce apare datorită masei deplasate 4 are expresia:

$$\tau_{max} = mgr * \sin(\theta) \quad (1)$$

Unde mg reprezintă greutatea masei deplasate 4 , τ_{max} reprezintă momentul de răsturnare în jurul axei Z , r reprezintă deplasamentul (brațul forței gravitaționale) masei față de centrul de masă al sferei iar θ reprezintă unghiul de rotație față de orizontală. Este evident că r nu poate depăși raza sferei și astfel cuplul maxim ce se poate obține este limitat superior.

Următoarele soluții au ca principiu schimbarea centrului de greutate:

1.1.1) Soluția roții de hamster



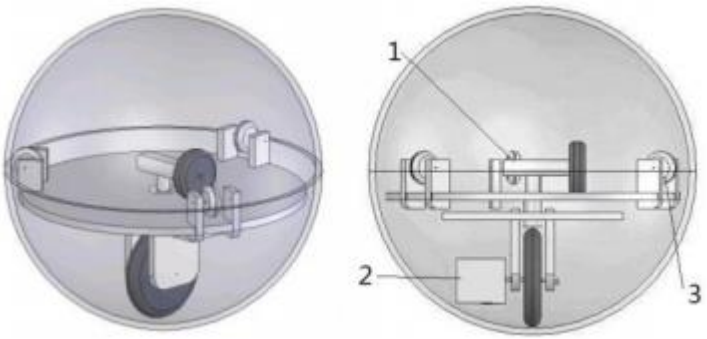
Sursa: hackaday.com

Figura 5: Soluția roții de hamster

Este soluția adoptată de compania Sphero și are în interior un mecanism cu două roți ce se poate deplasa în orice direcție prin conducerea diferențială a roților, inclusiv rotirea pe loc și astfel robotul posedă un caracter olonomic[1] parțial. Problema alunecării roților de conducere poate fi soluționată prin adăugarea de roți ce forțează contactul permanent între roțile de conducere și sferă. Jucăria Sphero, datorită dimensiunilor reduse, poate fi incărcată inductiv și astfel tot mecanismul este încapsulat în interior[6].
Avantaje: manevrabilitate ridicată, sistem aproape olonomic
Dezavantaje: e nevoie de o sferă perfectă, cea mai mică denivelare sau îmbinare va cauza probleme; nu există acces facil în interiorul robotului pentru depanare și schimbarea bateriilor; stabilitate redusă; există pierderi prin frecare

[1] Olonomie= în robotică un sistem olonomic este acela a cărui orientare nu afectează direcția de deplasare. Cu alte cuvinte robotul trebuie să poată să-și schimbe direcția de deplasare în orice moment al mișcării fără să se reorienteze.

1.1.2) Soluția cu o singură roată

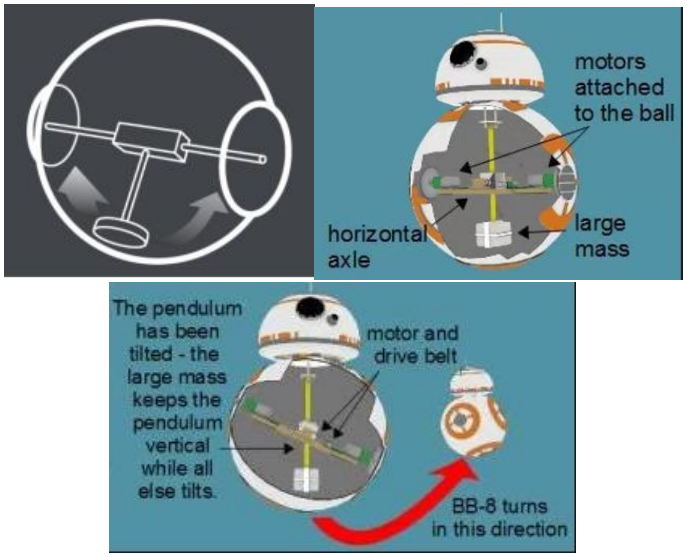


Sursa: mdpi.com/journal/robotics

Figura 6: Soluția cu o singură roată, 1- motor rotire, 2-motor principal, 3- roți ghidaj

Este asemănătoare cu soluția roții de hamster, dar sistemul de conducere și cel de virare sunt independente. Motorul pentru rotire învârte întregul mecanism de-a lungul ecuatorului interior în direcția dorită iar motorul pentru mișcare mută centrul de greutate al robotului și execută funcția de mișcare [5] .
Avantaje: manevrabilitate ridicată, sistem aproape olonomic
Dezavantaje: e nevoie de o sferă perfectă, cea mai mică denivelare sau îmbinare va cauza probleme; nu există acces facil în interiorul robotului pentru depanare și schimbarea bateriilor; stabilitate redusă ; există pierderi prin frecare

1.1.3) Soluția cu pendul



Sursa: hackaday.com

Figura 7: Soluția cu pendul

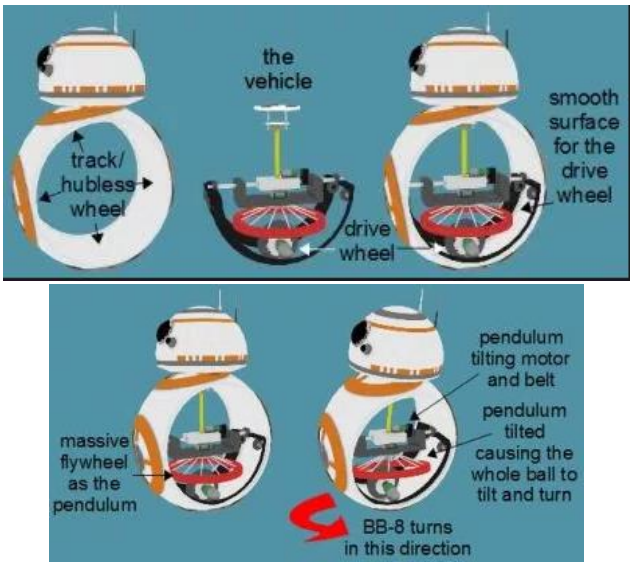
Este soluția adoptată de „Disney Research” și mulți membrii ai bb8builders.club. Robotul are un ax central fixat rigid în interiorul sferei în jurul căruia pivotează o greutate pentru a schimba centrul de masă al sistemului. Robotul virează prin intermediul unei mișcări combinate între mersul normal (înclinarea greutății față-spate) și înclinarea aceluiași braț vertical ce ține greutatea la stânga sau la dreapta .

Datorită acestui aspect robotul nu se poate învârti pe loc, dar greutatea de la baza sferei se poate înlocui cu un volant. Dacă volantul se poate roti în jurul axei verticale cu mare accelerație/ decelerție într-un sens, atunci datorită inerției mari a volantului, robotul se va roti în celălalt sens [6].

Avantaje: manevrabilitate bună, nu mai e nevoie de o sferă fără îmbinări

Dezavantaje: nu există acces facil în interiorul robotului pentru depanare și schimbarea bateriilor; e nevoie de un cuplu mare pentru rotirea greutății cu punctele de prindere situate la ecuator ; e nevoie de multe componente din metal pentru a avea rezistența necesară și implicit devine scump de construit

1.1.4) Soluția roții fără ax (sau soluția James Bruton)



Sursa: hackaday.com

Figura 8: Soluția roții fără ax

A fost realizată de James Bruton și este o combinație între roata de hamster și pendul. Pentru mecanismul de conducere e nevoie doar de o secțiune de sferă, asemănătoare unei roți, astfel accesul în interior este unul facil. Acționarea se face prin intermediul unei singure roți de conducere situată la baza secțiunii de sferă și astfel se obține, față de varianta cu pendul, un raport de reducție ce permite instalarea unui motor cu un cuplu mai redus. La robotul în mărime naturală raportul de reducție este de 5:1 iar acesta reduce și solicitările mecanice ale mecanismului intern. Pentru a reduce posibilitatea de alunecare a roții principale de conducere, dar și pentru a împiedica mecanismul intern să atingă secțiunea de sferă se vor folosi multiple roți de ghidaj. Și acest robot are un ax principal, dar spe deosebire de pendul, axul nu este legat rigid de sferă ci de căruciorul (colorat negru în figură) intern ce conduce mișcarea față-spate. De aceast ax este suspendat un volant (colorat roșu în figură) cu masă mare ce se poate înclina față de mecanismul de conducere față-spate și astfel prin mișcarea combinată de înclinare și conducere față-spate, robotul poate vira larg. De asemenea, volantul se poate roti în jurul axei verticale iar o accelerație bruscă într-o direcție dă naștere unei rotații a întregului robot în cealaltă direcție, datorită inerției volantului. Ceea ce mai lipsește din sferă poate fi completat ușor cu panouri detașabile.

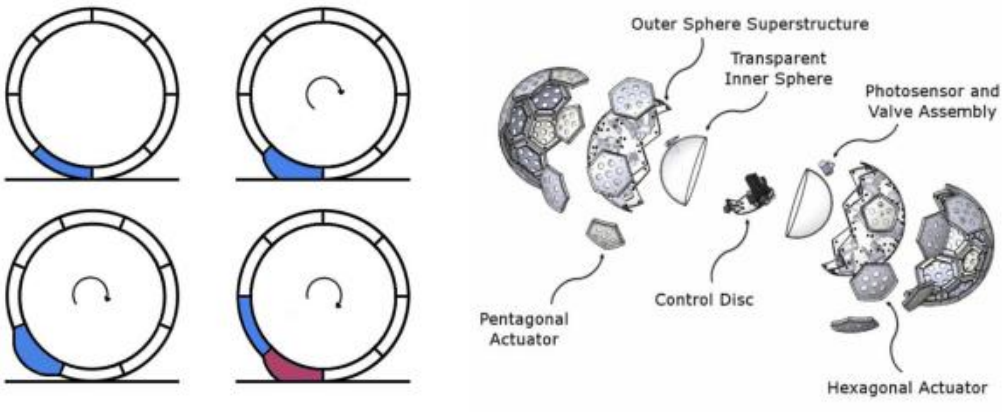
Avantaje: manevrabilitate bună, cost de realizare relativ redus, poate fi realizat în mare parte prin proces aditiv 3D, modele 3D disponibile online, are acces facil în interior

Dezavantaje: roata principală poate aluneca, apar pierderi prin frecare; nu poate vira strâns

1.2) Abordarea cu schimbare a formei sferei

O altă modalitate de obținere a mișcării controlate a unei sfere este prin modificarea formei robotului. Această abordare este o idee nouă și mai este loc pentru cercetare, acest tip de roboți sunt mult mai rar întalniți spre deosebire de cei de la punctul 1.1) , dar au avantajul că pot beneficia de olonomie.

1.2.1) Soluția cu camere pneumatice



Sursa: mdpi.com/journal/robotics

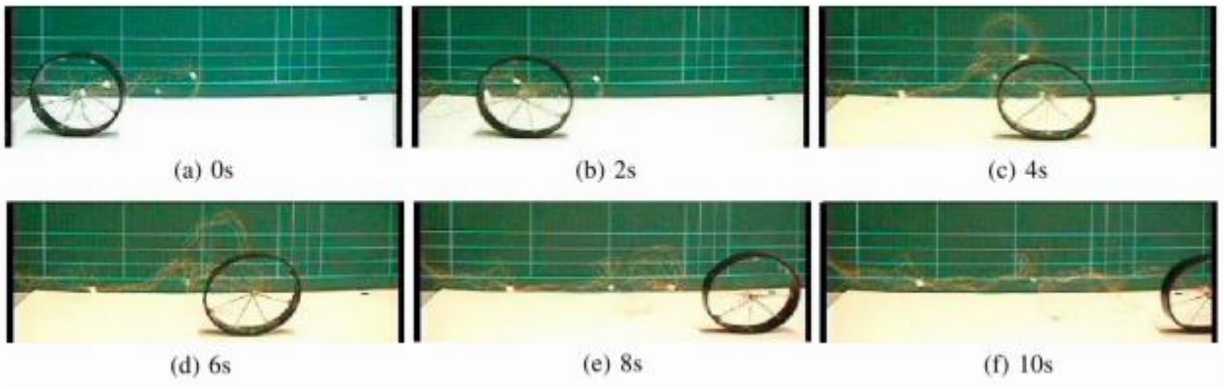
Figura 9: Soluția cu camere pneumatice

A fost propusă de M. Artusi [7] și se bazează pe presurizarea selectivă a unor pernițe pneumatice aflate pe suprafața exterioară a sferei. Teoretic sfera se poate controla fără un mecanism intern complicat iar robotul poate fi complet olonom.O metodă interesantă este soluția găsită pentru creearea unei mingi de fotbal ce are la exterior pernițe hexagonale și pentagonale, fiecare cu propriul ei fotosenzor orientat spre interior, ce controlează câte o valvă pneumatică. În interior se află o sferă transparentă ce conține un mecanism din cele prezentate anterior, soluția de tip roată de hamster. Pe partea inferioară a discului din sferă, ce constituie baza mecanismului roată de hamster, se află un LED cu intensitate mare ce luminează selectiv fotodiodele actuatoarelor pneumatice exterioare astfel încât acestea să presurizeze segmentul respectiv și să dea naștere mișcării dorite [8] .

Avantaje: rezistent la șocuri, sistem de mișcare inovativ

Dezavantaje: greu de construit, cost ridicat

1.2.2) Soluția aliajelor cu memorie (Shape memory alloys)



Sursa: mdpi.com/journal/robotics

Figura 10: Soluția aliajelor cu memorie

Soluția oferită are în componență arcuri realizate din aliaje cu memorie ce se pot micșora/alungi atunci când este aplicată o tensiune electrică la capetele lor. Dacă arcurile sunt conectate mecanic în interiorul unei roți în formă de stea, atunci această modificare controlabilă a arcurilor poate deforma învelișul exterior și astfel poate genera o mișcare[9].

Avantaje: nu are componente mecanice interne în mișcare, sistem de control relativ simplu

Dezavantaje: e nevoie de materiale rare, greu de stabilizat, greu de realizat sub formă sferică, nu oferă soluție de fixare a capului

1.3) Abordarea utilizând conservarea momentului unghiular

Sursa: mdpi.com/journal/robotics

Figura 11: Robot realizat utilizând giroscoape pentru un cuplu ridicat

Roboții prezentați la punctul 1.1) sunt de departe cei mai întâlniți deoarece sunt relativ ușor de realizat și de controlat însă au neajunsul cuplului maxim ce se poate obține datorită imposibilității scoaterii în afara sferei a masei inerțiale. S-a încercat adăugarea de giroscoape mecanice cu scopul de control al momentului(Control Moment Gyroscope sau CMG). Prin rotirea cu viteză a unui volant și rotirea lui în jurul unei axe, legile conservării momentului unghiular pot fi utilizate pentru controlul mișcării sferei. Utilizând această metodă, cuplul mecanismului intern este dat de viteza de rotație a

giroscopului CMG: dacă viteza crește, atunci crește și cuplul. O proprietate a utilizării de giroscoape CMG este că apar forțe de reacție în toate cele 3 axe spațiale. Dacă CMG se învârte în jurul axei X și este rotit în jurul axei Y, atunci va apărea un cuplu de forțe după axa Z , acest efect se numește *precesiune giroscopică.* Acesta poate fi un efect dorit, dând naștere unui cuplu controlabil, dar poate cauza și efecte nedorite pentru controlul robotului [10].

Cuplul ce se poate obține prin această metodă are expresia:

$$\tau_{max} = mgr * \sin(\theta) + h_{net} \quad (2)$$

,unde h_{net} este cuplul obținut prin efecte giroscopice.

Avantaje: foarte stabil, se poate obține un cuplu ridicat

Dezavantaje: foarte greu de controlat, e nevoie de materiale cu rezistență mecanică ridicată, cost ridicat.

Prin analiza tuturor soluțiilor am ajuns la concluzia că cea mai potrivită resurselor limitate de buget, materiale și timp este soluția roții fără ax sau soluția James Bruton.

Sursa: hackaday.com

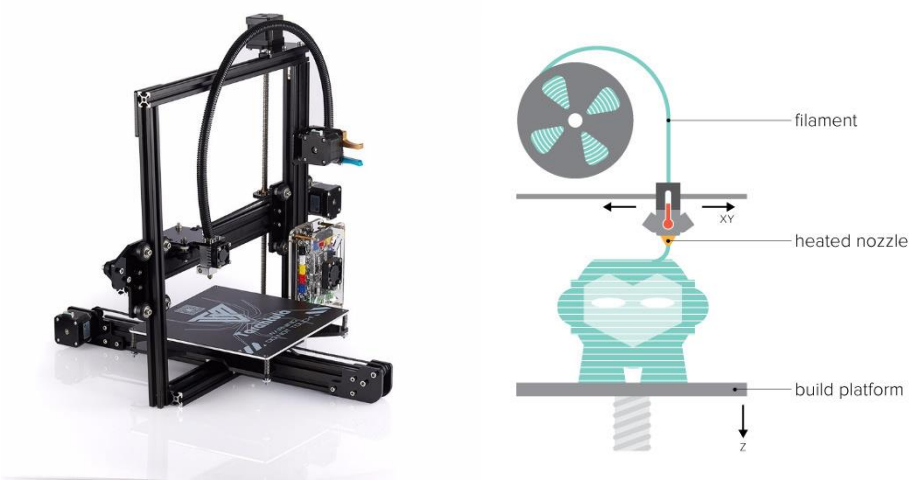
Figura 12: Soluția James Bruton

Cap. 2: Proiectul Mecanic

Proiectul mecanic a fost realizat în mare parte de către James Bruton [11] , modelele au fost create în mediul de proiectare mecanică asistată de calculator gratuit numit 123D Design. Acest program a fost retras de pe piață din luna martie, 2017 [12] .

Proiectul conține peste 70 elemente mecanice individuale, deci în continuare voi prezenta în detaliu numai subansamble mecanice fără să se pună neapărat accentul pe fiecare componentă în parte. Multe dintre aceste elemente sunt rezultatul unor segmentări cauzate de limitările tehnologice ale imprimantelor 3D adresate publicului larg.

Imprimanta 3D folosită pentru acest proiect se numește Tevo Tarantula i3, are un volum maxim de printare de 200 x 300 x 200 mm^3 și se bazează pe tehnologia numită Fused Deposition Modeling (FDM) sau Plastic Jet Printing (PJP). Aceasta presupune trecerea unui polimer filiform printr-un ajutaj încins la o temperatură bine controlată și depunerea acestuia în formă topită în straturi subțiri cu grosimi uzuale reglabile între 0.1mm și 1mm . O altă clasificare a imprimantelor 3D spune că această imprimantă este carteziană, deoarece lucrează cu coordonate carteziene după cele 3 axe X,Y și Z.



Sursa: og3dprinting.com/3d-printing

Figura 13: Imprimanta Tevo Tarantula i3 și principiul de funcționare FDM

Ultimul și unul dintre cei mai importanți pași înainte de realizarea efectivă a componentelor este alegerea materialului, cerințele de material pentru acest proiect sunt următoarele:

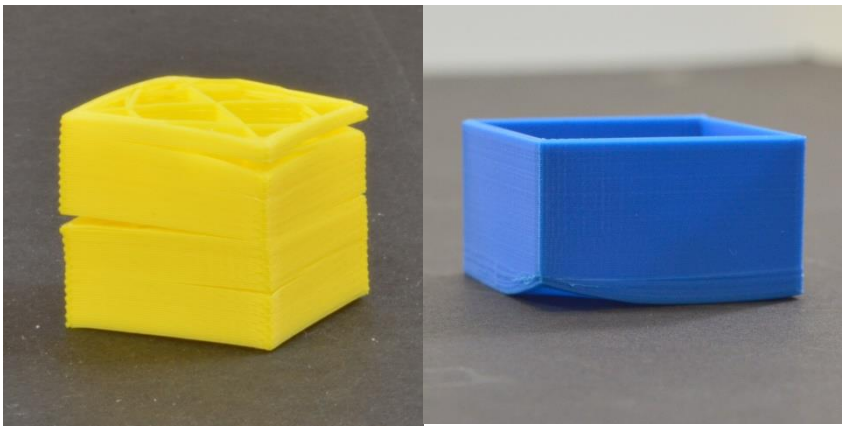
- Să fie ușor de procurat
- Să aibă rezistență mecanică ridicată
- Să aibă un mic grad de elasticitate pentru a rezista uzurii și șocurilor repetate
- Să poată fi lipit puternic de alte segmente din același material
- Să permită prelucrarea mecanică ușoară prin procese precum găurirea, șlefuirea și taierea
- Să fie un bun izolator electric
- Să aibă rezistență crescută la temperatură

Prima cerință restrânge lista de materiale disponibile la ABS(Acrylonitrile Butadiene Styrene) , PLA (Polylactic Acid) și Nailon.

PLA are avantajul că este cel mai ușor de printat, nu este nevoie de foarte multe calibrări și reglaje pentru imprimantă, dar lista cu dezavantaje este putin mai lungă: este casant, își pierde din proprietățile mecanice la temperaturi joase începând de la 40˚C, nu poate fi lipit cu ușurință și este greu de prelucrat mecanic.

Nailonul este un material cu proprietăți mecanice foarte bune, este cel mai rezistent la întindere, rupere și torsiune dintre cele trei, are și un mic grad de elasticitate și are cel mai ridicat punct de topire dintre cele 3 materiale. Acest din urmă fapt implică și ca imprimanta să aibă un cap de printare capabil de temperaturi de peste 240˚C, dar imprimanta Tevo Tarantula nu are. Un alt dezavantaj al acestui material este că nu poate fi lipit cu ușurință [13].

ABS este un termoplastic mai rezistent și mai flexibil decat PLA, rezistă la temperaturi mult mai mari de până la 100˚C, este foarte ușor de prelucrat mecanic și poate fi lipit foarte puternic cu un solvent uzual, acetona. Cel mai mare dezavantaj al acestui material este coeficientul relativ mare de dilatare cu temperatura, dacă materialul este extrudat la termperatura de 230˚C și este lăsat să se răcească la temperatura camerei atunci în urma procesului acesta se va contracta cu aproap 1.5% . Acesta face ca elementele printate să se contracte în timp ce se răcesc iar între straturile depuse, aflate la temperaturi diferite, apar crăpături.



Sursa: simplify3d.com/support/print-quality-troubleshooting

Figura 14:Fenomenele de separare a straturilor și warping

Un alt fenomen datorat tot coeficientului de dilatare cu temperatura se numește warping și are ca efect desprinderea în timp bazei elementului printat de pe suprafața de printare, chiar dacă inițial a aderat corespunzător. Printre soluțiile problemei se numără: utilizarea unei suprafețe de printare încălzite, printarea pe primul strat a unei baze extinse (brim), cu rol de sacrificiu , pentru element sau utilizarea de adezivi pentru a asigura buna fixare a elementului de patul de printare. Ca adeziv se pot folosi atât soluții comerciale, dar și ceea ce comunitatea de 3D-printing numește ABS slurry sau suc ABS(ABS juice). Acesta se realizează prin dizolvarea unei mici cantități de material ABS în acetonă.

Atât fenomenul de separație a straturilor, cât și cel de warping pot fi reduse sau eliminate prin utilizarea unei incinte pentru imprimantă care să țină elementul printat la o temperatură uniformă și constantă. Este recomandat ca incinta să fie construită dintr-un material ignifug și bun izolator termic.

2.1) Sfera

Sfera este corpul robotului sau elementul mecanic ce intră în contact cu mediul și trebuie să aibă rezistență structurală ridicată. Deoarece robotul se deplasează într-o singură direcție, cu mici variații date de unghiul de inlcinare față de verticală al sferei, suprafața efectivă de rulare este un segment de sferă. Am realizat segmentul de sferă din 20 de elemente cu bordură lipite între ele cu soluție de ABS dizolvat în acetonă. Elementele au o grosime de 5mm iar pentru o rezistență crescută, bordura este întărită cu suporți sub formă de segmente de cerc. Bordura are atât rol structural, cât și rol de ghidaj pentru mecanismul intern.

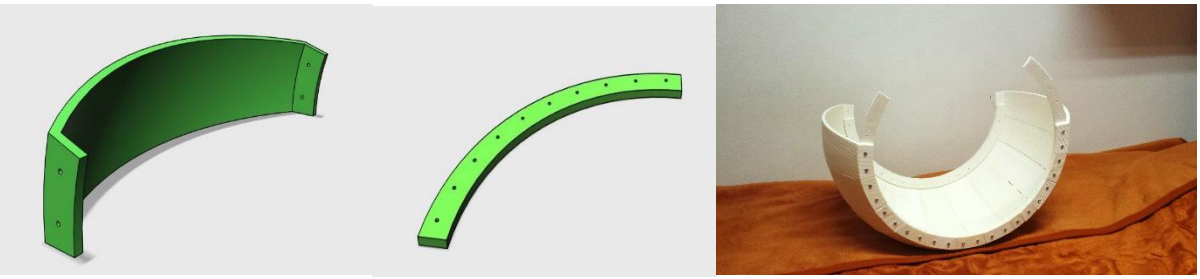


Figura 16: Un element, un suport și segmentul de sferă incomplet

Datorită fenomnului de contracție inegală a materialului și a dimensiunilor mari ale elementelor, diferentele de dimensiuni dau naștere unor denivelări perpendiculare pe suprafața de rulare. Acestor denivelări se adaugă și un alt defect tot sub formă de denivelări cu frecvență ceva mai mare, numit Z-wobble, cauzat de această dată de imperfecțiunile imprimantei la nivelul șurubului ce controleaza axa Z.

Deoarece suprafața de rulare trebuie să fie cât mai netedă, a fost nevoie de completarea spațiilor cu soluție de ABS urmată de șlefuire; realizată în mare parte cu un aparat specializat de șlefuit, dar și manual. Acest proces laborios l-am repetat de mai multe ori.

În total, pentru printarea și prelucrarea corpului robotului am avut nevoie de peste 300 de ore, 3.4 kg de ABS și 1 L de acetonă.

Figura 17: Corpul robotului înainte și după prelucrare

2.2) Căruciorul principal

Căruciorul este o componentă de bază a mecanismului intern și are rolul de a realiza deplasarea față-spate prin schimbarea centrului de greutate. Prin rotirea căruciorului în interiorul corpului, centrul de greutate se modifică și se îndepărtează de punctul de contact dintre robot și suprafața de rulare și duce la rostogolirea întregului robot spre poziția în care acesta are o energie potențială gravitațională minimă.

Căruciorul este format din 2 brațe sub formă de secțiune de cerc, fiecare realizat prin alipirea a 6 elemente separate pentru reducerea deformărilor cauze de contracția materialului. Cele două brațe sunt unite la fiecare capăt prin cate un suport și la mijloc prin suportul motorului principal. Poziționarea motorului este importantă, cu cât motorul este poziționat mai jos și mai aproape de corpul robotului, cu atât robotul va avea o stabilitate mai bună și va putea dezvolta un cuplu mai mare prin deplasarea motorului față de punctul de echilibru (vezi cap. 1.1).

Roata motorului principal este principala componentă în acționarea robotului, am imprimat-o din ABS și un material flexibil numit Thermoplastic polyurethane (TPU) [14] pentru suprafața de rulare, iar prin diametrul ei de 10cm oferă un raport de reducție de la diametrul sferei de 50cm de 5:1. Aceasta înseamnă că sfera se va roti cu o viteză unghiulară de 5 ori mai mică decât vietza unghiulară a roții motorului, dar și cuplul la nivelul sferei, în condiții de contact perfect fără alunecări între roată și sferă, va fi de 5 ori mai mare.

Pentru a asigura buna rulare a căruciorului și un bun contact al roții principale cu corpul robotului am atașat 11 roți cu diametrul de 5.5cm cu rol de ghidaj. Pentru a reduce frecarea, fiecare roată are un rulment 626 ZZ.

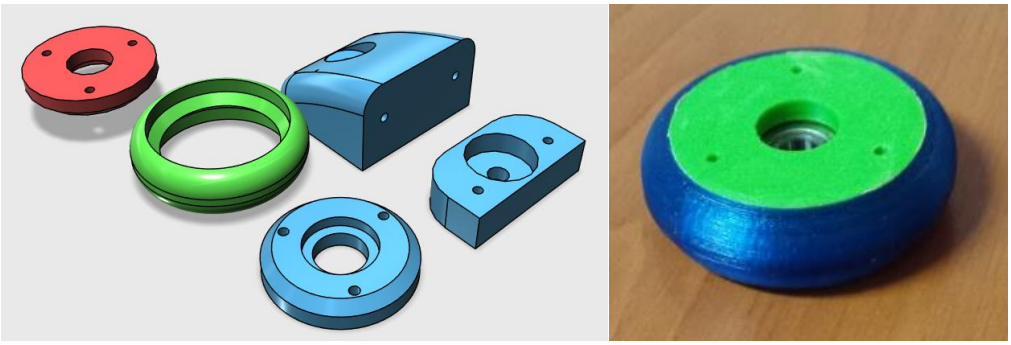


Figura 18: Elementele componente ale unei roți de ghidaj și o roată asamblată

Un alt rol important al căruciorului este de a susține axul central al robotului, în jurul căruia se va realiza mișcarea de înclinare.

Figura 19: Căruciorul în exteriorul și în interiorul roții

2.3) Mecanismul de înclinare

Pentru înclinarea robotului la stânga sau la dreapta e nevoie de o modalitate de mutare a centrului de greutate spre dreapta sau spre stânga. Această mișcare se execută sub forma unei rotații în jurul unui ax transversal sprijinit pe suporții căruciorului. Pentru o rotație cu frecare redusă am folosit 2 rulmenți de tip 6260 prinși atât de ax cât și de mecanismul de înclinare prin suporți de plastic.

Mecanismul este format dintr-o parte mobilă, cea care atârnă de axul principal și o parte fixă: o șină cu dinți curbată lipită cu solvent de căruciorul principal. Pe șina curbată va rula o roată cu dinți antrenată de un motor ce face parte din partea mobilă a mecanismului de înclinare.

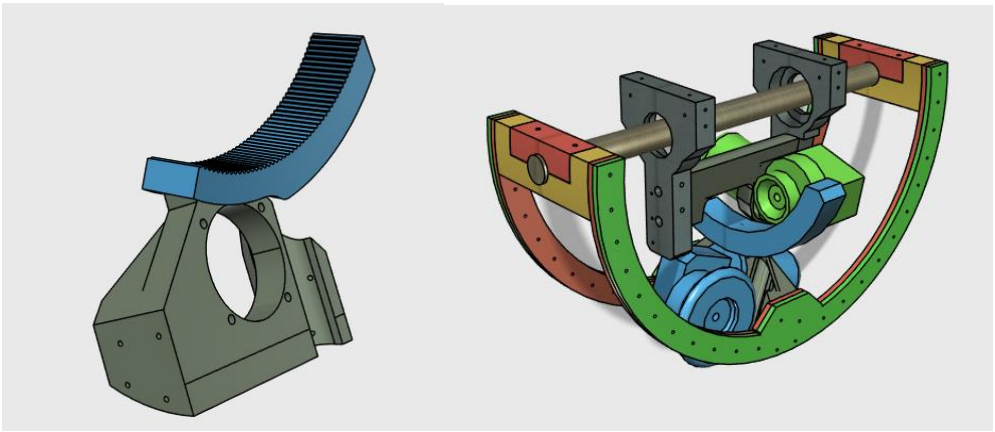
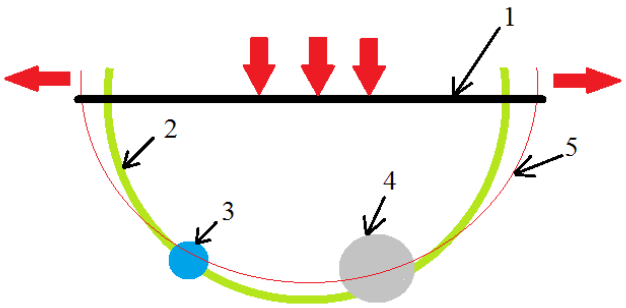


Figura 20: Șina curbată și parte a mecanismului de înclinare

Datorită diferențelor de formă și dimensiuni între motorul folosit în proiectul lui James Bruton și motorul disponibil pentru acest proiect, am reproiectat mare parte a mecanismului de înclinare tot în mediul de proiectare 123D Design și anume: toate cele 3 elemente ale suportului părții mobile a mecanismului de înclinare, șina cu dinți curbată fixată de cărucior și roata dințată antrenată de motorul pentru înclinare.

Dacă părții mobile i se atașează un corp cu masă mare comparativ cu masa segmentului de sferă și a căruciorului, atunci orice rotație a mecanismului mobil de înclinare va putea duce la schimbarea centrului de greutate a robotului și implicit la înclinarea sa.

O masă mare poate duce totuși la deformarea căruciorului pe care se sprijină axul principal, dacă asupra axului se exercită o forță verticală orientată în jos, atunci capetele căruciorului se vor deforma spre exterior, putând duce la o poziționare incorectă a roților de ghidaj și la contactul dintre cărucior și corpul robotului.



1-axul principal; 2- căruciorul înainte de deformare; 3- roată de ghidaj;
4- roata motorului principal; 5- căruciorul după deformare

Figura 21: Deformarea căruciorului

Pentru a împiedica apariția acestei deformări am fixat la ambele capete ale axului principal câte un opritor ce nu permite deformarea spre exterior prezentată anterior. Opritoarele sunt șaibe de oțel fixate de o bară filetată, situată în interiorul axului principal și străbate întreaga sa lugime, prin sudare cu electrozi înveliți.

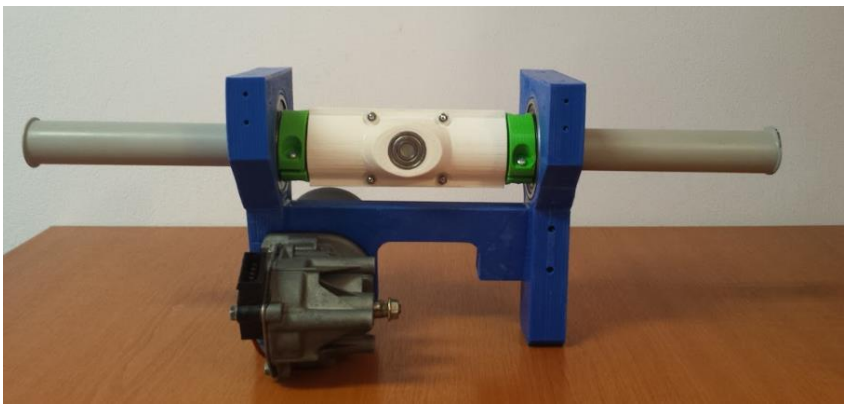


Figura 22: Parte a mecanismului de înclinare și opritoarele axului principal

Masa ce se atașează părții mobile a mecanismului de înclinare are rol dublu:

- ✓ Este folosită pe post de masă inerțială de mecanismul de înclinare pentru înclinarea robotului
- ✓ Este folosită pe post de volant pentru rotirea pe loc a robotului

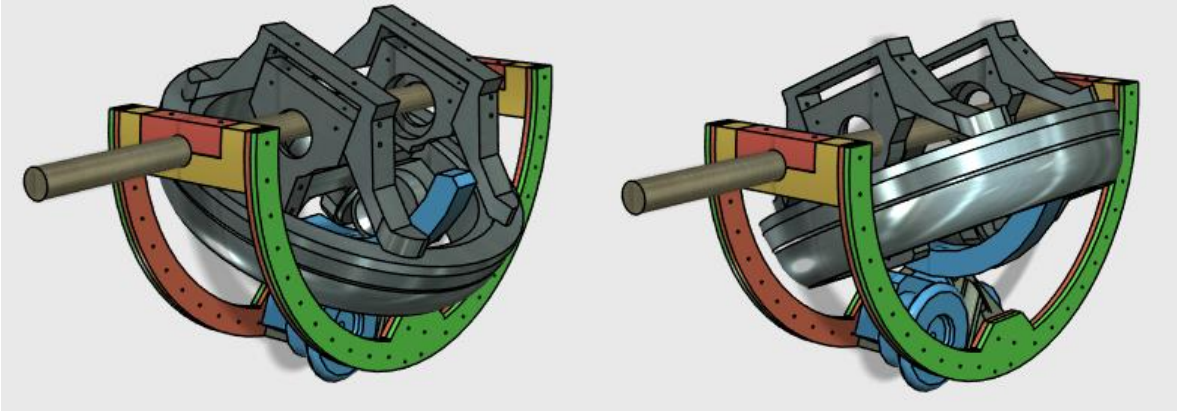
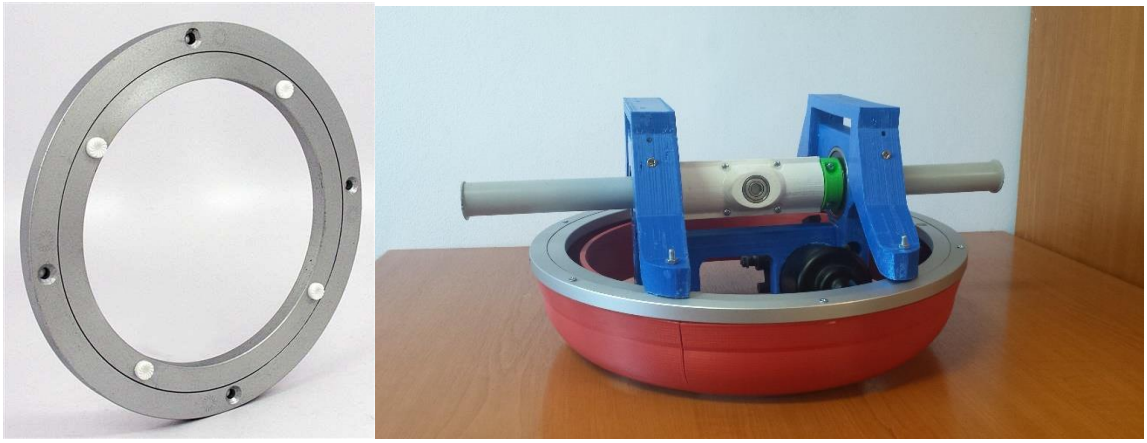


Figura 23: Mecanismul de înclinare al robotului

Pentru ca masa inerțială să aibă și rol de volant, ea este distribuită uniform sub formă de colac și fixată de mecanismul de înclinare prin intermediul unui rulment de tip Lazy Susan ce permite rotirea cu ușurință a volantului.



Sursa: bearingscanada.com

Figura 24: Rulment Lazy Susan și volantul

Pentru ca volantul să îndeplinească cerințele de masă de 6kg, trebuie ca formele din ABS să fie umplute cu un material dens. Volumul maxim de material cu care se pot umple formele pentru volant este de $1500cm^3$, ceea ce înseamnă că materialul ales trebuie să fie cel puțin 2 ori mai dens decât betonul, în condițiile în care acesta are o densitate de $2{,}5\,\frac{g}{cm^3}$. Un bun candidat este oțelul cu un preț redus și densitate de $8\,\frac{g}{cm^3}$ dar acesta evident nu poate fi turnat în forme de plastic iar umplerea formelor cu bucăți mici de oțel, spre exemplu cuie, ar rezulta într-un volum real de material insuficient pentru a atinge masa dorită. Materialul ce l-am ales este plumbul, cu densitatea suficientă de $11{,}34\,\frac{g}{cm^3}$ [15], sub formă de lamele de dimensiune 10 x 150 mm obținute din tablă de plumb și fixate în formele din ABS cu adeziv.

Acționarea volantului se face cu motoare situate diametral opus, în interiorul circumferinței volantului, fixate prin suporturi de partea mobilă a mecanismului de înclinare. Fiecare motor are câte o roată de acționare pe ax, roată în mare parte solidă cu o membrană din material flexibil la suprafața de contact cu volantul.

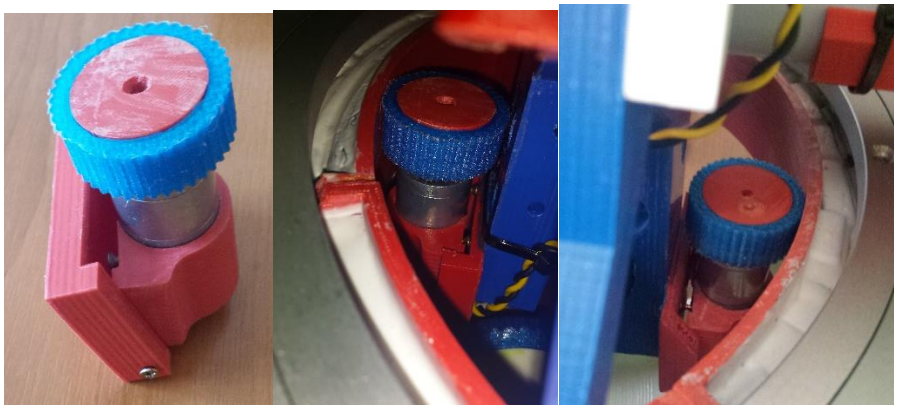


Figura 25: Motoarele de acționare a volantului

2.4) Brațul de control al capului

Brațul de control al capului este mecanismul intern prin care se poate controla pozitia capului într-un interval limitat. Brațul de control este fixat la bază de axa principală printr-o articulatie universală ce permite mișcarea în orice direcție, iar în partea superioară este conectat prin intermediul unor cuplaje semiflexibile la 2 servomotoare de putere. Cele două servomotare pot fi privite ca 2 colțuri fixe ale unui triunghi, celălalt colț fiind brațul de control al capului iar poziția lui este controlată prin modificarea a 2 laturi ale triunghiului ; ultima latura, cea care se găsește între servomotoare este evident fixă și nu se poate modifica.

Servomotoarele de control sunt legate rigid de căruciorul robotului și de axul principal prin intermediul unui suport compus din 8 elemente mecanice.

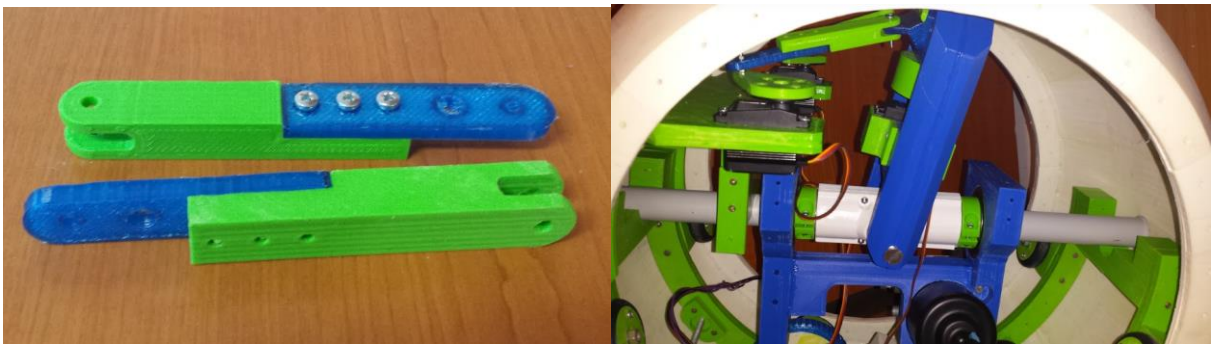


Figura 26: Cuplajele semiflexibile și poziția brațului de control al capului

Brațul de control este prevăzut cu un servomotor conectat rigid, printr-ul ax, la elementul ce face legătura dintre mecanismul intern și capul situat în exteriorul corpului și anume cuplajul magnetic. Acesta este de fapt un suport pentru 3 magneți din neodim N52 cilindrici.

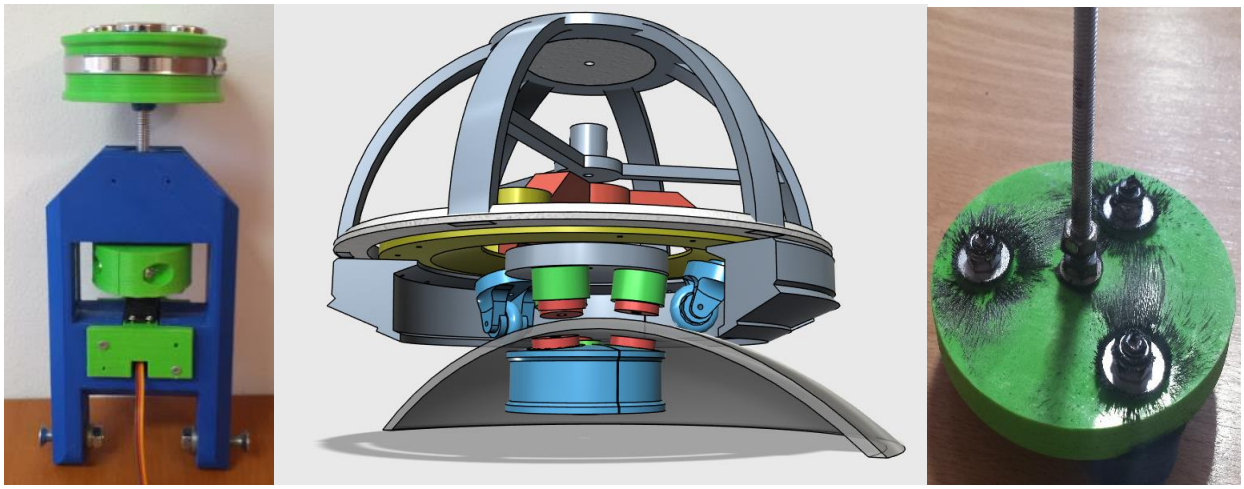


Figura 27: Brațul de control al capului și cuplajele magnetice

În figura 27 se observă alinierea piliturilor de fier în lungul liniilor de câmp magnetic și se poate deduce chiar din poză aranjarea N-S-N. Între magneții de aceeași polaritate nu se regăsesc pilituri de fier, în schimb acestea sunt aranjate între magneții de polarități opuse.

Capul este format principial din 2 părți: baza capului, de care sunt fixate roțile și domul capului, acesta se poate roti independent de baza capului. Magneții capului sunt conectați la dom și prin intermediul servomotorului aflat pe brațul de control al capului, din mecanismul intern, se poate controla rotația domului independent de baza capului. Dacă domul și baza capului ar fi fost legate rigid, atunci orice schimbare a poziției capului ar determina și o rotație nedorită a acestuia datorită caracterului non-olonomic al roților.La o analiză mai atentă a filmului „The Force Awakens” se poate observa că acesta este modul de funcționare al robotului și în film.

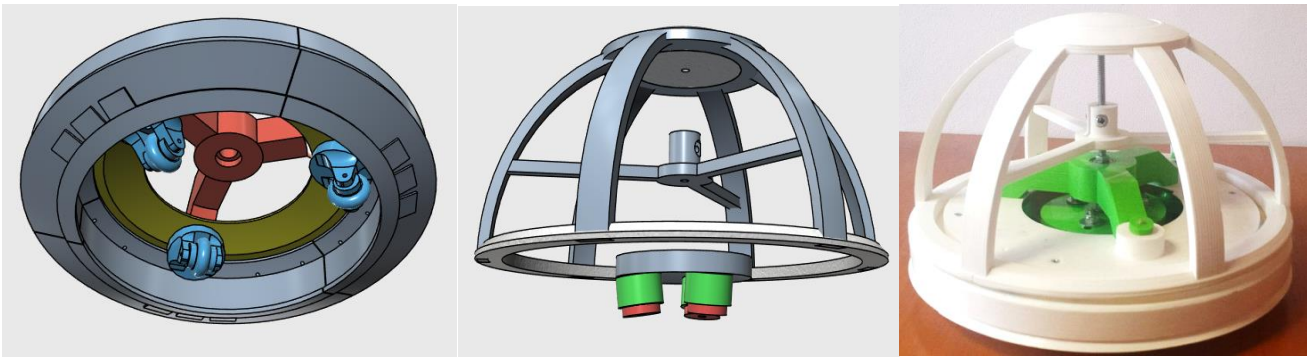


Figura 28: Capul și elementele componente

Motivele pentru care acționarea mișcării de rotație a capului se face în mecanismul intern sunt simple, capul trebuie să fie cât mai ușor și o acționare în interiorul capului este condiționată de existența unei surse de energie în interiorul capului ce duce ma departe la îngreunarea acestuia.

Forța de atracție între 2 magneți cilindrici identici are expresia [16] :

$$F(x) = \frac{\pi \mu_0}{4} M^2 R^4 \left[\frac{1}{x^2} + \frac{1}{(x+2t)^2} - \frac{2}{(x+t)^2} \right] \quad (3)$$

Unde: R reprezintă raza magnetului cilindric

t reprezintă înălțimea magnetului

x reprezintă distanța dintre magneți

M reprezintă magnetizația

$\mu_0 = 4\pi \cdot 10^{-7}$ H/m reprezintă permeabilitatea magnetică a vidului

$$\mathbf{M} = \frac{1}{\mu_0} \mathbf{B}_r \quad (4)$$

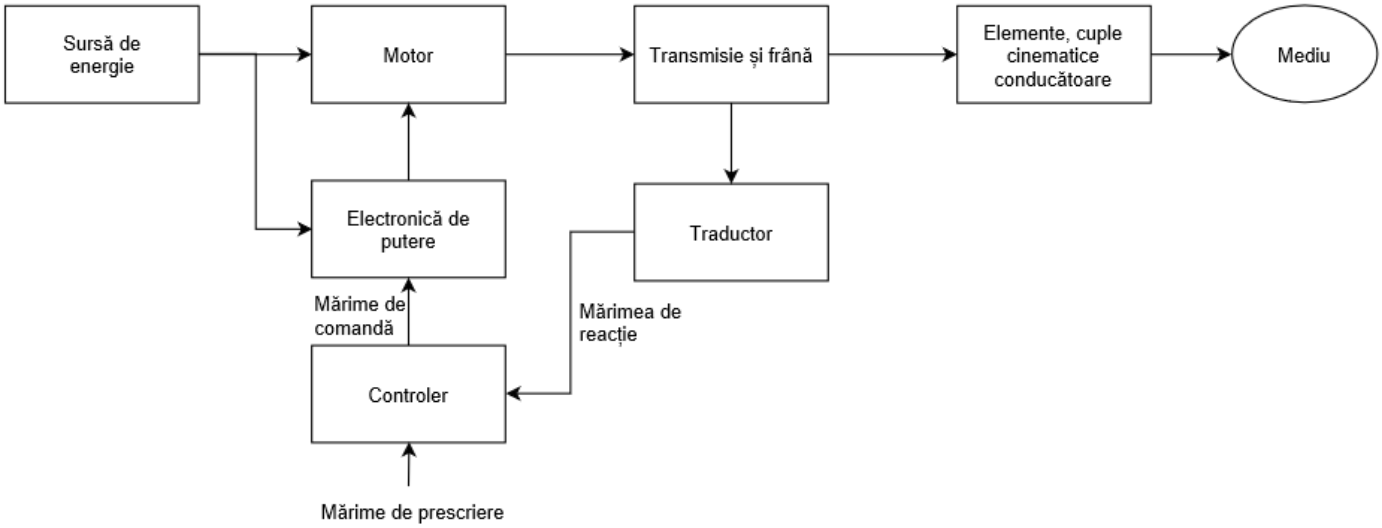
Unde: B_r reprezintă fluxul magnetic remanent al magneților [17].

Observăm din formula (3) că forța de atracție a magneților scade cu pătratul distanței dintre ei, deci din acest punct de vedere este vital ca greutatea capului să fie cât mai mică deoarece distanța dintre magneții din interiorul corpului robotului și magneții din cap este mare, de ordinul de mărime al înălțimii magneților.

Stabilitatea robotului este condiționată și ea de distribuția masei, iar pentru ca acesta să fie stabil este de dorit ca centrul de masă să fie cât mai jos, cât mai aproape de punctul de contact dintre robot și suprafața de rulare. Un cap mai greu duce la înălțarea centrului de masă al robotului, deci și din punct de vedere al stabilității se dorește ca masa capului să fie cât mai mică.

Cap. 3: Control

Controlul robotului este realizat de partea de electronică și programul aferent fiecărui microcontroler. Controlul global îi revine, prin intermediul unei telecomenzi, în totalitate operatorului uman. Totuși , la nivelul fiecărei cuple cinematice conducătoare a robotului, există câte un sistem de conducere locală ce reazlizează funcția de reglare automată. Aceasta se poate realiza numai prin închiderea unei bucle de reacție.



Sursa: carte Bogdanov

Figura 29: Sistem de conducere locală

În sistemul de conducere locală prezentat în figura 29, controlerul are funcția de regulator astfel:

- preia mărimea de prescriere sau setpoint (în cazul acestui proiect mărimea de prescriere pentru toate sistemele de conducere locală este un unghi) de la logica de nivel superior
- preia informații despre mișcarea executată în realitate de elementele mecanice de la traductor, informații ce reprezintă mărimea de reacție
- calculează diferența dintre mărimea de prescriere și mărimea de reacție
- calculează mărimea de comandă cu care se corectează eroarea; calculul se realizează după o formulă numită algoritm de reglare numerică
- transmite mărimea de comandă electronicii de putere și mai departe motorului cu scopul executării unei mișcări în sensul anulării erorii [1]

3.1) Sistemul electronic

Având în vedere că robotul este mobil și radiocomandat, proiectul este împărțit în 2 subsisteme electronice alimentate de la baterie.

3.1.1) Subsistemul telecomandă

Telecomanda are sarcina de a realiza interfața cu utilizatorul și a transmite informațiile citite la robot. De la telecomandă se pot comanda direct sistemele de conducere locală prin modificarea mărimii de prescriere.

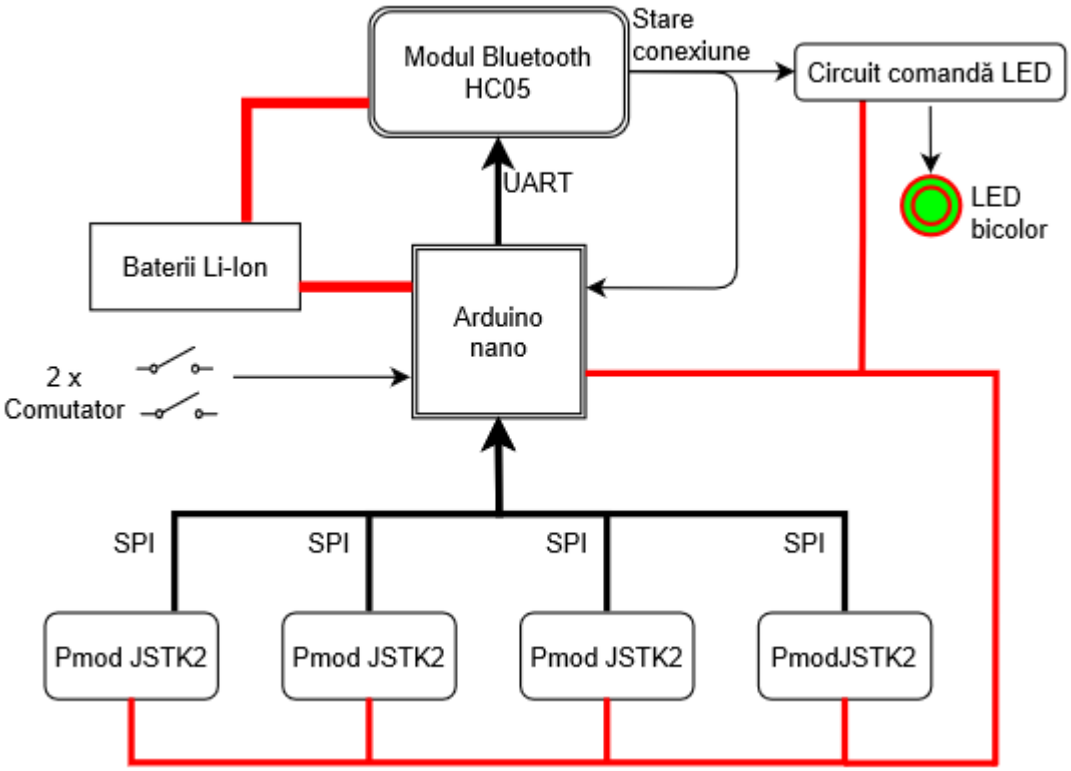


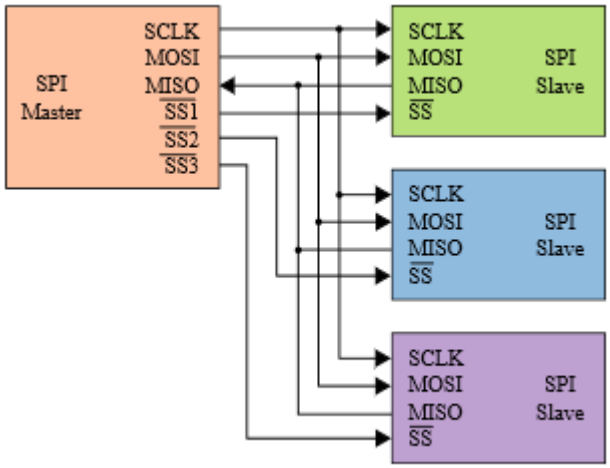
Figura 30: Schema bloc electronică a telecomenzii

Obiectul central al telecomenzii este placa de dezvoltare Arduino nano, versiunea 3.3V. Aceasta conține un microcontroler **atmega328** cu arhitectură RISC avansată pe 8 biți ,memorie flash 32KB, frecvență a tactului de 16MHz, 22 pini de intrare/ieșire din care 8 pot fi utilizați ca intrări analogice și 6 pot furniza semnal dreptunghiular modulat în lățime (PWM) și poate cel mai important pentru acest proiect, 2 periferice de tip interfață serială SPI și un periferic de tip interfață serială UART. După cum se vede și pe schema bloc, acestea sunt cele 2 interfețe de comunicare ce realizează transferul de informație între Arduino și modulele cu joystick Pmod JSTK2 și respectiv Arduino și modulul de comunicație fără fir HC05.

Protocolul SPI (Serial Peripheral Interface) – este un procol de comunicație serială sincronă dezvoltat la sfârșitul anilor 1980 pentru a fi folosit în comunicația pe distanțe mici. Dispozitivele ce

utilizează SPI pot comunica în regim *full duplex*[2] folosind o arhitectură cu un singur dispozitiv principal (master), acesta fiind cel care comandă intervalul de timp în care se scrie și se citește, la care se pot conecta mai multe dispozitive secundare (slave). Protocolul este unul cu 4 fire : SCLK – semnal de tact; MISO – linie de date de intrare pentru dispozitivul principal (master) și de ieșire pentru dispozitivul secundar (slave); MOSI – linie de date de ieșire pentru master și de intrare pentru slave ; SS – linie pentru selecția dispozitivului slave cu care vrea să comunice dispozitivul master [18].

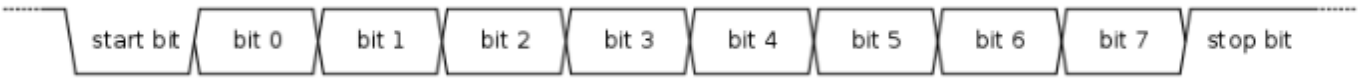
Pentru acest proiect am ales conectarea într-o configurație cu linii de selecție independente pentru fiecare dispozitiv secundar (fig. 31) deaorece astfel se pot culege separat informațiile de la modulele cu joystick cu o frecvență satisfăcătoare de cel puțin 25 citiri pe secundă.



Sursa: en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Figura 31: Modalitate de conectare a mai multor dispozitive prin SPI

Interfața UART (Universal Asynchronous Receiver/Transmitter) este o interfață serială asincronă configurabilă, în care perioada de transmitere sau recepție a unui mesaj nu este sincronizat prin intermediul unui semnal de tact între dispozitive, în schimb mesajul conține informație sub forma unor biți de start și stop. Și în acest caz transmisia poate fi *full duplex* , dar este esențial ca ambele dispozitive să aibă aceleași configurații pentru viteza de comunicație (numită și baud rate sau bit speed), lungimea caracterului, tipul de paritate și numărul de biți de stop , pentru ca dispozitivele să poată comunica corect [19].



Sursa: en.wikipedia.org/wiki/Universal_asynchronous_receiver

Figura 32 : Exemplu de mesaj transmis prin UART

[2] Full duplex= regim de funcționare în comunicație în care două dispozitive conectate pot transmite și recepționa informație în același moment de timp

În schema bloc elementele denumite **Pmod JSTK2** reprezintă module electronice cu joystick produse de firma Digilent. Acest modul dispune de un joystick rezistiv cu 2 axe, un buton încorporat în joystick, un buton suplimentar și un LED multicolor rgb capabil să ofere 2^{24} culori. Pentru citirea valorilor de la joystick, butoane, comanda LED multicolor dar și pentru a realiza comunicația prin SPI cu alte dispozitive, modulul dispune de un microcontroler PIC16F1618 pre-programat ce realizează funcțiile anterior prezentate. Aceste module funcționează la o tensiune de alimentare de 3.3V și consumă, atunci cand nu e aprinsă dioda led și nu sunt apăsate butoanele, 4.85mA [20].

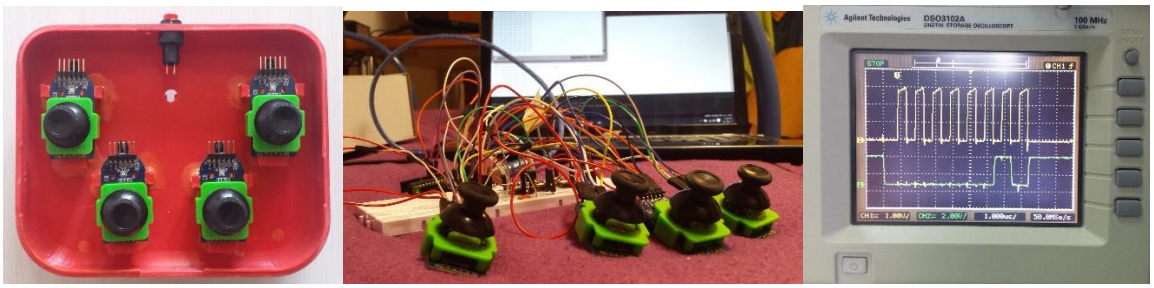
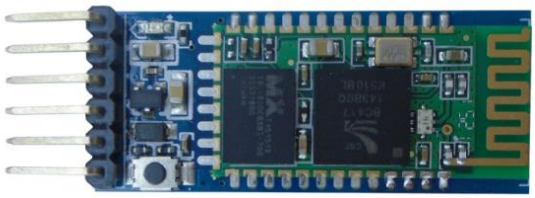


Figura 33: Module Pmod JSTK2 în timpul testării

Funcția de comunicație fără fir între telecomandă și robot este îndeplinită de **modulul HC05**. Acesta este un modul electronic ce utilizează tehnologia de comunicație fără fir Bluetooth 2.0 și comunică prin fir prin UART. Modulul are integrat un regulator liniar de tensiune ce permite alimentarea cu tensiuni cuprinse între 6V și 12V; el poate funcționa în mai multe moduri, spre exemplu modul de dispozitiv de interfață cu utilizatorul denumit în engleză Human Interface Device (de exemplu mouse, tastatură, căști) , dar în cadrul acestui proiect modul de funcționare folosit se numește Serial Port Profile (SPP) sau profil de port serial. În acest mod de funcționare modulul se poate conecta la alt modul cu tehnologie Bluetooth 2.0 și 2 astfel de dispozitive conectate pot forma o conexiune UART ca și cum ar fi conectate prin fire (dar nu sunt) .

Modulul dispune și de o ieșire numită *Stare conexiune* ce indică, așa cum sugerează numele, starea de conexiune în care se află modulul: 1 logic = conectat ; 0 logic = deconectat.



Sursa: aliexpress.com

Figura 34: Modul HC05

Acest modul a fost configurat să se auto-conecteze la celălalt modul HC05 din robot, criteriul de conectare fiind adresa fizică a dispozitivului și un pin format din 4 cifre care trebuie să fie identic pentru ambele dispozitive. Viteza de comunicație este de 115200 bit/s , mesaje de 8 bit, paritate de tip N și 1 bit de stop.

Este foarte util ca la o telecomandă să existe informații despre starea conexiunii cu dispozitivul comandat, pentru acest scop telecomanda are un LED bi-color cu catod comun ce se aprinde roșu atunci când nu există conexiune cu robotul și verde atunci cand există o conexiune. Led-ul nu poate fi comandat direct de către semnalul stare conexiune și astfel a fost introdus un circuit pentru comanda acestuia.

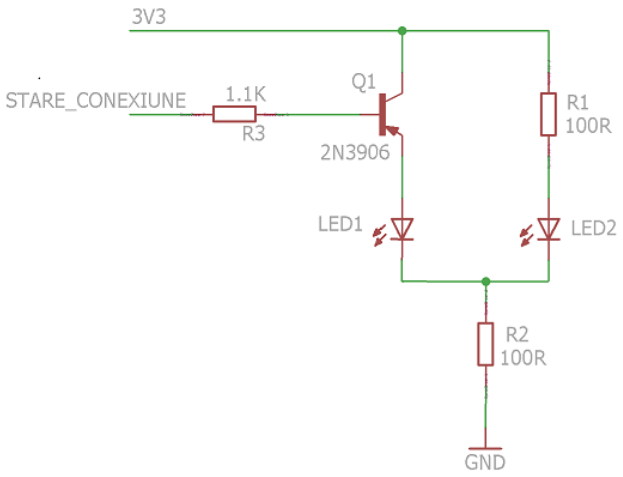


Figura 35: Circuit de comandă pentru LED bicolor cu catod comun

Funcționarea circuitului este simplă, la pornire linia STARE_CONEXIUNE este la 0V iar tranzistorul Q1 conduce și șuntează ramura cu dioda LED2 (culoarea verde) , astfel LED1 (culoarea roșie) luminează puternic. În acest moment se aprinde și LED2, dar insesizabil. În momentul conectării modului HC05, linia STARE_CONEXIUNE trece la 3.3V iar tranzistorul Q1 se blochează; acum conduce numai LED2 (culoarea verde) iar LED1 este complet stins.

Aproximare a curentului consumat:

- din foaia de catalog a Pmod JSTK2 [20], găsim ca unul consumă 5mA, ceea ce rezultă un consum total de $4 * 5mA = 20mA$
- din foaia de catalog a microcontrolerului atmega328 [21], găsim un consum de 2.5mA
- din foaia de catalog a circuitului de conversie usb-serial [22], găsim un consum de 15mA
- am eliminat de pe modul cele 4 diode LED pentru a reduce consumul de curent
- din foaia de catalog a modulului HC05 [23] ,găsim un consum de 40mA în regim de căutare a unei conexiuni
- led-ul bicolor consumă 20mA

În total telecomanda are un consum de 97.5mA. Am ales baterii de tip Li-Ion reîncărcabile cu o capacitate de 3300mAh, conectate în serie pentru a furniza o tensiune între 6.6V și 8.4V în funcție de starea de încărcare. La baterii se conectează direct numai modulul Arduino nano și modulul HC05 (linie roșie îngroșată în figura 30) , celelalte module fiind alimentate prin regulatorul de tensiune de 3.3V al modulului Arduino (linie roșie subțire în figura 30). La consumul estimat telecomanda poate să funcționeze continuu, în regim de consum de curent maxim, timp de 34 ore.

3.1.2) Subsistemul robot

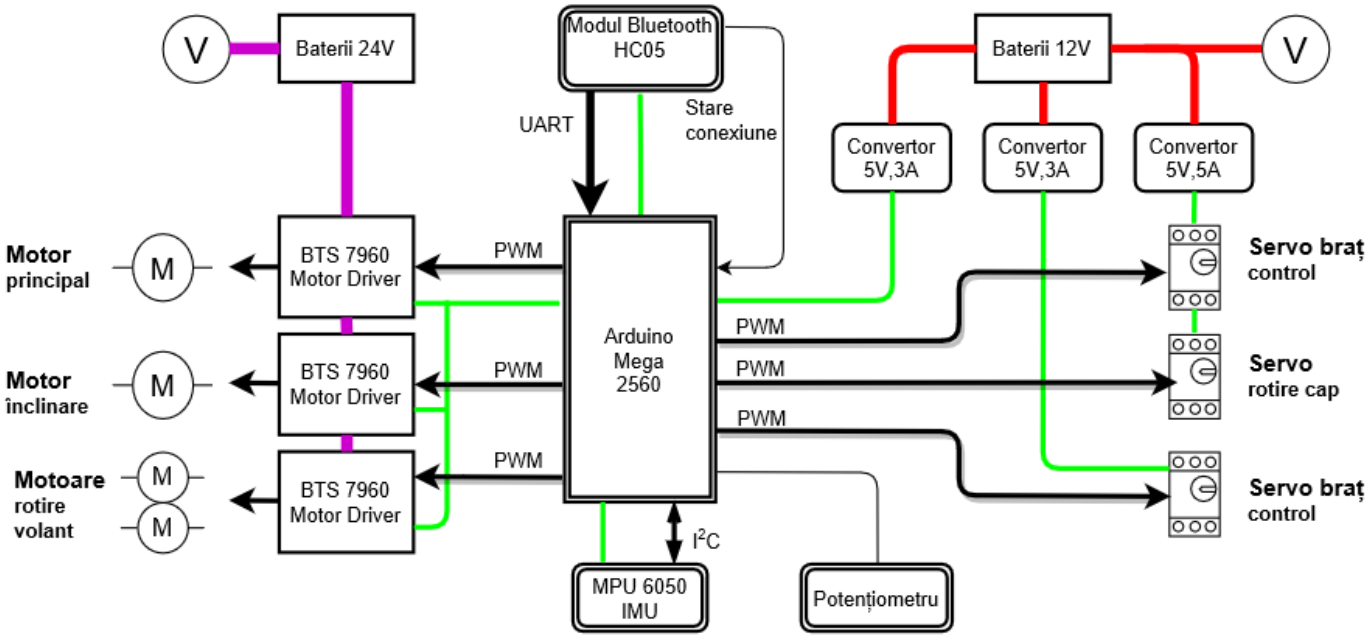


Figura 36: Schema bloc electronică a robotului

Robotul fiind în mărime naturală, are nevoie de motoare suficient de puternice pentru acționarea lui, iar o soluție foarte convenabilă din punct de vedere al raportului preț-performanță este motorul folosit în mod normal la ștergătoarele de parpriz de la autovehicule. Acestea sunt proiectate să funcționeze și la tensiuni de 24V, iar cuplul maxim dezvoltat la această tensiune, cu un curent absorbit de până la 15A, este de 50kg-cm datorită . Unitatea kg-cm nu este una standard, dar este des folosită în practică și reprezintă echivalentul a 0,098Nm [3]. Un astfel de motor am utilizat pentru acționarea principală față-spate și pentru înclinarea stânga-dreapta. Pentru rotirea volantului am ales 2 motoare de curent continuu cu perii, cu reducție, ce funcționează la 24V și o turație 300 rot/min . Curentul absorbit de acest motor cu axul blocat este de 800mA.

Sursa: cdn3.volusion.com ; optimusdigital.ro ; hdimagegallery.net

Figura 37: Motorul principal, bateria Turnigy 3S 5Ah și motorul volantului

Ca sursă de energie am ales 4 baterii în tehnologie litiu-ion cu electrolit de tip polimer deoarece acestea au energia specifică foarte mare și de asemenea un curent de descărcare mare [24]. Modelul ales este Turnigy 3S 5Ah, cu o capacitate de 5000mAh, 3 celule în serie ce oferă o tensiune între 9.9V

[3] Nm= newton-metru , unitatea de măsură pentru cuplu în sistemul internațional

și 12.6V . Două baterii vor fi conectate în serie pentru a crea blocul „Baterii 24V”, iar celelalte vor fi conectate în paralel pentru a crea blocul „Baterii 12V” din schema bloc electronică a robotului. Deoarece bateriile Li-Po pot fi distruse iremediabil prin descărcare excesivă (dacă tensiunea unei celule scade sub 3V ea nu va mai putea fi utilizată) am conectat la fiecare din cele 2 seturi de baterii câte un voltmetru pentru afișarea stării de încărcare a bateriilor.

Circuitele de comandă pentru motoarele de putere sunt module electronice realizate cu circuitul integrat **BTS 7960.** Modulul conține și un circuit buffer 74HCT244 cu rol de protecție între partea de control cu nivele logice de 0V și 5V și partea de putere ce utilizează tensiuni de până la 24V. Acest circuit asigură că la modulul de comandă, în acest caz Arduino mega 2560, nu pot ajunge tensiuni dăunătoare.

Circuitul BTS 7960 conține o semipunte și circuite de protecție pentru aceasta precum:

- protecție la subtensiune, pentru a evita mișcări necontrolate ale motorului comandat. Dacă tensiunea de alimentare a circuitului de putere scade sub 5.4V, atunci circuitul se oprește automat și va reporni doar după ce tensiunea crește peste 5.5V
- protecție la temperaturi ridicate, prin intermediul unui senzor de temperatură integrat, pragul de oprire este de 150 ˚C la nivelul joncțiunii din siliciu
- protecție la supracurent, curenții prin ambele tranzistoare ale semipunții sunt monitorizați, iar un curent mai mare de 44A are ca efect blocarea tranzistoarelor

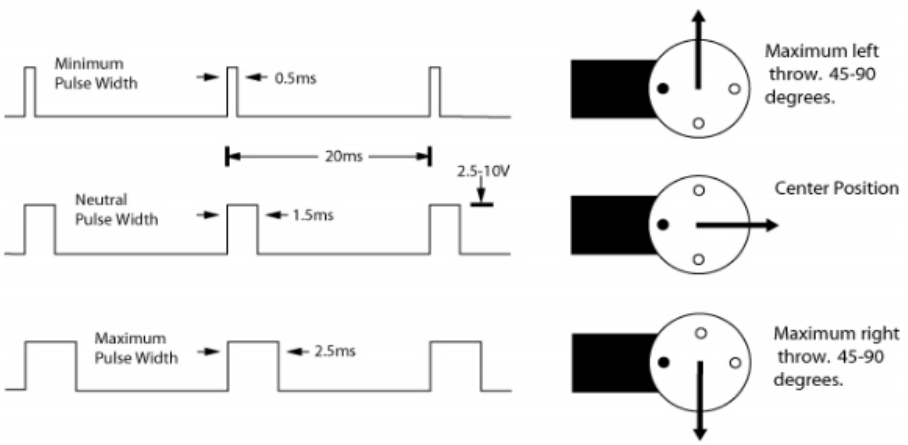
Semnalul de comandă este un semnal dreptunghiular modulat în lățime de impuls (PWM) cu frecvența maximă de 24kHz [25], iar microcontrolerul utilizat pentru acest proiect furnizează un semnal cu frecvența de 490 Hz.

Sursa: ebay.com ; hobbyking.com ș banggood.com

Figura 38: Modulul BTS7960 , servomotor HK15338, servomotor MG958

Pentru comanda poziției brațului de control al capului am ales servomotoare digitale de putere de tip **HK15338** , ce pot fi comandate direct cu semnale logice, alimentate la 5V ce dezvoltă un cuplu maxim de 20 kg-cm la un curent absorbit de 2.5A. Pentru mișcarea de rotație a capului am ales un servomotor digital de tip **MG958**, ce poate fi comandat direct cu semnale logice, alimentat la 5V ce dezvoltă un cuplu maxim de 10kg-cm la un curent de 1A.

Comanda servomotoarelor se face tot prin semnal dreptunghiular modulat în lățime, astfel:



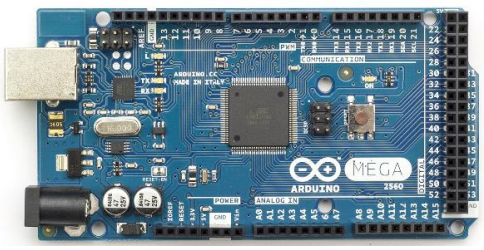
Sursa: mitchr.me

Figura 39: Semnalul de comandă pentru servomotoare

Observăm că frecvența semnalului este fixată la 50Hz (perioadă de 20ms) , iar lățimea impulsului se modifică între 0.5 ms și 2.5ms. Valoarea de mijloc este de 1.5ms și aceasta comandă de obicei poziția de 0˚ sau viteza de 0 rot/min în cazul servomotoarelor cu rotație continuă. Pentru intervale de 0.5ms sau 2.5ms servomotorul este comandat spre pozițiile extreme sau vitezele maxime, în cazult servomotoarelor cu rotație continuă.

Comunicația fără fir între robot și telecomandă se realizează prin module HC05 ce utilizează tehnologia Bluetooth 2.0. Modulele și modul de comunicație au fost prezentate în capitolul 3.1.1) ,iar motivul principal pentru alegerea tehnologiei bluetooth este compatibilitatea cu telefoanele mobile inteligente (smartphone), astfel în viitor telecomanda ar putea fi înocuită cu o aplicație pe telefon.

Modulul central, care controlează toate cele menționate mai sus, este placa de dezvoltare cu microcontroler **Arduino Mega 2560**. Aceasta este realizată cu atmega2560, un microcontroler cu arhitectură RISC avansată pe 8 biți, cu 54 de pini de intrare/ieșire, dintre care 16 pini pot avea funcție de citire analogică și 15 pini pot furniza semnal dreptunghiular modulat în lățime de impuls. Microcontrolerul are de asemenea și 4 periferice de tip interfață serială UART și periferic de tip interfață serială I^2C. Cu o frecvență de tact de 16MHz, acest microcontroler este suficient de performant pentru a comunica cu toate modulele si pentru a implementa toate cele 3 sisteme de conducere locală a robotului.

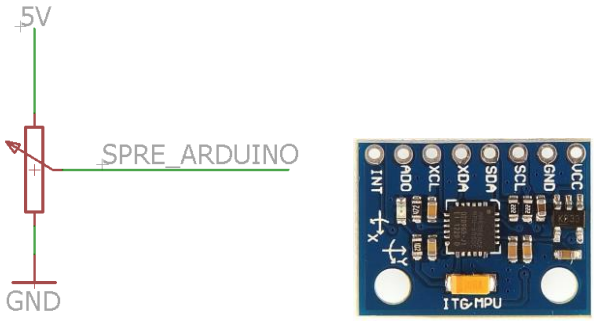


Sursa: arduino.cc

Figura 40: Arduino Mega 2560

Pe post de traductor pentru sistemele de conducere locală este un potențiometru ce măsoară unghiul dintre mecanismul de înclinare și căruciorul principal, care este de fapt unghiul dintre mecanismul de înclinare și corpul robotului. Modalitatea de conexiune este simplă, cele 2 capete ale potențiometrului sunt conectate la tensiunea de alimentare de 5V, iar cursorul este conectat la una din intrările cu posibilitate de conversie analog-digitală ale modulului Arduino Mega2560.

Celălalt traductor este o unitate de măsurare inerțială (IMU) realizată cu circuitul integrat **MPU6050**. Acesta conține senzori *MEMS*[4] de tip accelerometru și giroscop cu 3 axe de măsurare fiecare, un senzor de temperatură, un periferic principal de tip interfață de comunicare I^2C, un periferic secundar de tip interfață de comunicare I^2C ce permite conectarea altor senzori, de exemplu magnetometru și un procesor denumit de producător Digital Motion Processor (DMP) ce poate fi configurat pentru ca modulul să citească valorile pentru accelerații liniare și accelerații unghiulare periodic, să le transforme în unul sau mai multe dintre mărimile de ieșire și să le încarce într-o locație de memorie de unde pot fi citite de către un microcontroler extern. Printre mărimile ce se pot obține se numără unghiuri euler, unghiuri pentru tangaj,ruliu și derivă. Această funcție a circuitului MPU6050 poate ușura funcționarea microcontrolerului de sistem [26].



Sursa: blog.bitify.co.uk

Figura 41: Modul de conectare al potențiometrului și modulul MPU6050

Interfața I^2C(Inter-Integrated Circuit) este o interfață serială de comunicație sincronă pe 2 fire, utilizată pe distanțe mici, la care se pot conecta mai multe dispozitive principale(master) și mai multe dispozitive secundare (slave). Protocolul funcționează în regim *half duplex*[5], cele 2 linii utilizate sunt bidirecționale și sunt conectate la dispozitive cu *ieșirea cu drenă deschisă*[6], una din linii se numește SDA(Serial Data Line) și este utilizată la transferul de date, iar cealaltă linie se numește SCL (Serial Clock Line) și este linia de tact utilizată în comunicare. Fiecare dispozitiv are propria lui adresă ceea ce înseamnă că un mesaj poate fi transmis doar dacă se cunoaște adresa receptorului și a

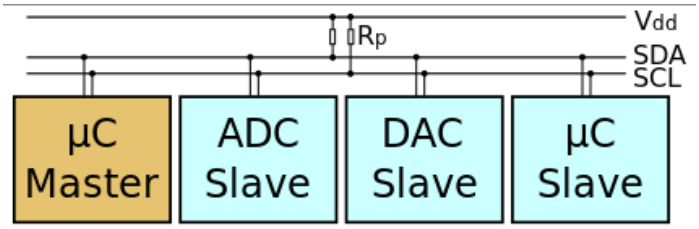
[4] *MEMS*= microsistem electro-mecanic

[5] *Half duplex*= regim de funcționare în comunicație în care două dispozitive conectate pot transmite și recepționa informație dar nu în același moment de timp.

[6] *Ieșire cu drenă deschisă*=circuit al cărui ieșire poate impune un nivel logic 0 (LOW), dar nu poate impune un nivel logic 1 (HIGH)

destinatarului.Fiecare dispozitiv monitorizează magistrala și nu scrie până nu se termină transmisia curentă de pe magistrală, se poate totuși ca două dispozitive să încerce să scrie pe magistrală în același timp, atunci are loc o arbitrare nedistructivă, iar dacă un dispozitiv constată că ceea ce a scris pe magistrală nu este identic cu ceea ce a citit de pe magistrală, atunci el oprește transmisia până când magistrala este liberă din nou [27].

Importanța interfeței este dată de posibilitatea de a conecta, cu doar 2 fire, un număr de până la 1008 dispozitive [28].

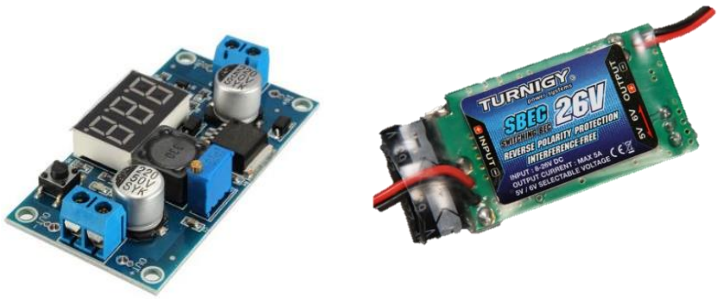


Sursa: wikipedia.org/wiki/I²C

Figura 42: Exemplu de conexiune I^2C

Pentru partea de conversie a tensiunii de la baterii am ales convertoare de curent continuu DC-DC în comutație datorită randamentului crescut al acestor module. În cadrul schemei bloc (figura 36) liniile de culoare verde reprezintă liniile de alimentare de 5V, cele roșii de 12V, iar cele violet de 24V. Deoarece consumul de curent este mare, de până la 15A pentru motoarele de putere, trebuie ținut cont de diametrul firelor de alimentare și al firelor de legătură cu motorul. La sursa de putere de 24V se conectează numai motoarele pentru mișcarea față-spate, stânga-dreapt și pentru rotirea volantului iar curentul total maxim absorbit de acestea este de $15A + 15A + 0.8A + 0.8A = 31.6A$. Firele alese pentru aceste conexiuni sunt din cupru cu aria secțiunii transversal de $6mm^2$, ce pot conduce un curent de până la 40A. Cele 2 motoare de ștergător de parbriz au fiecare fire cu secțiunea de $1{,}5mm^2$ ce suportă 25A, iar motoarele volantului au fiecare fire cu secțiunea de $0.8mm^2$, capabile să transporte 14A [29].

Firele de conexiune între sursa de 12V și convertoarele de tensiune DC-DC au și ele secțiunea de $0.8mm^2$, la fel și firele ce conectează servomotoarele la convertoare. Pentru liniile de semnal și pentru firele de alimentare ale celorlalte module am folosit fire cu diametru mai mic deoarece curenții vehiculați prin acestea sunt foarte mici, de ordinul zecilor de mA sau mai puțin.



Sursa: banggood.com ; hobbyking.com

Figura 43: Convertor 5V 3A, convertor 5V 5A

3.2)Software

Unul dintre motivele pentru care am ales module din familia Arduino pentru realizarea funcțiilor de comandă și control este compatibilitatea software-ului necesar pentru programarea microcontrolerelor.

Arduino este o companie ce dezvoltă și produce module electronice și programe de calculator cu licența GPL (GNU General Public License), adică pun la dispoziție în mod gratuit codul sursă pentru toate programele și proiectul ce include schme electronice și fișiere de fabricație pentru toate modulele produse prin intermediul internetului. Prin această decizie, compania permite oricui modificarea, producția și distribuția de module electronice și programe Arduino.

Microcontrolerele modulelor Arduino pot fi programate în orice limbaj de programare atâta timp cât există un compilator ce poate transforma programul în cod mașină pentru microcontrolerul-țintă. Compania Arduino pune la dispoziție un mediul de dezvoltare Arduino, compatibil cu mai multe platforme (Windows, Linux, MacOS), ce conține un editor de text și unelte ușor de utilizat pentru compilarea codului și programarea microcontrolerului-țintă cu codul mașină rezultat în urma compilării.

Mediul Arduino suportă limbajele de programare C și C++ utilizând reguli speciale de scriere și are incluse biblioteci software pentru funcții de intrare și ieșire. Codurile scrise de utilizatori trebuie să conțină 2 funcții: funcția de inițializare și funcția principală ce se execută repetitiv, în buclă [30].

3.2.1) Codul telecomenzii

Constantele de timp ale sistemului mecanic sunt în acest caz, dar și în general, mult mai mari decât cele ale circuitelor electronice; din acest motiv am ales o frecvență de transmisie de 25Hz. Pentru ca asteastă frecvență să fie respectată e nevoie de o temporizare, iar aceasta e realizată cu ajutorul funcției millis(). Funcția millis() este inclusă în pachetul standard de biblioteci de funcții al mediului de dezvoltare Arduino și returnează numărul de milisecunde trecute de la momentul alimentării modulului sau de la ultimul reset. Tipul de dată returnat este unsigned long, ce permite exprimarea numerelor pozitive pe 32 biți, deci se pot exprima numerele de la 0 la 4,294,967,295 ($2^{32} - 1$). Acest număr maxim poate fi atins după aproximativ 50 zile de funcționare fără reset și trebuie ținut cont de acest fapt pentru aplicațiile cu durată de viață mai mare de 50 zile.

Mod de funcționare : la intrarea în buclă se citește timpul curent prin instrucțiunea *currentMillis = millis()* , iar apoi se verifică dacă diferența de timp dintre momentul curent și ultimul moment de timp la care s-au executat instrucțiunile dorite depășește un interval prestabilit, în acest caz cele 40ms corespunzătoare frecvenței de 25Hz, prin instrucțiunea *if (currentMillis - previousMillis >= interval).* Dacă este depășit intervalul, auntci se execută din nou instrucțiunile, iar vechiul moment de timp devine momentul actual de timp : *previousMillis = currentMillis* .

Telecomanda este programată să nu transmită date spre modulul Bluetooth înainte ca acesta să fie conectat, aceasta se realizează prin funcția conectare:

```
void conectare(){
        while(state == 0)
        {
        state = digitalRead(2);
        }
}
```

Funcția (int) digitalRead(2) returnează nivelul logic de la pinul 2 al modulului; la acest pin este conectată ieșirea de stare a modului Bluetooth. State este o variabilă ce memorează starea conexiunii modulului Bluetooth, iar atâta timp cât modulul este deconectat, microcontrolerul așteaptă și nu execută alte instrucțiuni.

Citirea valorilor de la modulele PmodJstk2 se face prin intermediul interfeței SPI și pentru aceasta am utilizat biblioteca SPI ce se poate găsi la adresa *github.com/codebendercc/arduino-library-files/tree/master/libraries/SPI*. Prin instrucțiunea *SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0))* se stabilesc parametrii de comunicație pentru SPI și anume : frecvența de tact de 1MHz, modul de transmisie cu cel mai semnificativ bit transmis prima oară și modul SPI0 în care datele sunt citite pe frontul crescător al semnalului de tact și își modifică valoarea la frontul descrescător al semnalului de tact [18].

Din foaia de catalog a PmodJstk2 [20] găsim că acesta răspunde cu valorile de la joystick și de la butoane dacă primește comanda 0xC0. Între transferul mai multor *bytes*[7]de informație e nevoie de o pauză de cel puțin 10 μs, cerință impusă de modulul PmodJstk2, iar pentru această aplicație pauza este fixată la 100 μs.

```
c1=SPI.transfer (0xC0);
delay(0.1);
c2=SPI.transfer (0x00);
delay(0.1);
c3=SPI.transfer (0x00);
delay(0.1);
c4=SPI.transfer (0x00);
delay(0.1);
c5=SPI.transfer (0x00);
delay(0.1);
c6=SPI.transfer (0x00);
```

În urma acestor instrucțiuni în vairabila c6 se găsește valoarea pentru axa X, în variabila c1 se găsește valoarea pentru axa Y, iar în variabila c5 se găsește informația pentru butoane.

[7] Byte= unitate de informație digitală formată din 8 biți

Transmiterea valorilor către modulul Bluetooth se face prin interfața serială cu ajutorul funcției Serial.print(), inclusă și ea în bibliotecile standard ale mediului de dezvoltare Arduino. Este nevoie de o inițializare a interfeței seriale a priori apelării funcției Serial.print() și aceasta se realizează prin comanda *Serial.begin(115200)*. Cu această comandă se impune și viteza de transfer de 115200 bit/s.

Codul complet se găsește în Anexa 4.

3.2.2) Codul robotului

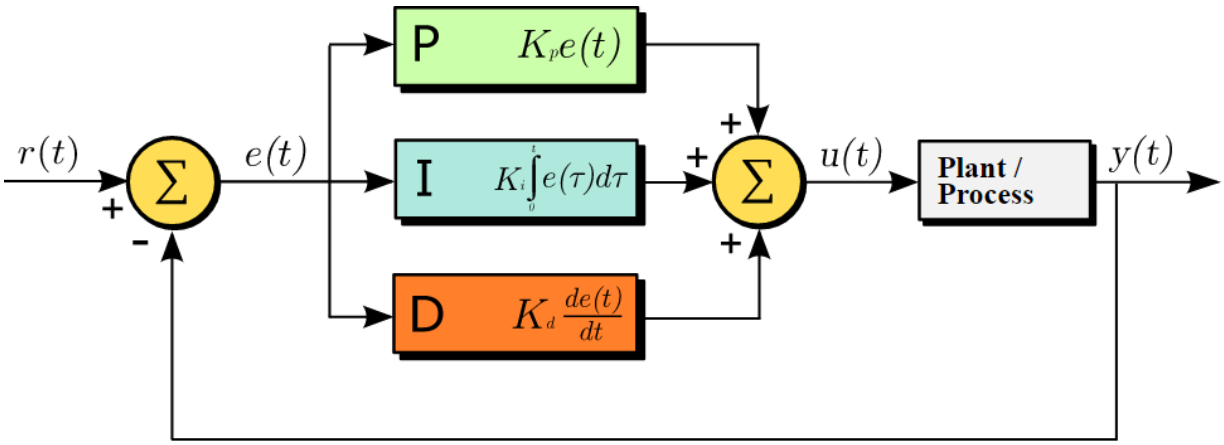
Ca algoritm de reglare numerică pentru sistemele de conducere locală ale robotului am ales PID (Proportional Integrativ Derivator) deoarece este cel mai utilizat în practică și funcționează bine în majoritatea situațiilor. Există pentru mediul de dezvoltare Arduino o bibliotecă specială ce ajută la implementarea unui algoritm PID; aceasta este disponibilă gratuit la adresa *http://playground.arduino.cc/Code/PIDLibrary*. Algoritmul PID pe care se bazează această bibliotecă se numește PID paralel [31] și este descris de următoarea relație:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt} \qquad (5)$$

Unde Kp,Ki, Kd sunt toți pozitivi și reprezintă cei 3 coeficienți: proporțional, integrativ și derivator. În relația (5) semnalul u(t) se numește mărimea de comandă, iar e(t) se numește semnal de eroare și are expresia:

$$e(t) = r(t) - y(t) \qquad (6)$$

Semnalul r(t) este mărimea de prescriere și y(t) este mărimea măsurată sau mărimea de reacție. Schema bloc a controlerului se găsește în figura următoare:



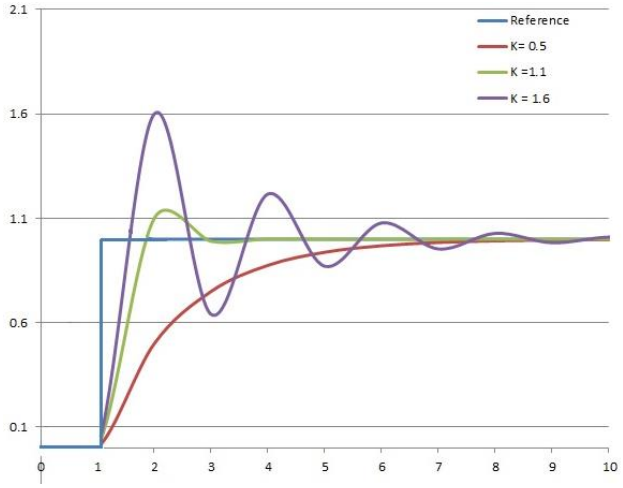
Sursa: en.wikipedia.org/wiki/PID_controller

Figura 44: Schema bloc PID paralel

Componenta P se spune că ține cont de valoarea prezentă deoarece mărimea de ieșire dată de componenta P este direct proporțională cu valoarea curentă a erorii. Răspunsul poate fi ajustat prin modificarea coeficientului Kp, numit și câștig proporțional. O valoare mare a câștigului proporțional dă o modificare mare a ieșirii pentru o modificare a erorii dată, iar dacă aceasta este prea mare atunci sistemul poate deveni instabil. Pe de cealaltă parte, o valoare prea mică duce la un răspuns al ieșirii

insuficient pentru o valoare mare a erorii și la un controler cu sensibilitate scăzută. Pentru a funcționa, controlerul de tip P are nevoie de o eroare nenulă, în general acesta funcționează cu o așa-numită eroare de stare staționară ce este proporțională cu câștigul procesului controlat și invers proporțională cu câștigul proporțional al controlerului. Pentru anularea erorii de stare staționară se poate compensa prin adăugarea unei constante mărimii de prescriere sau prin adăugarea unui termen integrativ.

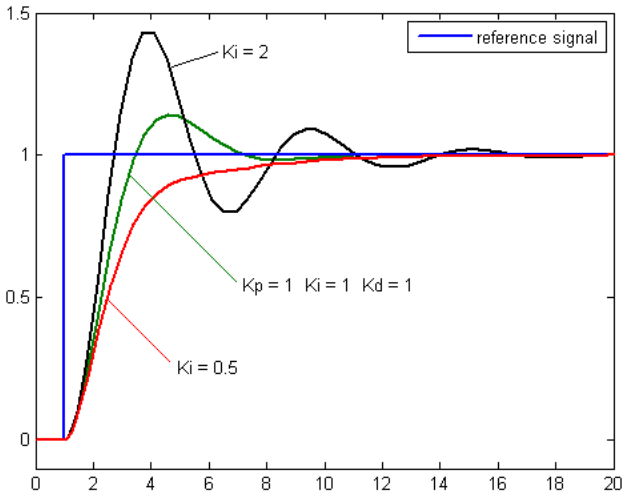
În figura următoare se prezintă comportamente ale controlerului pentru diferite valori ale câștigului proporțional Kp:



Sursa: en.wikipedia.org/wiki/PID_controller

Figura 45: Răspunsul unui controler pentru diferite valori ale Kp

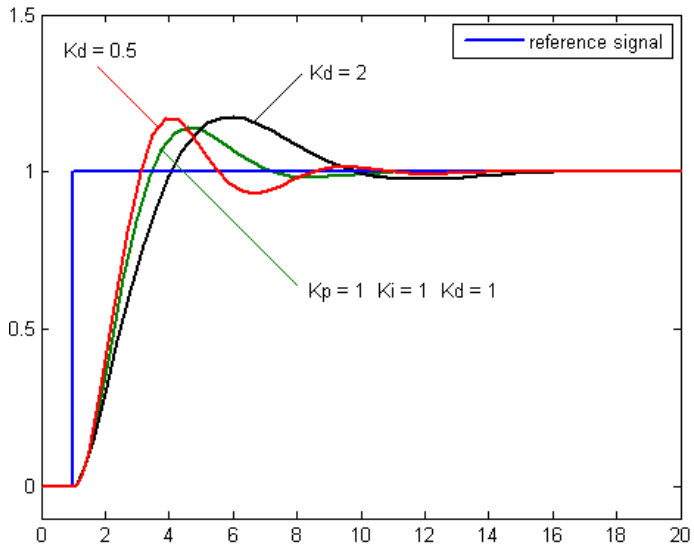
Componenta I ține cont de valorile din trecut ale erorii, dacă valoarea curentă a ieșirii nu este suficient de mare, integrala erorii va acumula în timp o valoare suficient de mare pentru mărimea de comandă. Contribuția integratorului este direct proporțională cu amplitudinea erorii și cu durata acesteia, eroarea cumulată în timp este înmulțită cu câștigul integral Ki și adăugată la mărimea de comandă, astfel componenta I elimină erorile de stare staționară. Deoarece ieșirea componentei I răspunde la acumulări ale erorii din trecut, dacă câștigul integral este prea mare, el poate face ca valoarea curentă a ieșirii să depășească mărimea de prescriere.



Sursa: en.wikipedia.org/wiki/PID_controller

Figura 46: Răspunsul unui controler pentru diferite valori ale Ki

Componenta D ține cont de posibile valori viitoare ale erorii, în funcție de valoarea curentă a vitezei de variație a erorii. Spre exemplu, considerând controlerul P, dacă o mărime de comandă pozitivă de valoare mare aduce cu succes procesul în starea dorită, aceasta are și tendința de apune procesul pe o traiectorie ce va da naștere în viitorul imediat a unei erori negative. În acest caz, la o scădere mare a erorii, ieșirea componentei D devine negativă și reduce din mărimea de comandă pentru a preveni fenomenul de supra-compensare. Mărimea de ieșire a componentei D este dată de produsul dintre viteza de variație a erorii și câștigul derviativ Kd și in general îmbunătățește stabilitatea sistemului.



Sursa: en.wikipedia.org/wiki/PID_controller

Figura 47: Răspunsul unui controler pentru diferite valori ale Kd

Foarte important la un controler PID este metoda de determinare a coeficienților Kp, Ki și Kd. Foarte important la determinarea coeficienților este dacă procesul poate fi trecut în modul *offline*, adică să poată fi deconectat de restul sistemului. În acest caz, cele mai bune metode de reglare implică în general aplicarea unui semnal traptă la intrare și măsurarea răspunsului în timp al ieșirii.În tabelul următor sunt prezentate avantajele și dezavantajele fiecărei metode [32] :

Metodă	Avantaje	Dezavantaje
Reglare manuală	Nu e nevoie de matematică, se poate realiza online	E nevoie de expreiență
Ziegler-Nichols	Metodă dovedită, se poate realiza online	E nevoie de încercare și eroare, reglare agresivă
Tyreus Luyben	Metodă dovedită, se poate realiza online	E nevoie de încercare și eroare, reglare agresivă
Instrumente Software	Online sau offline, permit simularea sistemului	Costă bani și e nevoie de formare
Cohen-Coon	Modele foarte bune ale procesului	E nevoie de matematică, offline și doar pentru sisteme de ordinul 1

Tabel 1: Metode de reglare PID

Pentru acest proiect reglarea parametrilor se va face manual, iar implementarea se observă în următoarele linii de cod:

```
//declarație PID
double Pk2 = 2.8;
double Ik2 = 0;
double Dk2 = 0;
double Setpoint2, Input2, Output2, Output2a;
 PID PID2(&Input2, &Output2, &Setpoint2, Pk2, Ik2 , Dk2, DIRECT);
// setup PID
PID2.SetMode(AUTOMATIC);
PID2.SetOutputLimits(-255, 255);
PID2.SetSampleTime(20);
// actualizare marime de prescriere
Setpoint2 = map(ch6, 0,255,-45,45);          // ch6 reprezinta un canal de la telecomanda
// calcul mărime de comandă
 PID2.Compute();                              // mărimea de comandă se va găsi în variabila Output2
```

Comanda motorului principal, pentru deplasarea față-spate, se face ca în teorie: mărimea de prescriere este primită de la telecomandă și reprezintă unghiul căruciorului din mecanismul intern; mărimea de ieșire a controlerului PID este transmisă către electronica de putere ce comandă motorul, iar mărimea de reacție este unghiul citit de IMU. În acest caz controlerul va comanda motorul astfel încât mărimea de reacție să se apropie cât mai mult de mărimea de prescriere, cu alte cuvinte semnalul de eroare să fie cât mai mic.

Comanda motorului de înclinare a robotului este puțin mai specială și se realizează prin 2 regulatoare PID. Unul din regulatoare are același rol ca și regulatorul motorului principal, descris mai sus. Necesitatea celui de-al doilea regulator este dată de faptul că în realitate mărimea ieșire a uni motor de curent continuu nu este un unghi, ci o viteză. Electronica de putere primește un semnal modulat în lățime de impuls la intrare și comandă motorul cu o valoare medie a tensiunii de ieșire direct proporțională cu factorul de umplere al semnalului de intrare, iar la un motor de curent continuu se știe că viteza de rotație a motorului este direct proporțională cu valoarea medie a tensiunii aplicate motorului. Este adevărat că în cazul motorului principal viteza de rotație a motorului determină mișcarea căruciorului, care la rândul său dă naștere mișcării de rostogolire. Dacă viteza de rotație a motorului este bine aleasă, atunci va genera o rostogolire continuă în cadrul căreia căruciorul va fi orientat la un unghi constant (dacă și viteza de rostogolire este constantă). Pentru direcția stânga-dreapta rostogolirea continuă este imposibilă datorită geometriei mecanismului intern, astfel mărimea de ieșire a motorului de încliare trebuie să fie un unghi.

Tipul de motor ce îndeplinește această condiție este servomotorul, iar pentru obținerea unui comportament de servomotor e nevoie de un controler separat ce îndeplinește această funcție. Acest controler (PID 1) are ca mărime de prescriere unghiul dat de controlerul superior (PID 2), mărimea de ieșire comandă direct motorul de înclinare, iar mărimea de reacție este unghiul dintre corpul robotului și mecanismul de înclinare, măsurat de potențiometru. Această funcționare este schițată în figura următoare:

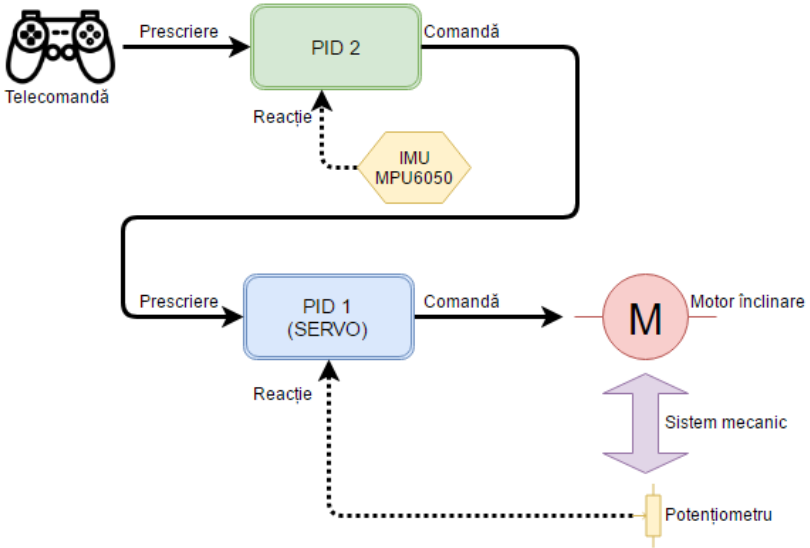


Figura 48: Schema de comandă a motorului de înclinare

Un aspect foarte important, în special în acest caz datorită dimensiunilor robotului, este siguranța operatorului uman și a tuturor ce se află în vecinătatea robotului. Pentru a evita situațiile periculoase, robotul este programat să funcționeze doar atunci când operatorul are control total. În caz contrar, robotul nu va funcționa deloc. Un nivel de siguranță este asigurat de comutatoarele fizice de *validare*[8] a circuitelor de comandă pentru motoare BTS7960, dar aceste comutatoare sunt situate în robot în apropierea circuitelor respective iar atunci cand robotul va fi complet din punct de vedere estetic (cu panourile laterale fixate) accesul la acele comutatoare va fi blocat. Pentru a da totuși control operatorului asupra stării de validare sau invalidare a motoarelor, comutatoarele nu au fost conectate direct la nivelul logic necesar validării, ci la ieșirile microcontrolerului ce au fost programate să atingă un nivel logic necesar validării dacă și numai dacă robotul este conectat la telecomandă și unul din comutatoarele telecomenzii este trecut în poziția de validare. Dacă spre exemplu robotul se conectează la telecomandă, dar mesajele transmise de telecomandă sunt corupte atunci robotul va invalida motoarele pentru că nu primește corect mesajul de validare generat de unul din comutatoarele telecomenzii. Această funcție este realizată de următoarea secvență de cod:

```
BTstate = digitalRead(33);
    if(but3 == 0 && BTstate == 1) {
      digitalWrite(31, HIGH);
      }
    else if (but3 == 1 || BTstate == 0) {
      digitalWrite(31, LOW);
    }
```

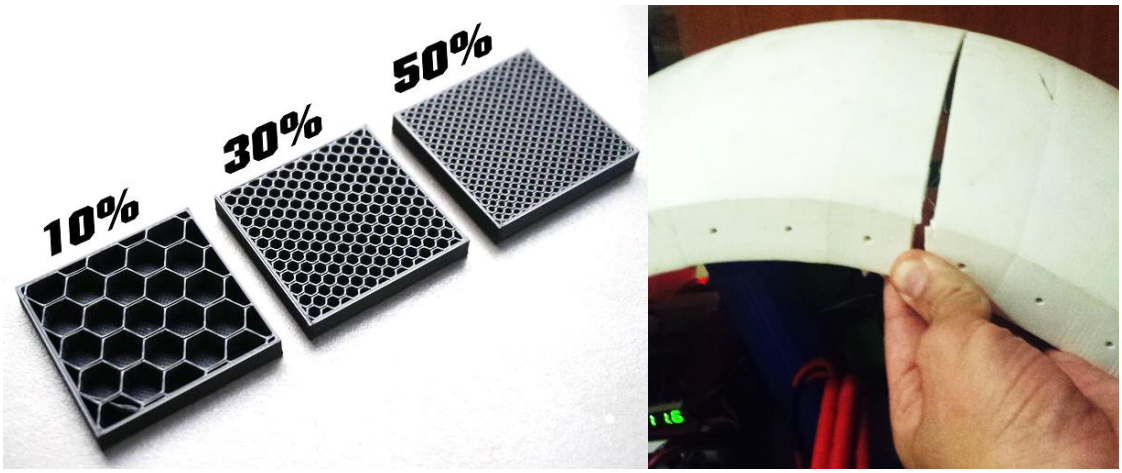
Codul complet se găsește în Anexa 5.

[8] *Validare*= în electronică o intrare de validare a unui circuit poate opri activarea ieșirii acelui circuit și implicit dezactivează și funcționarea circuitului (în caz de invalidare), sau poate activa ieșirea circuitului (în caz de validare)

Cap. 4: Rezultate experimentale

Buna funcționare a robotului este condiționată de calitatea elementelor mecanice. Deoarece pentru acest proiect acestea au fost obținute prin tehnica aditivă de tipărire 3D, se pot trage concluzii despre cerințele de material pentru fiecare componentă în parte. Cele mai importante aspecte sunt densitatea, grosimea pereților, precizia dimensională și aspectul suprafeței obținute. Precizia dimensională și aspectul suprafeței nu au nevoie de explicație, iar densitatea în cazul componentelor obținute prin tehnica 3D este la prima vedere dată de materialul utilizat, dar depinde și de gradul de umplere al elementului. Umplerea totală a elementului este foarte rar utilizată din cauza consumului excesiv de material, în schimb se utilizează un perete exterior de o anumită grosime și pentru interior se alege un model de umplere cu o anumită densitate. Combinația acestor două caracteristici dau rezistența finală a elementului obținut.

Componentele ce necesită rezistență ridicată sunt corpul robotului, căruciorul principal și mecanismul de înclinare, iar pentru acestea se impune o densitate de cel puțin 70% din densitatea materialului și o grosime a pereților de cel puțin 0.8mm. În ciuda faptului că toate elementele corpului principal au fost realizate cu aceste caracteristici, unul dintre elemente a cedat. Poziționarea și sensul ruperii materialului, cât și faptul că pe toata durata de testare acest eveniment a fost singular, sunt indicatori puternici ai cauzei defecțiunii: în acest caz a fost o problemă de adeziune între straturile de material. În figura următoare sunt prezentate diferite niveluri de densități ale aceluiași material și defectul apărut în procesul de testare.



Sursa: triplaxis.deviantart.com

Figura 49: Diferite densități ale materialului și un defect de material

Un rezultat notabil este durata reală de funcționare a telecomenzii, o singură încărcare a bateriilor a fost suficientă pentru o utilizare timp de 30 zile, iar cu o medie de 3 ore pe zi rezultă o durată efectivă de funcționare de peste 90 ore. Acest fapt indică spre un curent mediu de utilizare de 30mA, adică de 3-4 ori mai puțin decât curentul maxim consumat de telecomandă.

Figura 50: Telecomanda

Pentru stabilizarea robotului atât pe direcția față-spate, cât și pentru stânga-dreapta, cea mai bună metodă de stabilizare a fost efectuarea unei acțiuni în direcția perturbației(asemănpptor cu modul de control al unui pendul invers) , spre deosebire de sistemele electronice, unde la apariția unei perturbații ce scoate sistemul din echilibru, regulatorul încearcă să readucă sistemul în echilibru printr-o acțiune în sensul opus perturbației.

Aceasta se datorează proprietăților fizice ale sistemului, ce are o tendință naturală de a reveni la un punctul de echilibru, asemenea unui pendul. Dacă la acest sistem se încearcă o stabilizare prin accentuarea tendinței naturale de revenire la poziția de echilibru, atunci rezultatul va fi o supra-compensare ce duce la oscilații. Valorile cele mai satisfăcătoare obținute pentru coeficienții Kp, Ki și Kd sunt:

Controler	**Kp**	**Ki**	**Kd**
SERVO înclinare	10	0	0
Stabilitate înclinare	2,8	0	0
Stabilitate față-spate	6	0	3

Tabel 2: coeficienții regulatoarelor

Observație: Regulatorul ce realizează funcția de servomotor de înclinare funcționează normal, acționează în sensul opus perturbației.

Funcția de stabilizare automată în ambele direcții de deplasare aduce un plus de controlabilitate robotului și acum acesta poate efectua unele manevre asemănătoare cu robotul fictiv din film.

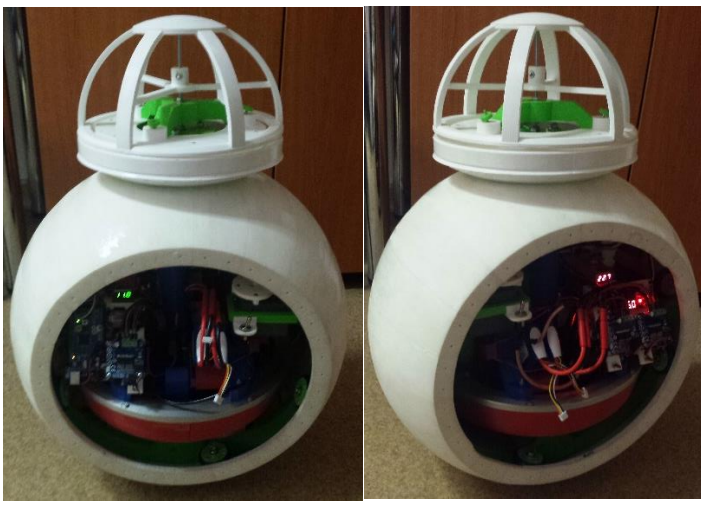


Figura 51: Modelul experimental complet funcțional

Cap. 5: Concluzii

Acest proiect a atins toate punctele propuse și poate constitui baza unui proiect mai mare ce include și elemente avansate de inteligență artificială. Deși domeniul roboților sferici este unul atractiv pentru cercetare, utilitatea lor este, cel puțin momentan, redusă. Problemele ce trebuie rezolvate pentru ca astfel de roboți să poată opera pe teren neuniform sunt legate de mobilitatea limitată și comportamentul greu de controlat în timpul mersului.

Performanțele robotului realizat sunt foarte dependente de suprafața pe care se află, dar roboții realizați cu metoda de conducere ce beneficiază de conservarea momentului unghiular au rezultate încurajatoare și par să îmbunătățească mobilitatea roboților sferici (spre exemplu pot urca o treaptă).

Dacă problemele de mobilitate și control ale acestor roboți vor fi soluționate, eu cred că putem deveni martori la reinventarea roții.

Cap.6 : Referințe

[1] Ivan Bogdanov, Conducerea Roboților , Editura Orizonturi Universitare, ISBN 978-973-638419-6

[2] Roborace.com

[3] Lin, X.; Guo, S.; Tanaka, K.; Hata, S. Development of a Spherical Underwater Robot

[4] Rotundus.com

[5] R. Chase;A. Pandya,A Review of Active Mechanical Driving Principles of Spherical Robots

[6] http://hackaday.com/2016/06/24/driving-bb-8-more-than-one-way-to-move-this-bot/

[7] Artusi, M.; Potz, M.; Aristizabal, J.; Menon, C.; Cocuzza, S.; Debei, S. Electroactive elastomeric actuators for the implementation of a deformable spherical rover

[8] Wait, K.; Jackson, P.; Smoot, L. Self Locomotion of a Spherical Rolling Robot Using a Novel Deformable Pneumatic Method.

[9]Sugiyama, Y.; Shiotsu, A.; Yamanaka, M.; Hirai, S. Circular/Spherical Robots for Crawling and Jumping

[10]Schroll, G. Dynamic Model of a Spherical Robot from First Pinciples. M.S. Thesis, Colorado State University, Fort Collins, CO, USA, 2010.

[11] https://github.com/XRobots/BB83_Public

[12] https://en.wikipedia.org/wiki/Autodesk_123D

[13]http://www.matterhackers.com/articles/printing-with-nylon

[14] http://gizmodorks.com/blog/all-about-tpu-filament-/

[15] https://en.wikipedia.org/wiki/Density

[16] https://en.wikipedia.org/wiki/Magnet#Force_between_two_bar_magnets

[17] https://en.wikipedia.org/wiki/Magnetization

[18] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[19] https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

[20] https://reference.digilentinc.com/reference/pmod/pmodjstk2/reference-manual

[21]http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_datasheet.pdf

[22] http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[23] http://www.robotshop.com/media/files/pdf/rb-ite-12-bluetooth_hc05.pdf

[24] https://en.wikipedia.org/wiki/Lithium_polymer_battery

[25] https://www.instructables.com/id/Motor-Driver-BTS7960-43A/

[26] https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[27] https://en.wikipedia.org/wiki/I²C

[28] https://learn.sparkfun.com/tutorials/i2c

[29] http://i.stack.imgur.com/nJdlZ.png

[30] https://en.wikipedia.org/wiki/Arduino

[31] http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[32] https://en.wikipedia.org/wiki/PID_controller

Anexa 1

Anexa 2

Anexa 3

Anexa 4

```
#include <SPI.h>
unsigned long previousMillis = 0;
const long interval = 40;           // interval de repetitie
int state = 0;

void conectare(){
        while(state == 0)
        {
        state = digitalRead(2);  // verifica conexiunea Bluetooth
        }
}
void setup() {
  pinMode(2, INPUT);
  pinMode(5, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  Serial.begin(115200);
  conectare();
}

void loop() {
  unsigned long currentMillis = millis();              // verifica cat e timpul
  if (currentMillis - previousMillis >= interval) {     // daca a trecut timpul impus, executa

  previousMillis = currentMillis;                    // salveaza momentul ultimei executii
  state = digitalRead(2);
  if (state == 0) {                                 // asteapta pana la realizarea conexiunii
    conectare();
  }

  digitalWrite(SS, HIGH);
  digitalWrite(7, HIGH);
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  SPI.begin ();
  SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));
  delay (1);

  byte c1,c2,c3,c4,c5,c6;
  uint16_t x1,x2,x3,x4,y1,y2,y3,y4,but1,but2,but3,but4,sw1,sw2;

  sw1=digitalRead(5);                              // citire comutatoare telecomanda
  sw2=digitalRead(6);

  digitalWrite(7, LOW);                             // selectarea primului joystick
    c1=SPI.transfer (0xC0);                         //citire valori
    delay(0.1);
    c2=SPI.transfer (0x00);
    delay(0.1);
    c3=SPI.transfer (0x00);
    delay(0.1);
    c4=SPI.transfer (0x00);
    delay(0.1);
    c5=SPI.transfer (0x00);
    delay(0.1);
```

```
   c6=SPI.transfer (0x00);
   y1=c1;
   but1 = (c5==129 || c5==131 ) ? 1 : 0 ;
   x1=c6;
 digitalWrite(7, HIGH);                    //dezactivarea primului joystick

 digitalWrite(8, LOW);                     // selectarea joystick 2
   c1=SPI.transfer (0xC0);                 //citire valori
   delay(0.1);
   c2=SPI.transfer (0x00);
   delay(0.1);
   c3=SPI.transfer (0x00);
   delay(0.1);
   c4=SPI.transfer (0x00);
   delay(0.1);
   c5=SPI.transfer (0x00);
   delay(0.1);
   c6=SPI.transfer (0x00);
   y2=c1;
   x2=c6;
 digitalWrite(8, HIGH);                    //dezactivarea joystick 2

  digitalWrite(9, LOW);                    // selectarea joystick 3
   c1=SPI.transfer (0xC0);                 //citire valori
   delay(0.1);
   c2=SPI.transfer (0x00);
   delay(0.1);
   c3=SPI.transfer (0x00);
   delay(0.1);
   c4=SPI.transfer (0x00);
   delay(0.1);
   c5=SPI.transfer (0x00);
   delay(0.1);
   c6=SPI.transfer (0x00);
   y3=c1;                                  // aceasta axa nu este utilizata
   but3= (c5==129 || c5==131 ) ? 1 : 0 ;
   x3=c6;
 digitalWrite(9, HIGH);                    //dezactivarea joystick 3

  digitalWrite(10, LOW);                   // selectarea joystick 4
   c1=SPI.transfer (0xC0);                 //citire valori
   delay(0.1);
   c2=SPI.transfer (0x00);
   delay(0.1);
   c3=SPI.transfer (0x00);
   delay(0.1);
   c4=SPI.transfer (0x00);
   delay(0.1);
   c5=SPI.transfer (0x00);
   delay(0.1);
   c6=SPI.transfer (0x00);
   y4=c1; // axis not used!
   but4= (c5==129 || c5==131 ) ? 1 : 0 ;
   x4=c6;
 digitalWrite(10, HIGH);                   //dezactivarea joystick 3

////////////// Transmitere valori prin interfata seriala catre modulul bluetooth ////////////////////////////

 Serial.print(x1);Serial.print(",");Serial.print(y1);Serial.print(",");
 Serial.print(x2);Serial.print(",");Serial.print(y2);Serial.print(",");
 Serial.print(x3);Serial.print(",");
 Serial.print(x4);Serial.print(",");
// Serial.print(but1);Serial.print(",");Serial.print(but2);Serial.print(",");
```

```
 Serial.print(but3);Serial.print(",");Serial.print(but4);Serial.print(",");
 Serial.print(sw1);Serial.print(",");Serial.print(sw2); Serial.print('\n');
////////////////////////////////////////////////////////////////////////////////////////////////////
 SPI.end ();
  }
}
```

Anexa 5

```
#include "Wire.h"    //Biblioteca pentru comunicatia prin I2C
#include "MPU6050.h"
#include <PID_v1.h>  //Biblioteca PID de la http://playground.arduino.cc/Code/PIDLibrary
#include <Servo.h>
//======================================================================//
////////////////////////////////////////////////// Declaratii variabile ////////////////////////////////////////////////////////
//======================================================================//

//////////////////////////Semnale de la telecomanda//////////////////
int ch1=128;
int ch2=128;
int ch3=128;
int ch4=128;
int ch5=128;
int ch6=128;
int but1 = 1;
int but2 = 1;
int but3 = 1;
int but4 = 1;

//////////////////////////////////////////////////////////////////////
int pot;                // variabila valoare potentiometru
int BTstate = 0;     // variabila stare conexiune modul bluetooth

////////////////////////////// Variabile IMU ////////////////////////////
MPU6050 accelgyro;
int16_t ax, ay, az;      // variabile valori acceleratie
int16_t gx, gy, gz;      // variabile valori giroscop

#define Gyr_Gain 0.00763358

float AccelX;
float AccelY;
float AccelZ;

float GyroX;
float GyroY;
float GyroZ;

float mixX;
float mixY;

float pitch;
float roll;
float pitchAccel, rollAccel;

////////////////////////////////////// Variabile pentru temporizare ////////////////////////////////////
long previousMillis = 0;
unsigned long currentMillis;
long interval = 10;         // perioada de citire accelerometru masurata in ms

float dt=0.01;              // intervalul de integrare masurat in secunde – pentru IMU

////////////////////////////////////////// Variabile PID ////////////////////////////////////////
double Pk1 = 10;
double Ik1 = 0;
double Dk1 = 0.05;
double Setpoint1, Input1, Output1, Output1a;
```

```
PID PID1(&Input1, &Output1, &Setpoint1, Pk1, Ik1 , Dk1, DIRECT);    // declaratie SERVO PID

double Pk2 = 2.8;
double Ik2 = 0;
double Dk2 = 0;
double Setpoint2, Input2, Output2, Output2a;

PID PID2(&Input2, &Output2, &Setpoint2, Pk2, Ik2 , Dk2, DIRECT);    // declaratie PID stabilitate-inclinare
double Pk3 = 6;
double Ik3 = 0;
double Dk3 = 3;
double Setpoint3, Input3, Output3, Output3a;

PID PID3(&Input3, &Output3, &Setpoint3, Pk3, Ik3 , Dk3, DIRECT);    // declaratie PID principal

////////////////////////////////////////////// Servomotoare//////////////////////////////////////////////////////////////
Servo servo1;          //declarația obiectului de tip servomotor
Servo servo2;          //declarația obiectului de tip servomotor
Servo servo3;          //declarația obiectului de tip servomotor

////////////// Variabile auxiliare ///////////
int varServo1;
int varServo2;

int ch1a;
int ch3a;
int ch4a;
int ch5a:

//=============================================================================//
////////////////////////////////////// SETUP - se executa o singura data dupa RESET ////////////////////////////////
//=============================================================================//

void setup() {

      Wire.begin();
      accelgyro.initialize();
      pinMode(LED_PIN, OUTPUT);
      pinMode(31, OUTPUT);      // pin enable
      pinMode(33, INPUT);       // stare conexiune bluetooth
      pinMode(3, OUTPUT);       // semnal motor inclinare
      pinMode(5, OUTPUT);       // semnal motor inclinare
      pinMode(6, OUTPUT);       // semnal motor rotire volant
      pinMode(7, OUTPUT);       // semnal motor rotire volant

      PID1.SetMode(AUTOMATIC);            // PID "SERVO" inclinare
      PID1.SetOutputLimits(-255, 255);
      PID1.SetSampleTime(20);

      PID2.SetMode(AUTOMATIC);            // PID stabilitate inclinare
      PID2.SetOutputLimits(-255, 255);
      PID2.SetSampleTime(20);

      PID3.SetMode(AUTOMATIC);            // PID motor principal
      PID3.SetOutputLimits(-255, 255);
      PID3.SetSampleTime(20);

      Serial.begin(115200);
      Serial1.begin(115200);
}
```

```
//=====================================================================//
////////////////////////////////////// BUCLA PRINCIPALA - se executa continuu ////////////////////////////////
//=====================================================================//
void loop() {

  currentMillis = millis();               // Interogare timp, millis returneaza cate milisecunde s-au scurs
                                          // Va depasi vloarea maxima unsigned long dupa 50 zile
  if (currentMillis - previousMillis >= interval)          // daca s-a scurs periada de timp setata, atunci
     {
     previousMillis = currentMillis;                       // se actualizeaza ultima inregistrare

     accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);          // se citesc valorile de la accelerometru

     AccelX = ax;
     AccelY = ay;
     AccelZ = az;
     GyroX = Gyr_Gain * (gx);
     GyroY = Gyr_Gain * (gy)*-1;
     GyroZ = Gyr_Gain * (gz);

     AccelY = (atan2(AccelY, AccelZ) * 180 / PI);
     AccelX = (atan2(AccelX, AccelZ) * 180 / PI);

     float K = 0.8;
     float A = K / (K + dt);

     mixX = A *(mixX+GyroX*dt) + (1-A)*AccelY;
     mixY = A *(mixY+GyroY*dt) + (1-A)*AccelX;

     pitch=mixY;
     roll=mixX;
////////////////////////////////////////////// Citire si actualizare semnale telecomanda ////////////////////////////////////////
      if (Serial1.available() > 0) {
       ch1 = Serial1.parseInt();
       ch2 = Serial1.parseInt();
       ch3 = Serial1.parseInt();
       ch4 = Serial1.parseInt();
       ch5 = Serial1.parseInt();
       ch6 = Serial1.parseInt();
       but1 = Serial1.parseInt();
       but2 = Serial1.parseInt();
       but3 = Serial1.parseInt();  //comutator validare
       but4 = Serial1.parseInt();
           }
////////////////////////////////////////////////////////////// Citire potentiometru inclinare ////////////////////////////////////////
       pot = analogRead(A0);
////////////////////////////////////// Validare comenzi pentru motoare și servomotoare//////////////////////////////////
       BTstate = digitalRead(33);
       if(but3 == 0 && BTstate == 1) {
         digitalWrite(31, HIGH);
         }
       else if (but3 == 1 || BTstate == 0) {
         digitalWrite(31, LOW);
       }

       if( but4 == 0)
       {
       servo1.attach(9);
       servo2.attach(10);
       servo3.attach(8);
       }
       else
       {
```

```
      servo1.detach();
      servo2.detach();
      servo3.detach();
      }
////////////////////////////////////////////////////////////// Comanda motor inclinare ////////////////////////////////////////////
      Input2 = roll -1;
      Setpoint2 = map(ch6, 0,255,-45,45);          // fixare canal 1 telecomanda ca marime de prescriere
      PID2.Compute();
      Setpoint1 = Output2;

      Input1 = map(pot, 160, 330, -45,45);         // scalare valori potentiometru
      Input1 = Input1-9;
      Input1 = constrain(Input1,-45,45);           // limitare la +/- 45 grade

      Setpoint1 = constrain(Setpoint1, -45,45);    // limitare la +/- 45 grade
      PID1.Compute();

// Comunicatie pnetru testare si depanare
//       Serial.print(" INPUT1 ");
//       Serial.print(Input1);
//       Serial.print(" SETPOINT1 ");
//       Serial.print(Setpoint1);
//       Serial.print(" OUTPUT1 ");
//       Serial.println(Output1);
//       Serial.print(" SETPOINT2 ");
//       Serial.print(Setpoint2);
//       Serial.print(" OUTPUT2 ");
//       Serial.println(Output2);

        if (Output1 <= -35)                // comanda efectiva a motorului cu semnal PWM
        {
        Output1a = abs(Output1);
        analogWrite(3, Output1a);
        analogWrite(5, 0);
        }
      else if (Output1 >= 35)
        {
        Output1a = abs(Output1);
        analogWrite(5, Output1a);
        analogWrite(3, 0);
        }
      else
        {
        analogWrite(5, 0);
        analogWrite(3, 0);
        }
////////////////////////////////////////////////////////////Comanda_motor_principal/////////////////////////////////////////////////
      Input3 = pitch;

      Setpoint3 = map(ch2, 0,255,-50,50);
      PID3.Compute();

        if (Output3 <= -30)        // comanda efectiva a motorului cu semnal PWM in functie de                semnul iesirii controlerului
  PID
        {
        Output3a = abs(Output3);
        analogWrite(12, Output3a);
        analogWrite(11, 0);
        }
      else if (Output3 > 30)
        {
        Output3a = abs(Output3);
        analogWrite(11, Output3a);
```

```
          analogWrite(12, 0);
          }
          else
          {
          analogWrite(11, 0);
          analogWrite(12, 0);
          }
//////////////////////////////////////////////////////////////////Rotire_volant/////////////////////////////////////////////////////////////////
        ch1a = map(ch1,0,255,-255,255);

        if (ch1a <= -10)                // selectie sens de rotatie
        {
          ch1a = abs(ch1a);
          analogWrite(7, ch1a);          // trimitere comandă către circuitul de comandă al motoarelor
          analogWrite(6, 0);
          }
        else if (ch1a > 10)             // selectie sens de rotatie
          {
          ch1a = abs(ch1a);
          analogWrite(6, ch1a);          // trimitere comandă către circuitul de comandă al motoarelor
          analogWrite(7, 0);
          }
        else {
          analogWrite(7, 0);             //frână
          analogWrite(6, 0);
        }
////////////////////////////////////////////////////////////////////// Controlul capului ////////////////////////////////////////////////////
        ch3a=map(ch4,0,255,180,0);
        ch3a = ch3a-(pitch*6);         // se misca capul in sensul opus inclinarii, pentru stabilizare

        varServo1 = ch3a;
        varServo2 = ch3a;
        ch4a=map(ch3,0,255,50,-50);
        ch4a = ch4a-(roll*15);      //se misca capul in sensul opus inclinari, pentru stabilizare

        varServo1 = varServo1-ch4a;
        varServo2 = varServo2+ch4a;

        varServo2 = map(varServo2,0,180,180,0);
        varServo1 = constrain(varServo1,0,180);
        varServo2 = constrain(varServo2,0,180);

        if (but3 == 1) {      // comanda de validare nu este activa
         servo1.write(60);    //trimitere către poziția de parcare
         servo2.write(120);   //trimitere către poziția de parcare
        }

        else if (but3 == 0) {       // comanda de validare este activa
        servo1.write(varServo1);    //comanda servomotoare prin semnal PWM
        servo2.write(varServo2);    //comanda servomotoare prin semnal PWM
        Serial.print("S1:");
        Serial.print(varServo1);Serial.print("S2:");
        Serial.println(varServo2);
        }
//////////////////////////////////////////////////////////////////Rotire cap/////////////////////////////////////////////////////////////////
       ch5a = map(ch5, 0,255,0,180);
       servo3.write(ch5a);
////////////////////////////////////////////////// Sfârșitul buclei temporizate//////////////////////////////////////////
     }
//////////////// Sfârșitul buclei principale (main)/////////////////////////////////
 }
```