Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Tehnica ” Gheorghe Asachi ”, Iași

Specializarea: Electronică Aplicată

Lucrare de licență

Coordonator științific:

Asist. Dr. Ing. Amăriuței Roxana Daniela

Student:

Danu Elena-Diana

Iasi, 2018

Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Universitatea Tehnica ” Gheorghe Asachi ”, Iași
Specializarea: Electronică Aplicată

Senzoristică si actuatori pentru sistem autonom de conversie a energiei solare in energie electrică

Coordonator științific:

Asist. Dr. Ing. Amăriuței Roxana Daniela

Student:

Danu Elena-Diana

Iasi, 2018

Cuprins

Capitolul 1. Introducere

1.1 Memoriu justificativ

In momentul de față, la nivel mondial, principala resursă energetică o constituie combustibilii: petrol, lemn, cărbune, reziduuri combustibile dar și energia produsă de hidrocentrale și de centralele nucleare. Deoarece numărul populației este în creștere iar cerințele sunt tot mai mari, trebuie sa promovăm noi tehnologii privind utilizarea resurselor energetice. O tehnologie nepoluantă și din ce în ce mai populară este reprezentată de folosirea panourilor solare.

Energia solară a căpătat rapid notorietate, fiind un mijloc important de a folosi energia regenerabilă. Astfel, este important sa fie cunoscute tehnologiile asociate cu această arie de lucru.

Panourile solare ar reprezenta soluții simple pentru problema încălzirii dar și pentru generarea de electricitate. Ideea utilizării efectului termic al radiației solare este veche. Încă din antichitate Archimede a incendiat flota romană concentrând razele solare cu ajutorul oglinzilor. Odată cu dezvoltarea societății, pe plan tehnologic s-au implementat sisteme de proiectare avansate pentru a obține sisteme eficiente și fiabile.

În această lucrare îmi propun să descriu funcționarea unui sistem autonom bazat pe folosirea energiei solare. Aplicația reprezintă un ansamblu de blocuri electro-mecanice, motoare, senzori și module electrice de comandă pentru asigurarea controlului.

Am realizat acest proiect în vederea alimentarii cu energie electrică a unui consumator, pentru care a fost necesar controlul nivelului de încărcare și descărcare a unei baterii de 12 V dar și pentru protecția acesteia la supratensiune, supracurent, supraîncărcare. Panoul preia radiația luminoasă cu ajutorul senzorilor, datele colectate de aceștia fiind mai departe transmise către microcontroler. Pentru un randament maxim de încărcare a bateriei, panoul este ghidat automat către cea mai puternică sursă de lumină. Capacitatea maximă este atinsă atunci când razele soarelui sunt ghidate perpendicular pe suprafața panoului. Datele oferite de senzor sunt transmise la microcontrolerul care determina starea sistemului și în acest fel reglează poziția celor 2 motoare.

Pentru a avea acces la informații precum intensitatea luminii, nivelul bateriei, tensiunea convertorului, aplicația oferă utilizatorului date în timp real prin intermediul unei interfețe grafice (GUI - Graphical User Interface).

1.2 Celula fotovoltaică

Principiul de funcționare a unui panou solar electric se bazează pe faptul că permite fotonilor, numite și particule de lumină, să se ciocnească de electroni desprinși de atomi, generând astfel curent electric.

Sistemele de panouri solare sunt formate din unități mai mici, numite celule fotovoltaice. Celula fotovoltaică sau celula solară este un dispozitiv ce realizează conversia energiei luminii în electricitate.

Efectul fotovoltaic constă în apariția unui curent electric sau a unei tensiuni pe suprafața materialului supus la o radiație luminoasă. Randamentul tehnologiei prin care funcționează celulele fotovoltaice variază in functie de intensitatea luminii soarelui cat si de direcția sub care cad razele de lumina. Astfel, celulele operează atat timp cat soarele straluceste, o cantitate mai mare de electricitate este produsă atunci când razele soarelui cad perpendicular pe suprafața respectivă.

Celula solară absoarbe o parte din particulele de lumină ce cad pe aceasta, numite și fotoni. Fiecare foton conține o cantitate mică de energie. Atunci când un foton este absorbit, acesta eliberează un electron din materialul celulei solare. In acest mod se va genera un curent electric care va fi mai departe colectat prin cablul conectat la respectiva celulă solară. Celula va produce electricitate ce poate fi folosită instantaneu sau înmagazinată în acumulatori.

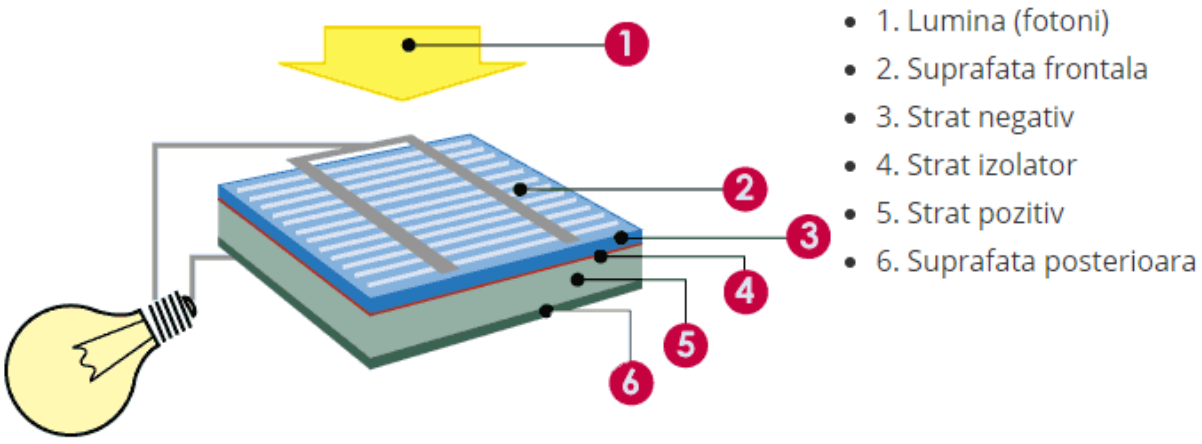


Figura 1.

1.3 Radiația solară

Soarele emite în mod continuu cantități uriașe de energie. O parte din această radiație ajunge pe Pământ. O altă parte a luminii este absorbită de atmosferă. Cantitatea de energie ce ajunge pe Pământ într-o zi este mai mare decât întregul consum al Pământului pe durata unui an întreg. Datorită trecerii sale prin atmosferă, radiația solară este supusă fenomenelor de absorbție, difuzie și transmisie, fiind redusă cu aproximativ 30%.

Radiația solară se poate clasifica în:

-radiație directă;
-radiație difuză;
-radiație globală;
-radiație reflectată;

Lumina ce s-a împrăștiat în atmosferă este ceea ce noi numim lumina difuză sau radiație difuză. Raza de lumină ce ajunge pe suprafața solului fără sa fie împrăștiată este denumită radiație directă. Radiația solara directă este cunoscută și simțită în mod direct de către oameni.

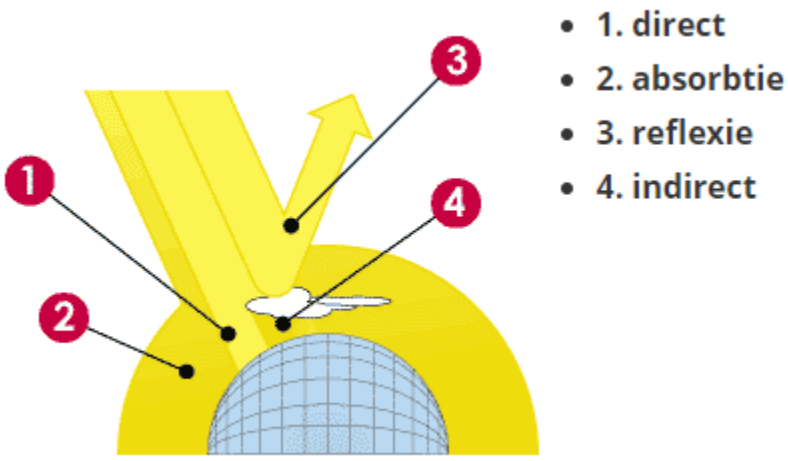


Figura 2.

Capitolul 2. Fundamentare teoretică

2.1 Descrierea blocurilor componente:

Schema bloc a aplicației:

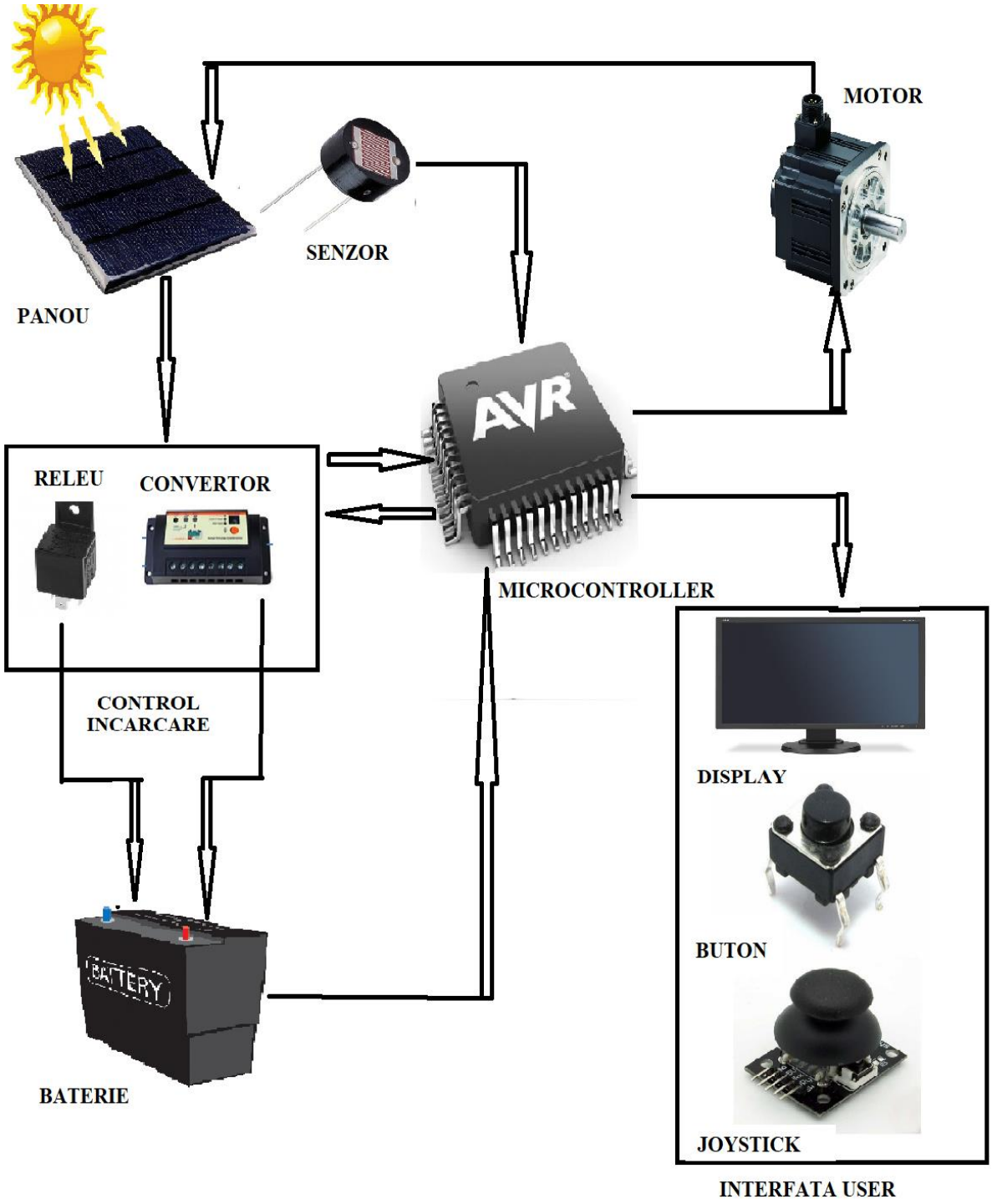


Figura 3.

1.Senzorul:

Senzorul implementat în acest proiect are la bază 4 divizoare rezistive de tensiune, fiecare fiind format dintr-o rezistență fixă, potențiometru și fotorezistență. Fotorezistențele sunt componente electronice cu proprietatea de a-și modifica rezistență electrică în funcție de intensitatea radiației luminoase.

Rolul fotorezistențelor este de a prelua separat informații pe care le transmit la microcontroler.

2.Motor:

În proiect folosim un motor DC utilizat pentru a roti in plan orizontal panoul si un motor Stepper utilizat pentru a inclina in plan vertical panoul.

a. **Motoarele DC**:

Motoarele DC sau motoarele de curent continuu sunt dispozitive electromecanice care utilizează interacțiunea câmpurilor magnetice și conductorilor pentru a transforma energia electrică în energie mecanică rotativă. Acest lucru se realizează prin producerea unei rotații unghiulare continue care poate fi utilizată pentru a roti pompele, ventilatoarele, compresoarele, roțile, în general aplicații care necesită controlul vitezei.

Un motor DC este alcătuit din două părți:

-statorul - partea staționară;

-rotorul - partea rotativă;

b. Motoarele Stepper:

Motoarele Stepper sunt dispozitive electromecanice care convertesc un semnal de intrare într-o mișcare discretă mecanică. Motorul pas cu pas nu se rotește în mod continuu, ci se mișcă în pași sau incrementări discrete deoarece rotorul intern are un număr mare (finit) de dinți de magnet permanent cu un număr de dinți electromagnetici montați pe stator. Fiecare electromagnet de pe stator este polarizat și depolarizat secvențial determinând rotorul să se rotească câte un pas la un moment dat.

Direcția de rotație poate fi de asemenea selectată împreună cu modul de pas: într-un singur pas sau rotație continuă (fără trepte) în direcția selectată.

3.Panoul:

Panoul solar este un ansamblu de celule solare care folosește drept sursă de energie soarele. Există o gamă largă de aplicații ce utilizează energia produsă de către celule solare începând de la ceasuri de mână, calculatoare și ajungând până la mijloace de transport și sateliți.

Principalul avantaj al folosirii panourilor solare este reducerea costurilor dar și protejarea mediului, deoarece energia solară este gratuită, inepuizabilă, ecologică și autonomă.

4.Baterie:

In general, bateria este un mediu electrochimic de stocare a energiei. Bateriile folosite în aplicațiile solare pot fi acid FLA (flooded) sau sigilate VRLA. Bateriile solare VRLA sunt de tip AGM sau GEL. Tipul de baterie este important la setarea regulatorului de încărcare deoarece parametrii de încărcare diferă de la un tip de acumulator solar la altul.

Bateriile de acumulatori pentru sisteme fotovoltaice au rolul de a înmagazina energia produsă în timpul zilei de panourile fotovoltaice și de a o reda sistemului pe perioada nopții sau în cazul în care producția nu este suficientă. Acestea au o construcție specială care suportă un număr mare de cicluri de încărcare - descărcare.

5.Control încărcare:

Prin controlerele de încărcare (regulatoarele) din sistem realizăm încărcarea bateriilor solare de către panourile fotovoltaice. Rolul controlerului de încărcare este de a gestiona încărcarea și de a prelungi durata de viață a bateriilor solare din sistemul fotovoltaic. Dispozitivul stabilizează tensiunea produsă de panouri deoarece aceasta nu este constantă în timp.

Regulatoarele de încărcare au 3 cicli de bază pe perioada încărcării acumulatorilor solari:

-Faza de BULK - Prima etapă a încărcării – curentul de încărcare este maxim iar tensiunea crește gradual.
-Faza de absorbție - Tensiunea este menținută la nivel de bulk dar curentul scade pe perioada încărcării.
-Faza de FLOAT -ultima faza a încărcării, de eliminare a gazelor produse la încărcare.

6.Microcontroler:

Microcontrolerul este un sistem autonom cu periferice, memorie și un procesor care poate fi folosit ca sistem încorporat în dispozitive controlate automat, inclusiv scule electrice, jucării, dispozitive medicale implantabile, mașini de birou, sisteme de control al motoarelor, aparate, telecomenzi și alte tipuri de sisteme încorporate.

Un microcontroler poate fi folosit pentru creearea de diverse aplicaţii, nefiind necesară adăugarea altor cipuri si realizându-se astfel o economie de spaţiu şi de investiţie.

Microcontrolerul este cel care determină când bateria este incărcată complet fără a permite supraîncărcarea :

-previne transferul de energie din baterie către celula solară pe timpul nopții;
-reduce deteriorarea bateriei printr-o descărcare totală;
- pune la dispoziția utilizatorului starea sistemului;
-asigură mecanismul de protecție la scurtcircuit;

INTERFAȚĂ UTILIZATOR:

7.Display:

Display-ul este un periferic de ieșire utilizat pentru afișarea grafică de imagini și date folosit în tehnica prelucrării datelor. Informațiile sunt reprezentate prin caractere și simboluri.

Datele sunt oferite utilizatorului prin intermediul unuei interfețe grafice (GUI -Graphical User Interface). GUI-ul este un tip de interfață care permite utilizatorilor să interacționeze cu diverse dispozitive electronice prin intermediul icoanelor grafice. Sunt folosite în multe dispozitive mobile, cum ar fi playerele portabile, dispozitive de jocuri, telefoane inteligente și dispozitive de control casnice.

8.Driver pentru display:

Driverul este o parte de software care conectează sistemul de operare de partea hardware a aplicației. Calculatorul are nevoie de drivere pentru a utiliza diferite părți. Ne putem gândi la drivere ca la manualele de instrucțiuni pe care sistemul de operare le utilizează pentru a folosi în mod corespunzător computerul. Un driver de afișare este driver-ul care ajută computerul să utilizeze placa video. Fără această parte de software, placa video nu este 100% utilizată iar calculatoarele vor avea o rezoluție mult mai mică și vor afișa grafice slabe.

În electronică, un driver de afișare este de obicei un circuit integrat care oferă o interfață între un microprocesor, microcontroller sau o interfață periferică generală și un anumit tip de dispozitiv de afișare.

Dispozitive de afișare:

- LCD -liquid-crystal display;
- LED - light-emitting diode;
- OLED - organic light-emitting diode;
- ePaper - Electronic paper ;
- CRT -cathode ray tube ;
- Fluorescent vid -vacuum fluorescent display;
- Nixie - Nixie tube;

Driverul va accepta comenzi și date cu ajutorul unei interfețe standard serială sau paralelă precum TTL, RS232, CMOS, I2C, SPI și generează semnale de tensiune, curent, temporizare și demultiplexare pentru a ajuta la expunerea dorită, de text sau imagine.

Acesta poate fi un microcontroler specific și poate include memoria RAM, memoria flash, EEPROM și / sau ROM.

9.Buton:

Butonul este un dispozitiv de intrare hardware cu rolul de a schimba modurile sistemelor de operare. Designul interfeței hardware (HID) este un câmp de proiectare interdisciplinar care formează conexiunea fizică între oameni și tehnologie pentru a crea noi interfețe hardware care transformă procesele pur digitale în metode analogice de interacțiune. Un element al interfeței îl reprezintă butonul, pe lângă touchscreen, glisoare, comutatoare, senzori de intrare, microfoane, camere și accelerometre.

10.JoyStick:

Un joystick este un dispozitiv de intrare alcătuit dintr-un baston care pivotează pe o bază și raportează unghiul sau direcția spre dispozitivul pe care îl controlează. Joystick-ul, cunoscut și sub denumirea de coloană de comandă, este dispozitivul principal de comandă în cabina de pilotaj a mai multor aeronave civile și militare.

Fiind un dispozitiv analogic, joystick-ul funcționează prin raportarea unghiului de deformare. Mișcările dispozitivului de indicare sunt afișate pe ecran prin mișcările cursorului, creând o modalitate simplă și intuitivă de navigare a interfeței grafice a utilizatorului (GUI) a unui computer.

2.2 Descrierea aplicațiilor folosite:

1. **Vectary:**

Vectary este un instrument 3D pentru modelare și personalizare, care oferă modalități ușoare de a crea forme complexe și o gamă largă de instrumente, permițând utilizatorilor de orice nivel sa obțină efectele dorite. Cu această aplicație putem crea suprafețe complexe și forme netede, ajustând geometria prin diverse glisoare, mânere și selecții.

2. **Atmel Studio:**

Atmel Studio 7 este o platformă de dezvoltare integrată (IDP) utilizată pentru depanarea aplicațiilor microcontrolerlor AVR și SAM. Atmel Studio 7 IDP oferă un mediu ușor de utilizat pentru scrierea, construirea și depanarea aplicațiilor scrise în C / C ++ sau codul de asamblare. Se conectează fără probleme la programele de depanare, programatori și kiturile de dezvoltare care suportă dispozitivele AVR și SAM.

3. **AVR Extreme Burner:**

EXtreme Burner-AVR este o serie completă de interfețe grafice (GUI) AVR care suportă mai multe tipuri de surse de clock pentru diferite aplicații. Acesta permite citirea și scrierea pe un Oscilator RC sau un oscilator de cristal de mare viteză.

4. **Eagle:** 

Eagle este o aplicație automatizată de proiectare electronică (EDA), scriptabilă, cu captare schematică, imprimare layout pentru plăci de circuite(PCB), caracteristici de rout-are și asistări de calculator.

EAGLE = Easily Applicable Graphical Layout Editor ;

2.3 Listă componente:

1.Listă pentru control panou:

- 2 motoare;
- 1 soclu AVR ISP;
- 4 rezistențe;
- 4 potențiometre;
- 4 fotorezistențe;

2.Listă pentru control încărcare:

- 1 convertor tensiune ;
- 1 convertor mini360;
- 1 soclu AVR ISP;
- 2 conectori jack;
- 2 diode redresoare;
- 2 rezistențe de 10k;
- 2 rezistențe de 3k;
- 1 rezistența de 1.1k;
- 1 rezistența de 120k;
- 1 condensator de 100nF;
- 2 condensatoare de 330nF;
- 2 switch-uri;
- 1 releu;

3.Listă pentru microcontroler:

- 1 microcontroler ATMEGA 32A;
- 1 adaptor display;
- 3 socluri AVR ISP;
- 6 pini 1x8 ;
- 1 rezistența de 10k;
- 1 condensator electrolitic 470 uF;
- 1 conector jack;
- 1 regulator liniar 7805;
- 1 radiator;
- 2 switch-uri;

2.4 Schema aplicației:

Schema electrică utilizată pentru interfațarea microcontroler-ului:

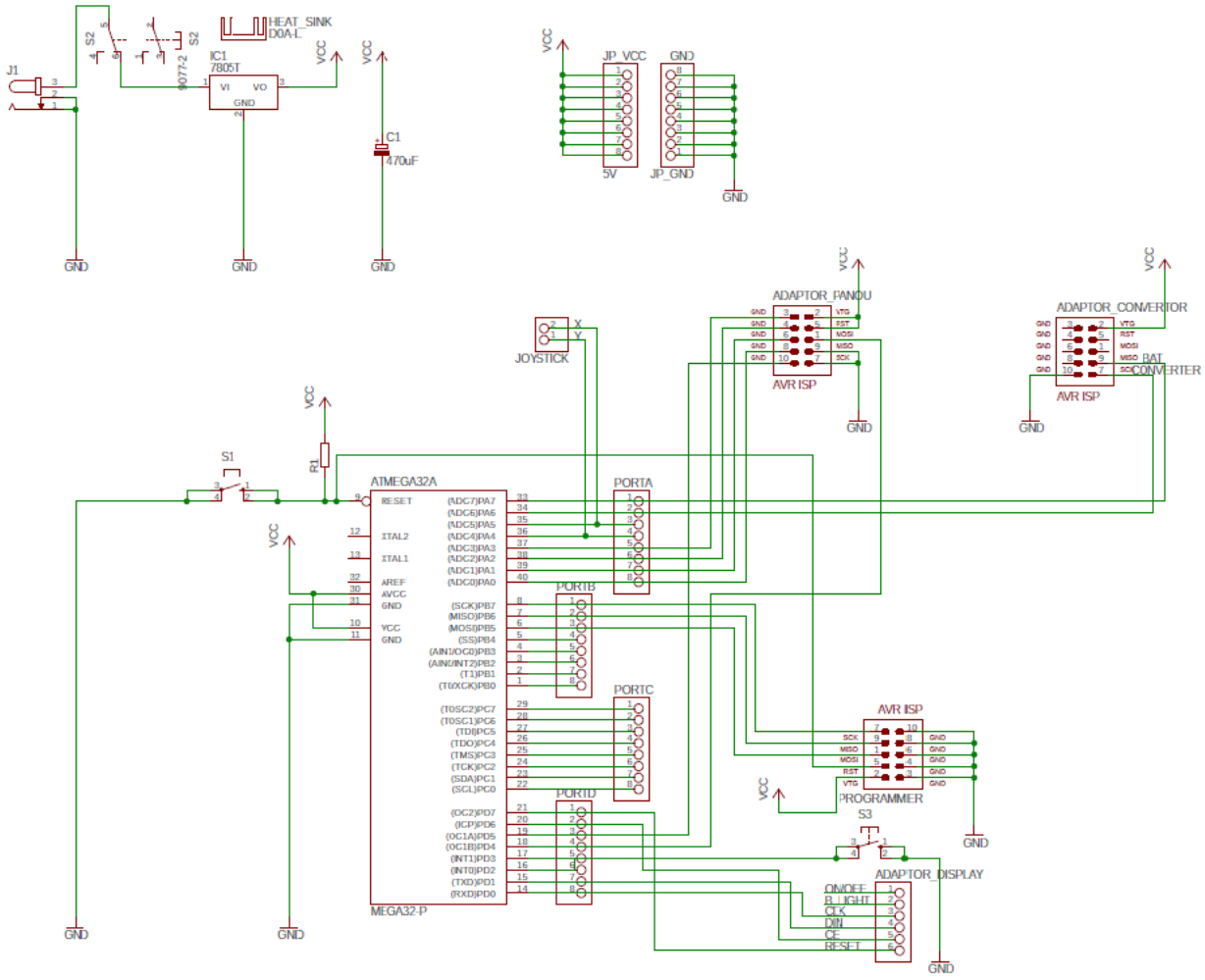


Figura 4.

Schema electrica pentru convertor:

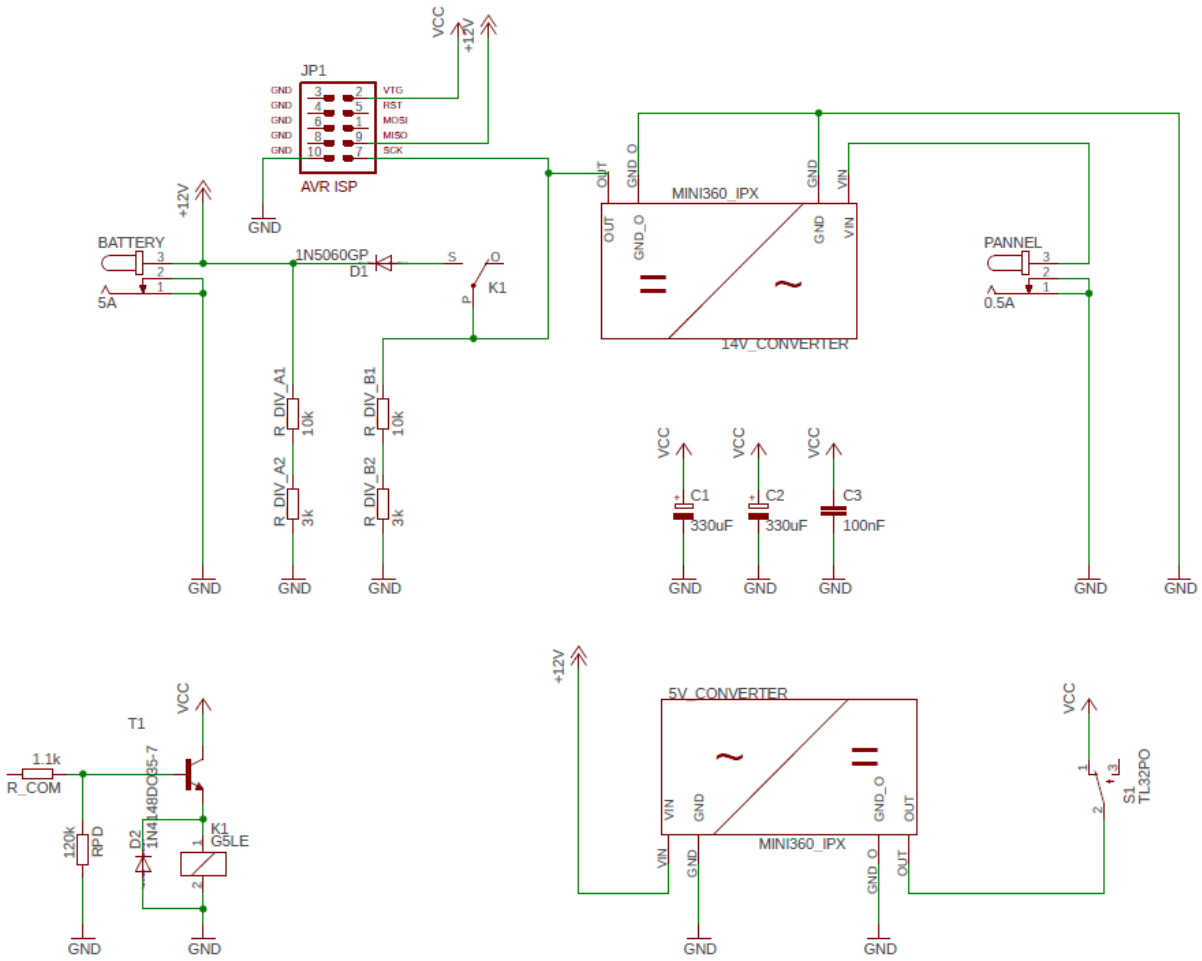


Figura 5.

Schema electrica pentru control panou:

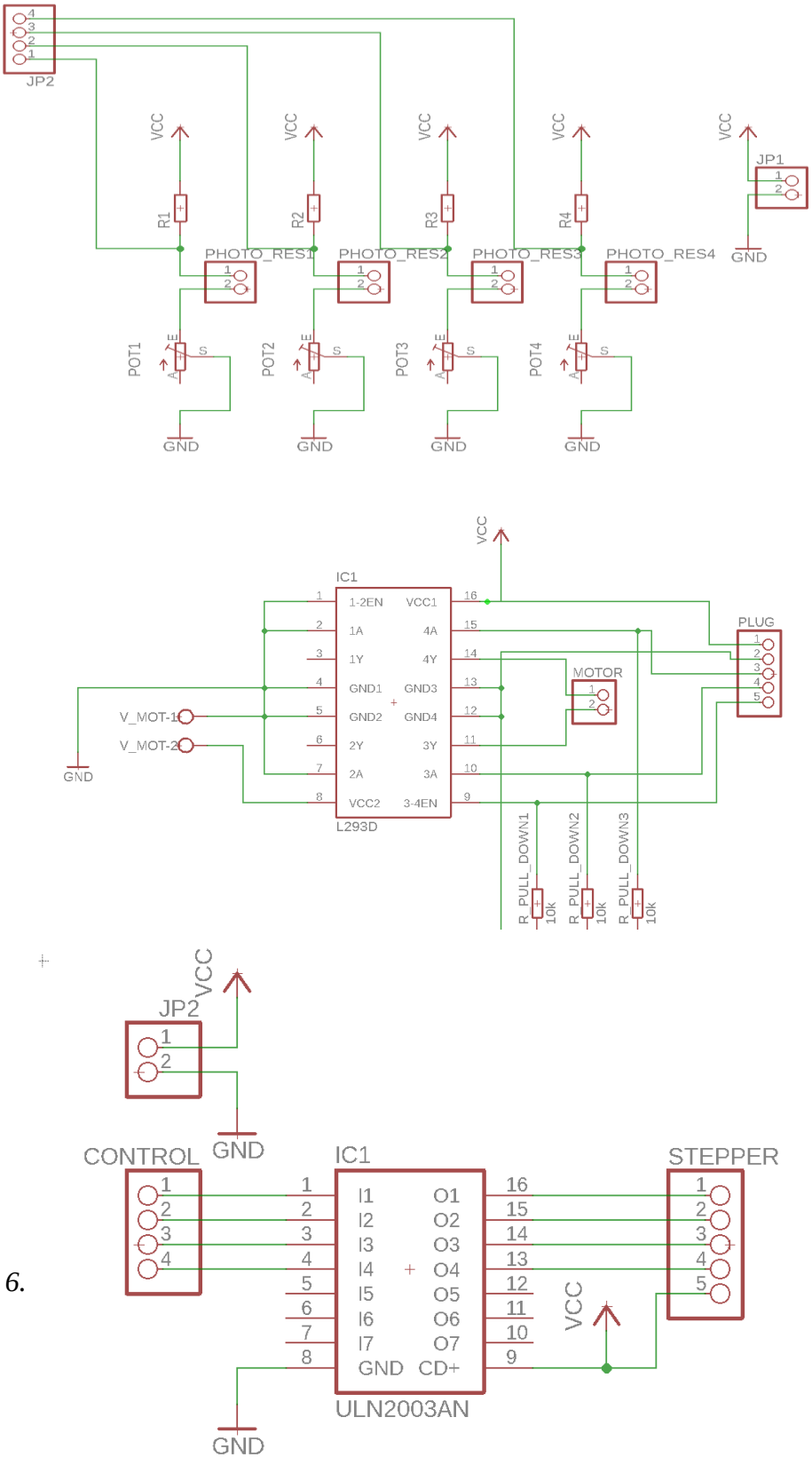


Figura 6.

Capitolul 3.Module

3.1 Senzorul

3.1.1 Generalități

Un senzor este un dispozitiv, un modul sau un subsistem al cărui scop este să detecteze evenimente sau modificări în mediul său și să trimită informațiile către alte electronice, adesea un procesor de calculator. Un senzor este utilizat întotdeauna împreună cu alte sisteme electronice.

Senzorii sunt utilizați în obiectele de zi cu zi, precum butoanele sensibile la atingere și lămpile care diminuează sau strălucesc prin atingerea bazei. Odată cu avansarea platformelor de microprocesoare și de microcontrolere, utilizările senzorilor s-au extins dincolo de câmpurile tradiționale de măsurare a temperaturii, presiunii sau debitului. Mai mult, senzorii analogici, cum ar fi potențiometrele și rezistențele sensibile la forță, sunt încă utilizate pe scară largă.

Aplicațiile includ industria aerospațială, medicina, robotica, industria auto și multe alte aspecte ale vieții noastre de zi cu zi.

Un senzor bun trebuie să aibă următoarele caracteristici:

- să fie sensibil la proprietatea măsurată;
- să fie insensibil la orice altă proprietate care ar putea fi întâlnită în aplicația sa;
- să nu influențeaze proprietatea măsurată;

Există astăzi senzori pentru mai mult de 100 de mărimi fizice, iar dacă se iau în considerare și senzorii pentru diferite substanțe chimice, numărul lor este de ordinul sutelor. Se pot pune în evidență circa 2000 de tipuri distincte de senzori, oferite în 100.000 de variante, pe plan mondial.

Clasificare:

a. În funcție de tipul mărimii fizice de intrare :

- absoluți, când semnalul electric de ieșire poate reprezenta toate valorile posibile ale mărimii fizice de intrare, raportate la o origine aleasă;

- incrementali, când nu poate fi stabilită o origine pentru toate punctele din cadrul domeniului de măsurare;

b. In funcție de tipul mărimii de ieșire:

- senzori analogici, pentru care semnalul de ieșire este în permanență proporțional cu mărimea fizică de intrare;

- senzori numerici (digitali), la care semnalul de ieșire poate lua numai un număr finit de valori discrete, care permit cuantificarea semnalului fizic de intrare;

c. In funcție de domeniul în care sunt utilizați:

- În industrie : robotică, fabricație flexibilă, controlul calității, activități de birou etc.
- În protecția mediului ;
- În transporturi ;
- În automatizarea clădirilor și locuințelor;

3.1.2 Schematic

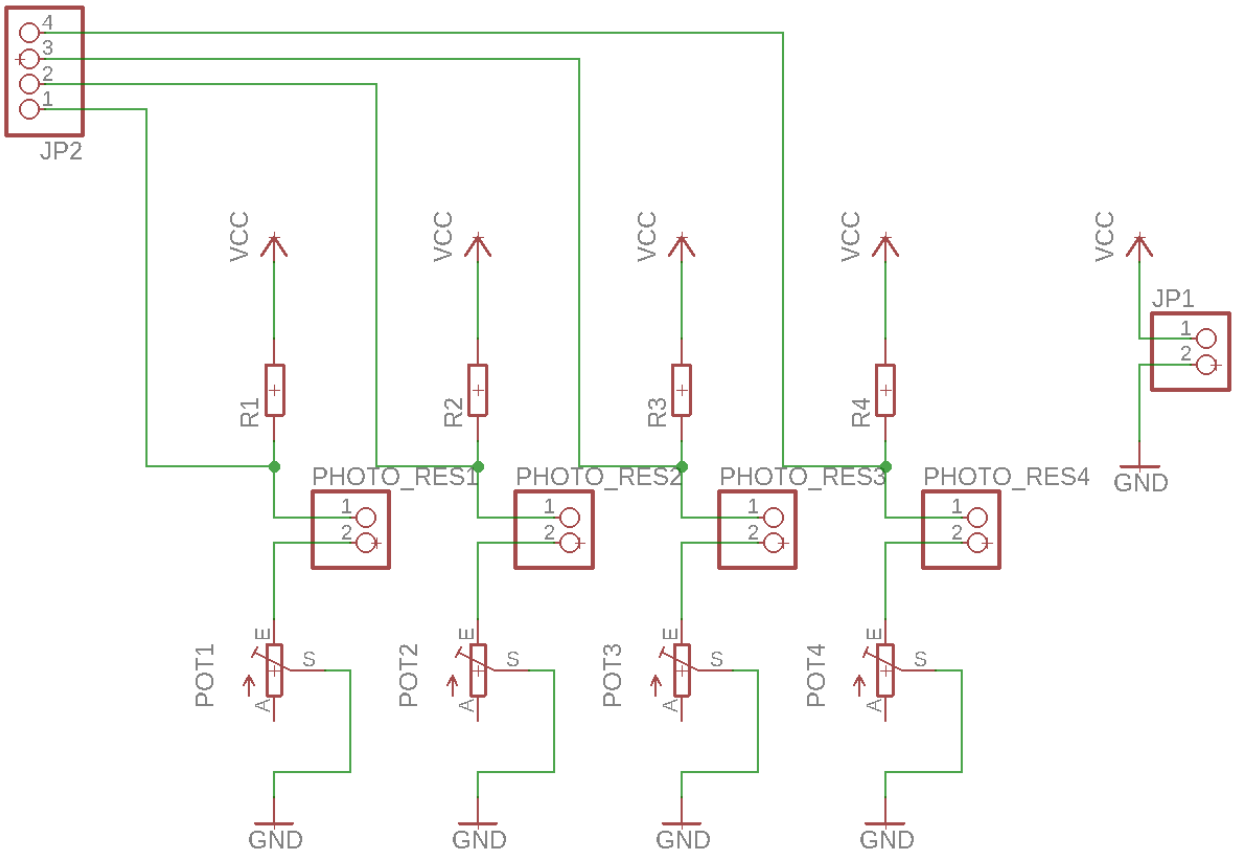
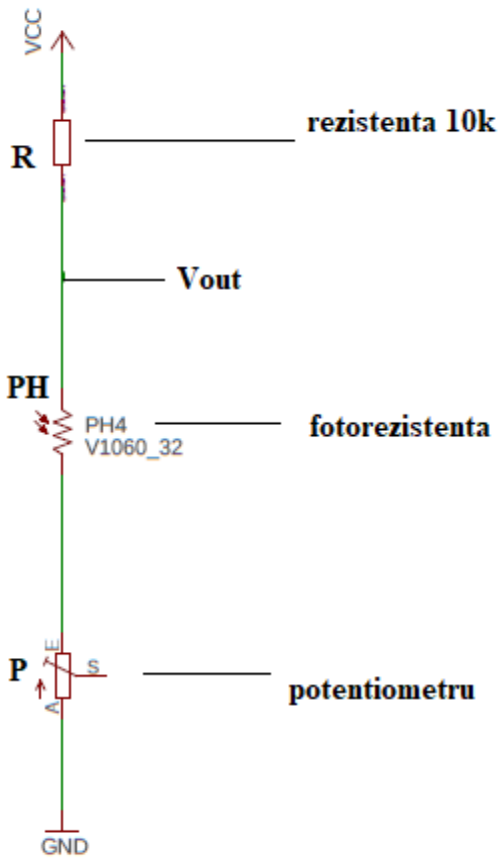


Figura 7.

Senzorul folosit are la baza 4 divizoare rezistive de tensiune, fiecare fiind format din rezistență fixă, potențiometru și fotorezistență.

Divizorul rezistiv:



$$Vout = \frac{PH + P}{PH + P + R} * VCC$$

Figura 8.

Rezistență fixa:

Rezistorul este o componentă din circuitele electrice și electronice a cărei principală proprietate este rezistența electrică. Aceștia sunt folosiți cu convertizăorii pentru a forma un subsistem al unui senzor. Convertizorii sunt componente electrice care convertesc energia dintr-o formă în alta, unde o formă de energie dintre cele două este electrică.

În circuit, rolul rezistorului poate fi:

- producerea căderii tensiunii dorite între două puncte din circuit;
- determinarea curentului dorit printr-o altă componentă a circuitului;
- divizarea unei tensiuni într-un raport dat (circuit divizor de tensiune);
- terminarea unei linii de transmisie(ca rezistență de sarcină).

Fotorezistența:

Denumita și LDR - Light Dependent Resistor, este un rezistor, realizat dintr-un material semiconductor omogen, a cărui rezistență se modifică sub incidența unui flux luminos.
Se bazează pe fenomenul de fotoconductivitate prin care sub influența radiației luminoase sunt eliberați electroni liberi care cresc conductivitatea electrică a semiconductorului și implicit scad rezistența rezistorului.

Fotorezistențele se pot utiliza atât în curent continuu cât și în curent alternativ. Rezistența ohmica a fotorezistenței scade cu creșterea iluminării.

Coeficientul de modificare a rezistenței cu temperatura este scăzut și se micșorează cu creșterea iluminării.

Schematic fotorezistențe:

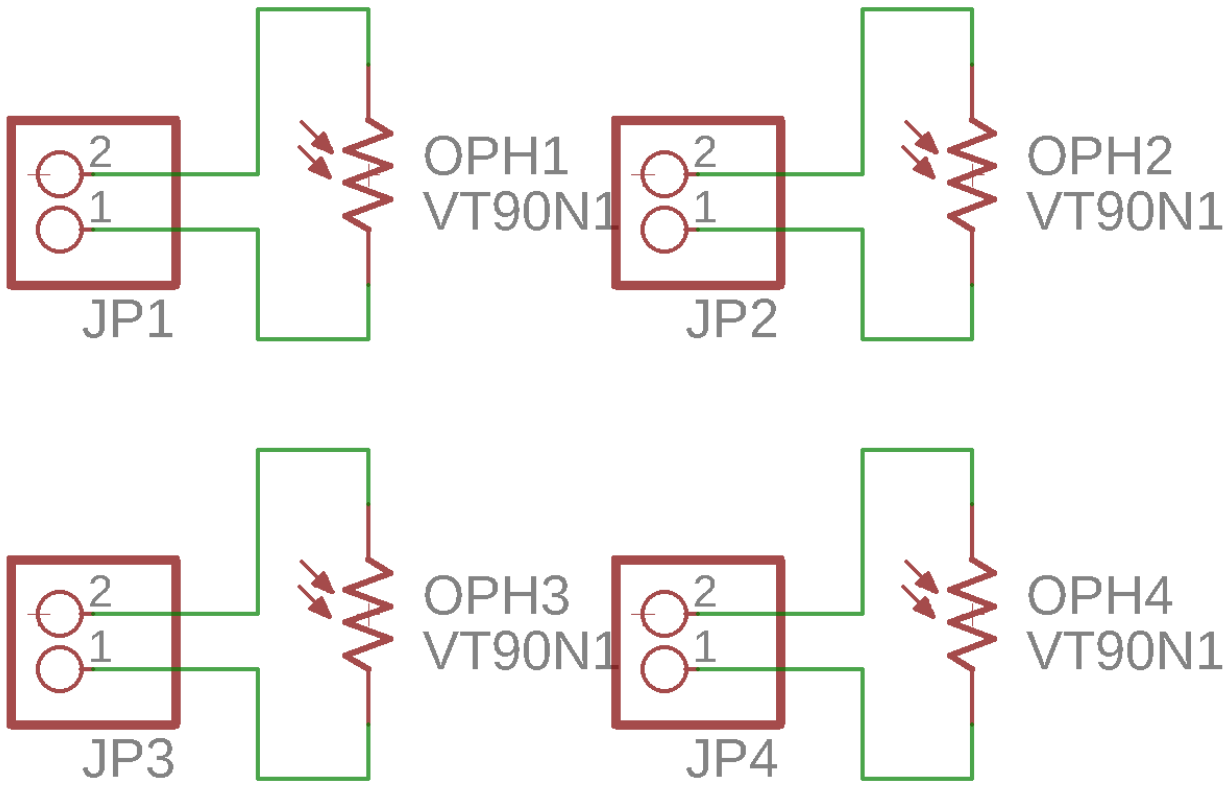


Figura 9.

Potențiometru:

In proiectul prezent folosim 4 potențiometre cu ajutorul cărora setăm valori pentru fotorezistențele folosite.

Potențiometrul este un instrument utilizat pentru a varia tensiunea într-un circuit, un rezistor cu un element mobil poziționat cu ajutorul unei manete și funcționează ca un divizor de tensiune.

Elementul mobil, denumit și perie, face contact cu un material rezistiv dezizolat, în oricare dintre punctele selectate manual.

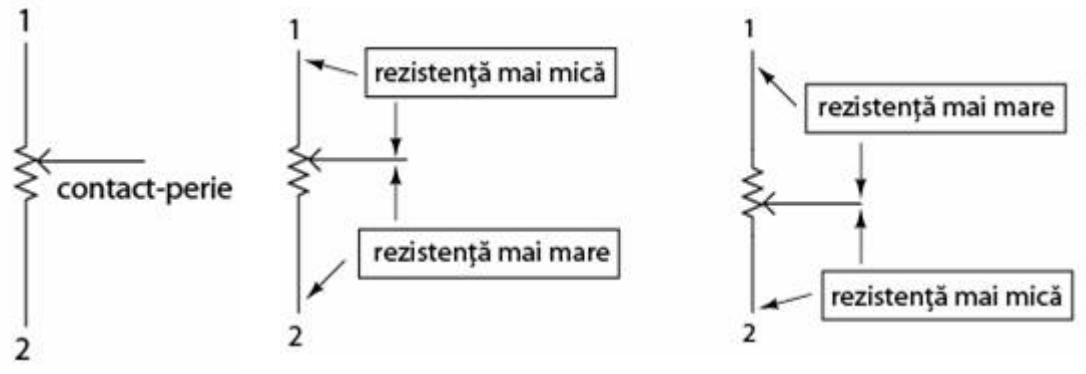


Figura 10.

Pe măsură ce contactul periei se apropie de terminalul 1 și se îndepărtează de terminalul 2, rezistența spre terminalul 1 scade iar cea către terminalul 2 crește. Dacă apropiem contactul de terminalul 2, vom obține efectul contrar. Rezistența între cele două puncte (1 și 2) este constantă indiferent de poziția contactului periei.

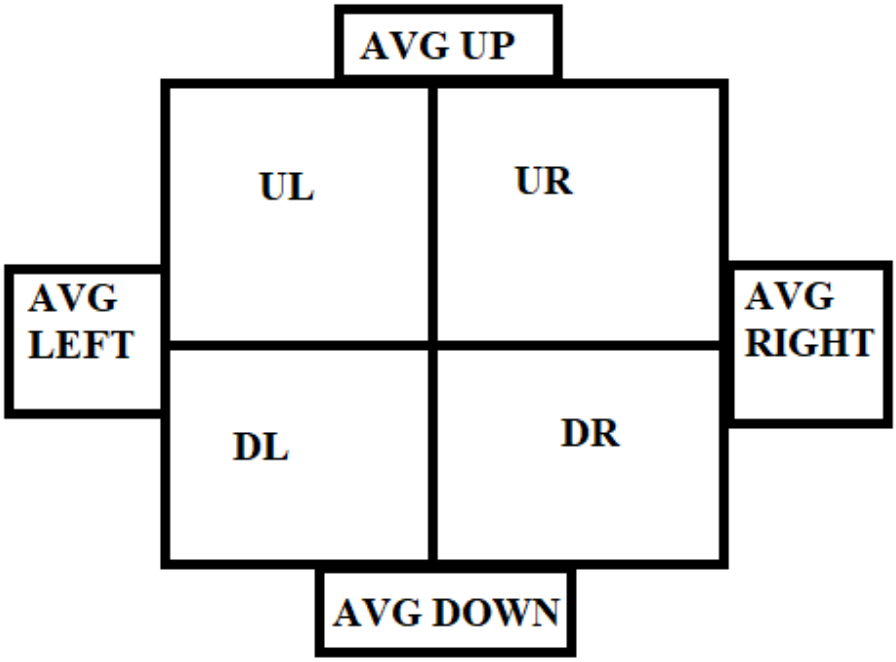
3.1.3 Scutul

Pentru o funcționare adecvată și cât mai corectă a senzorului a fost nevoie de un scut. Rolul scutului este de a împărți razele luminoase în 4 direcții de interes, câte una pentru fiecare fotorezistență. Scutul a fost creat 3D cu ajutorul aplicației Vectary.

Figura 11.

3.1.4 Modul de funcționare

Pentru a sesiza modul în care se mișcă panoul este necesar să facem media celor 4 valori ale zonelor de interes delimitate de scut.



$$\text{AVG UP} = \frac{UL+UR}{2} \qquad \text{AVG LEFT} = \frac{UL+DL}{2}$$

$$\text{AVG DOWN} = \frac{DL+DR}{2} \qquad \text{AVG RIGHT} = \frac{UR+DR}{2}$$

Figura 12.

UL = up left;

UR = up right;

DL = down left;

DR = down right;

AVG = average = medie;

UL, UR, DL, DR sunt calculate fiecare ca media aritmetică a 15 valori citite de la ADC de pe fiecare senzor în parte.

Comandă:

AVG UP > AVG DOWN => panoul se inclină în sus;

AVG UP < AVG DOWN => panoul se inclină în jos;

AVG LEFT > AVG RIGHT => panoul se va roti spre stânga;

AVG LEFT < AVG RIGHT => panoul se va roti spre dreapta;

Codul de achiziție al intensității luminoase:

```
#define FILTLER_RANK15

int get_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = ADC_get_value(sensor);
	adc_value = percentage_value(adc_value);
	return adc_value;
}
int get_filtered_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = 0;
	for(char i = 0; i < FILTLER_RANK; i++)
	{
		adc_value += ADC_get_value(sensor);
	}
	adc_value /= FILTLER_RANK;
	adc_value = percentage_value(adc_value);
	return adc_value;
}
int percentage_value(int raw_value)
{
	raw_value = raw_value*((long)100)/1023;
	raw_value = 100 - raw_value;
	return raw_value;
}
```

Cod pentru determinarea intensității mediilor :

```
light_up_left = get_filtered_light_intensity(LS_UP_LEFT);
light_up_right = get_filtered_light_intensity(LS_UP_RIGHT);
light_down_left = get_filtered_light_intensity(LS_DOWN_LEFT);
light_down_right = get_filtered_light_intensity(LS_DOWN_RIGHT);

up_intensity_average = light_up_left + light_up_right;
up_intensity_average >>= 1;

down_intensity_average = light_down_left + light_down_right;
down_intensity_average >>= 1;

left_intensity_average = light_up_left + light_down_left;
left_intensity_average >>= 1;

right_intensity_average = light_up_right + light_down_right;
right_intensity_average >>= 1;
```

3.1.5 Achiziție de date

Pentru procesarea informației folosim microcontrolerul Atmage 32A care conține 8 canale ADC partajate cu PORTA, din care vom utiliza doar 4 pentru conversia luminii.

Un convertor analog-digital (ADC) este un dispozitiv electronic care convertește semnale analogice în semnale digitale, astfel încât acestea să poată fi ușor citite de dispozitivele digitale.

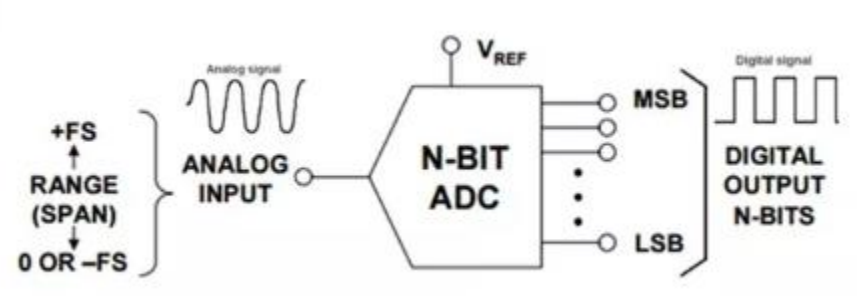


Figura 13.

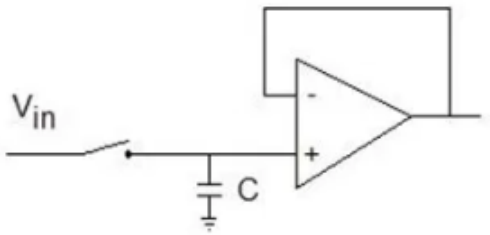
1. Prescaler ADC
- este furnizat pentru a produce o frecvență acceptabilă pentru ADC de la orice frecvență clock.
-frecventa ADC intre 50kHz și 200kHz.

2. Registre ADC
-**ADC Multiplexer Selection Register – ADMUX** – pentru selectarea tensiunii de referință și a canalului de intrare;
-ADC Control and Status Register A – ADCSRA -- pentru control;
-The ADC Data Register – ADCL and ADCH -- aici găsim rezultatul conversiei;

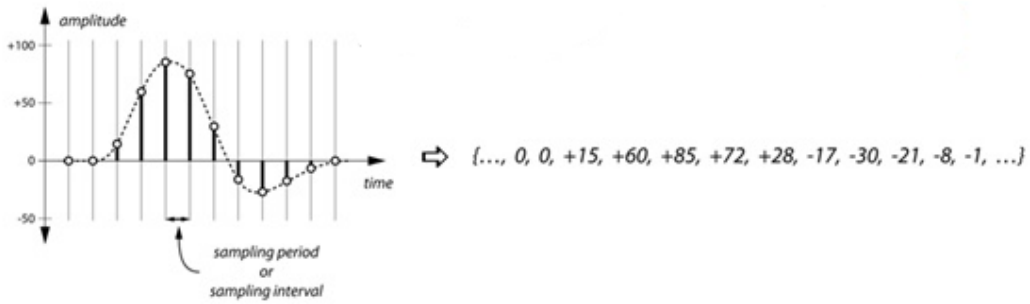
Funcționare:

Procesul de conversie se realizează in doi pași:

1. Eșantionarea (S/H - Sampling and Holding): funcția principală este de a captura imaginile din semnalul analogic și de a menține valoarea acestuia până când ADC-ul poate procesa informația. Menținerea semnalului are beneficiul de a determina cu precizie valoarea conversiei.
   Circuitul pentru procesul de eșantionare:



2. Cuantificare si codificare (Q/E - Quantizing and Encoding)
   Cuantificare: împărțirea semnalului de referință intr-un numar discret de valori și potrivirea acestuia cu valoarea corespunzatoare.
   Codificare: codificarea valorii numerice sub forma unui număr binar.



Codul pentru modulul ADC:

```
#include <avr/io.h>
void ADC_init(void)
{
	DDRC = 0x00;
	// AREF = AVcc
	ADMUX = (1<<REFS0);

	// ADC Enable and prescaler of 128
	// 16000000/128 = 125000
	ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}
// read ADC value
uint16_t ADC_get_value(uint8_t ch)
{
	// select the corresponding channel 0~7
	// ANDing with '7' will always keep the value
	// of 'ch' between 0 and 7
	ch &= 0b00000111;  // AND operation with 7
	ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

	// start single conversion
	// write '1' to ADSC
	ADCSRA |= (1<<ADSC);

	// wait for conversion to complete
	// ADSC becomes '0' again
	// till then, run loop continuously
	while(ADCSRA & (1<<ADSC));

	return (ADC);
}
```

3.2 Motorul DC

3.2.1 Generalități

Motorul DC sau motorul de curent continuu este o mașină electrică reversibilă ce se rotește, realizând conversia energiei electrice absorbite de la generator. Este alcătuit din două elemente constructive principale: stator și rotor. Este potrivit atât pentru acționări electrice de puteri mici și medii, cât și pentru acționări ce nu necesită câmp magnetic de excitație variabil.

Statorul este format dintr-o carcasă de fontă sau de oțel în miezul căreia sunt fixați polii cu bobinajele respective sau fără bobinaje în cazul magneților permanenți. În părțile laterale ale carcasei sunt situate cele două scuturi ce poartă lagărele.

Rotorul este confecționat din tole de oțel electrotehnic, fixate pe arbore, având crestături periferice în care se află laturile active ale bobinelor indusului. Rotorul posedă un colector cilindric din lamele de cupru, izolate, montate în coadă de rândunică pe un butuc al arborelui.

Motorul de curent continuu are pe stator polii magnetici și bobinele polare concentrate care creează câmpul magnetic de excitație. Pe axul motorului este situat un colector ce schimbă sensul curentului prin înfășurarea rotorică astfel încât câmpul magnetic de excitație să exercite în permanență o forță față de rotor.

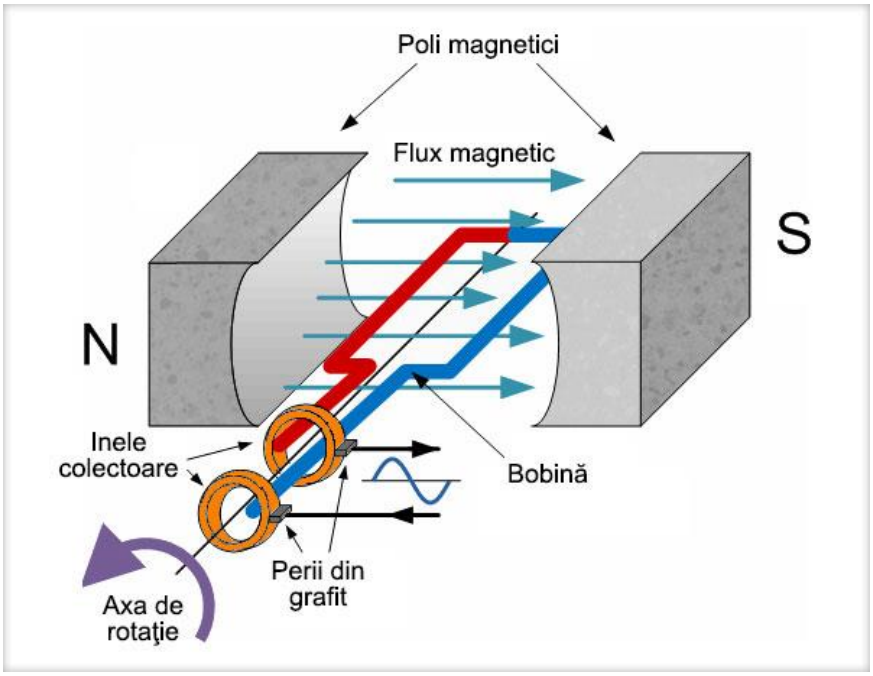


Figura 14.

Principiu de funcționare:

În momentul în care curentul trece prin rotorul unui motor DC, se generează un câmp magnetic. Acesta generează la rândul său o forță electromagnetică care face ca rotorul să se rotească lucru care induce o tensiune în bobinajul rotorului. Rotorul se deplasează în câmpul magnetic de excitație până când polii rotorici se aliniază în dreptul polilor statorici opuși. În același moment, colectorul schimbă sensul curenților astfel încât polaritatea se inversează și rotorul va continua deplasarea până la următoarea aliniere a polilor magnetici.

Viteza cu care un motor de curent continuu funcționează, depinde de puterea câmpului magnetic care acționează asupra rotorului, cât și de curentul rotorului.

Tipuri de motoare de curent continuu:

1.Motor cu excitație independentă - unde înfășurarea statorică și înfășurarea rotorică sunt conectate la două surse separate de tensiune;
2.Motor cu excitație paralelă - unde înfășurarea statorică și înfășurarea rotorică sunt legate în paralel la aceeași sursă de tensiune;
3.Motor cu excitație serială - unde înfășurarea statorică și înfășurarea rotorică sunt legate în serie;
4.Motor cu excitație mixtă - unde înfășurarea statorică este divizată în doua înfășurări, una conectată în paralel și una conectată în serie;

Aplicații:

- In domeniul electric urban sau feroviar (tramvaie, locomotive etc);
- In aplicații casnice de puteri mici și viteze mari de rotație (aspirator, mixer, râșnite de cafea etc);
- In aplicații industriale (utilaje, pompe, ventilatoare etc);
- Aplicații în care este necesară variația de turație pe un domeniu larg;

Puncte forte:

- Eficiență deosebita;
- Consum minim de energie, cuplu de pornire mare;
- Zgomot și vibrații reduse;
- Fiabilitate și întreținere ușoară;

3.2.2 Schematic motor DC

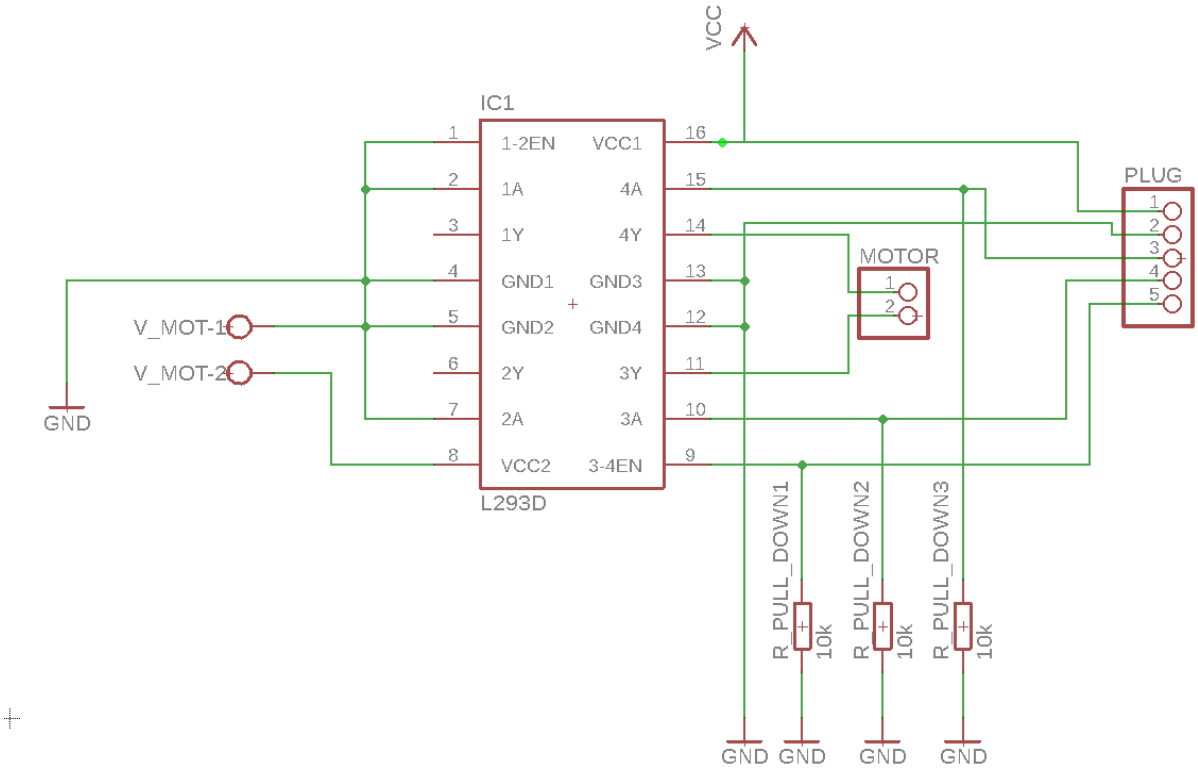


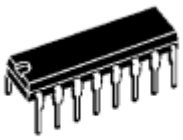
Figura 15.

Specificații:

- Tensiune de alimentare: 24 V;
- RPM = 1.85;

Rolul motorul DC în această aplicație este de a roti panoul către cea mai puternică sursă de lumină. Pentru control am folosit integratul L293D care de altfel reprezintă și driver pentru motor.

3.2.3 Integratul L293D



Powerdip (12+2+2)

Figura 16.

L293D este un circuit integrat care poate fi folosit pentru controlul simultan a două motoare mici.

Acest dispozitiv este potrivit pentru utilizarea în comutarea aplicațiilor la frecvențe de până la 5 kHz.

BLOCK DIAGRAM

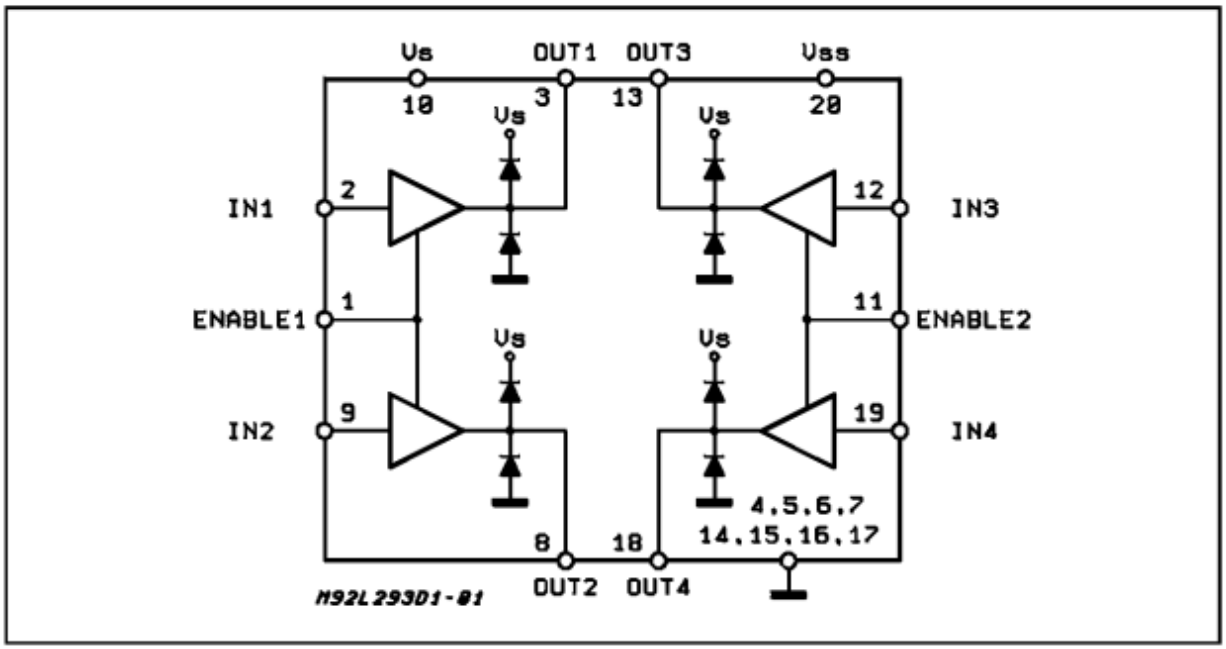


Figura 17.

In mod obișnuit, integratul L293D este o punte H cu o singură intrare electrică care activează ambele comutatoare într-o pereche, făcând astfel mai ușor de utilizat. O altă funcție a unei punți H este că vă permite să controlați și să blocați un motor.

Puntea H conține un motor controlat de 4 comutatoare. Comutatoarele mecanice sunt în realitate tranzistoare care acționează ca întrerupătoare electrice. Această configurare ne permite să controlăm motorul și să îl deplasăm înainte sau înapoi.

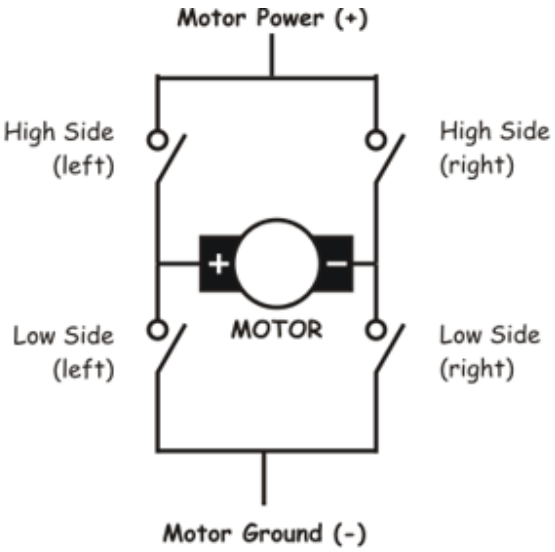


Figura 18.

Funcții pini pentru integrat:

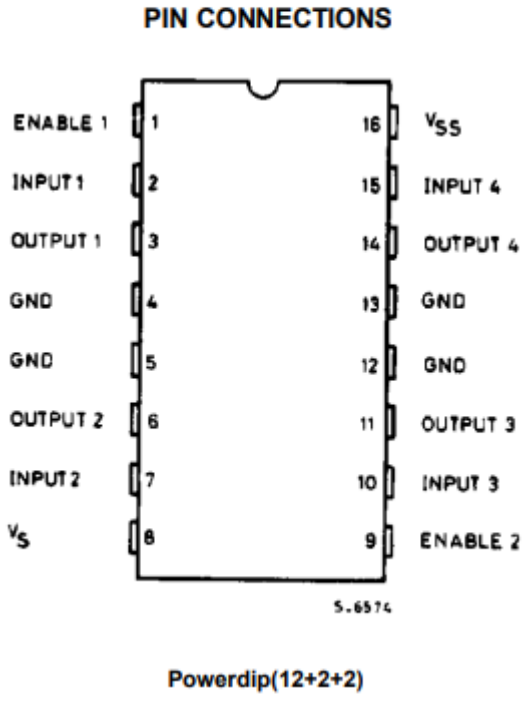


Figura 19.

Cod sursa de control pentru driver-ul L293D:

```
Void init_l293d_control(void)
{
	L293D_DDR |= ((1 << L293D_HB2_DIRECTION_LEFT) | (1 <<
L293D_HB2_DIRECTION_RIGHT) | (1 << L293D_HB2_ENABLE));
	L293D_PORT &= ~((1 << L293D_HB2_DIRECTION_LEFT) | (1 <<
L293D_HB2_DIRECTION_RIGHT) | (1 << L293D_HB2_ENABLE));
}
void l293d_hb2_rotate_left(void)
{
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_LEFT;
	L293D_SET_HB2_ENABLE;
}
void l293d_hb2_rotate_right(void)
{
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_RIGHT;
	L293D_SET_HB2_ENABLE;
}
void l293d_hb2_stop(void)
{
	L293D_CLEAR_HB2_ENABLE;
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
}
```

3.3 Motorul Stepper

3.3.1 Generalități

Un motor pas cu pas este un motor electric brushless, sincron care transformă impulsurile digitale în rotație mecanică a arborelui. Acesta poate împărți o rotație completă de 360˚ a axului într-un număr discret de pași, în multe cazuri 200 și trebuie să trimită câte un impuls separat pentru fiecare.

Fiecare impuls determină rotirea motorului cu un unghi precis iar dacă frecvența crește, mișcarea pasului se transformă în rotație continuă de unde rezultă că viteza de rotație este direct proporțională cu frecvența impulsurilor.

Datorită costului redus, fiabilității ridicate, cuplului ridicat la viteze reduse și a unei construcții simple, robuste care funcționează în aproape orice mediu, motoarele pas cu pas sunt utilizate zi de zi atât în aplicații industriale cât și comerciale.

Motoarele pas cu pas sunt deseori folosite în aplicații ce necesită un control precis al mișcării.

Clasificare:

A. Tipuri constructive:

Există trei tipuri de motoare pas cu pas:

a. Motor pas cu pas cu reluctanță variabilă:

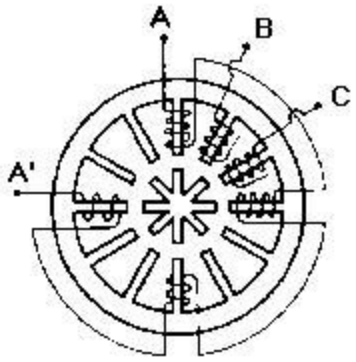


Figura 20.

Caracteristici:

- Rotor construit din fier moale și cu înfășurările pe stator;
- Realizează mișcări de rotație ale axului intre 5˚ și 15˚;
- Nu își poate menține poziția axului pe durata lipsei tensiunii de alimentare a bobinelor ;

b. Motor pas cu pas cu magneți permanenți:

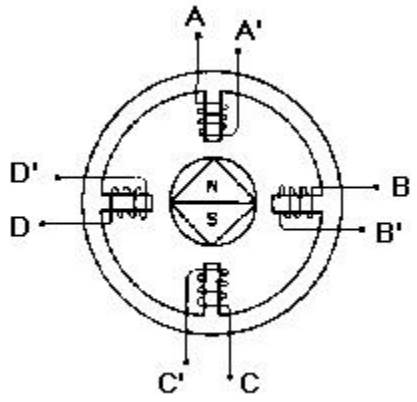


Figura 21.
Caracteristici:

- Rotorul este realizat din magneți permanenți fără dinți, magnetizați perpendicular pe axa care separă polii;
- Înfășurările sunt realizate pe stator;
- Realizează mișcări ale axului de 45˚ sau 90˚ cu un cuplu ridicat dar la viteze de rotație mici;

c. Motor pas cu pas hibrid:

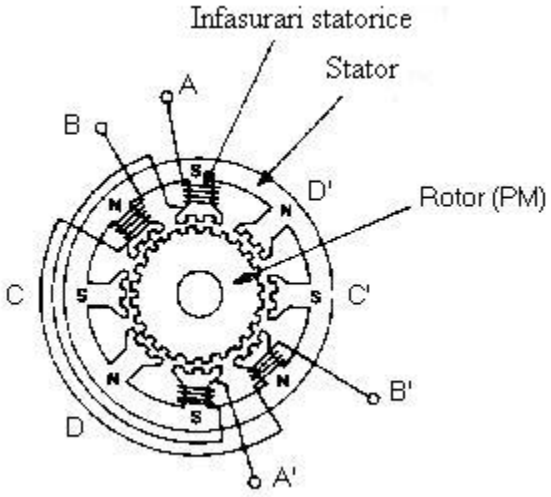


Figura 22.
Caracteristici:

- Funcționează la viteze mari;
- Cuplu dinamic mare;
- Cuplu care asigura menținerea fixă a poziției;

B. După tipul înfășurărilor din stator:

a. Motorul pas cu pas unipolar:

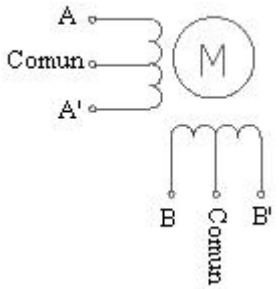


Figura 23.

Caracteristici:

- Statorul e compus din câte două bobine pe pol;
- Are nevoie doar de un element de comutație pentru fiecare bobina;
- Pentru a obține mișcarea de rotație a axului motorului, sensul curentului prin bobinele din statorul motorului nu trebuie schimbat;

b. Motorul pas cu pas bipolar:

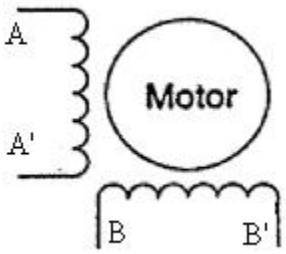


Figura 24.

Caracteristici:

- Are o înfășurare pe pol;
- Trebuie schimbat sensul curentului prin bobine, astfel avem nevoie de un circuit de comandă mai complex, de obicei o punte H ;

c. Motorul pas cu pas “8 - lead”:

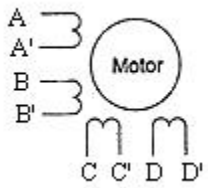


Figura 25.

Caracteristici:

- Au 8 fire, adică 4 bobine (2 pe pol) cu ambele capete accesibile la exterior;

Avantaje:

- Unghiul de rotație al motorului este proporțional cu impulsul de intrare;
- Răspuns excelent la pornire / oprire / inversare;
- Răspunsul motoarelor la impulsurile digitale de intrare oferă control în buclă deschisă, ceea ce face ca motorul să fie mai simplu și mai puțin costisitor de controlat.

3.3.2 Schematic motor Stepper

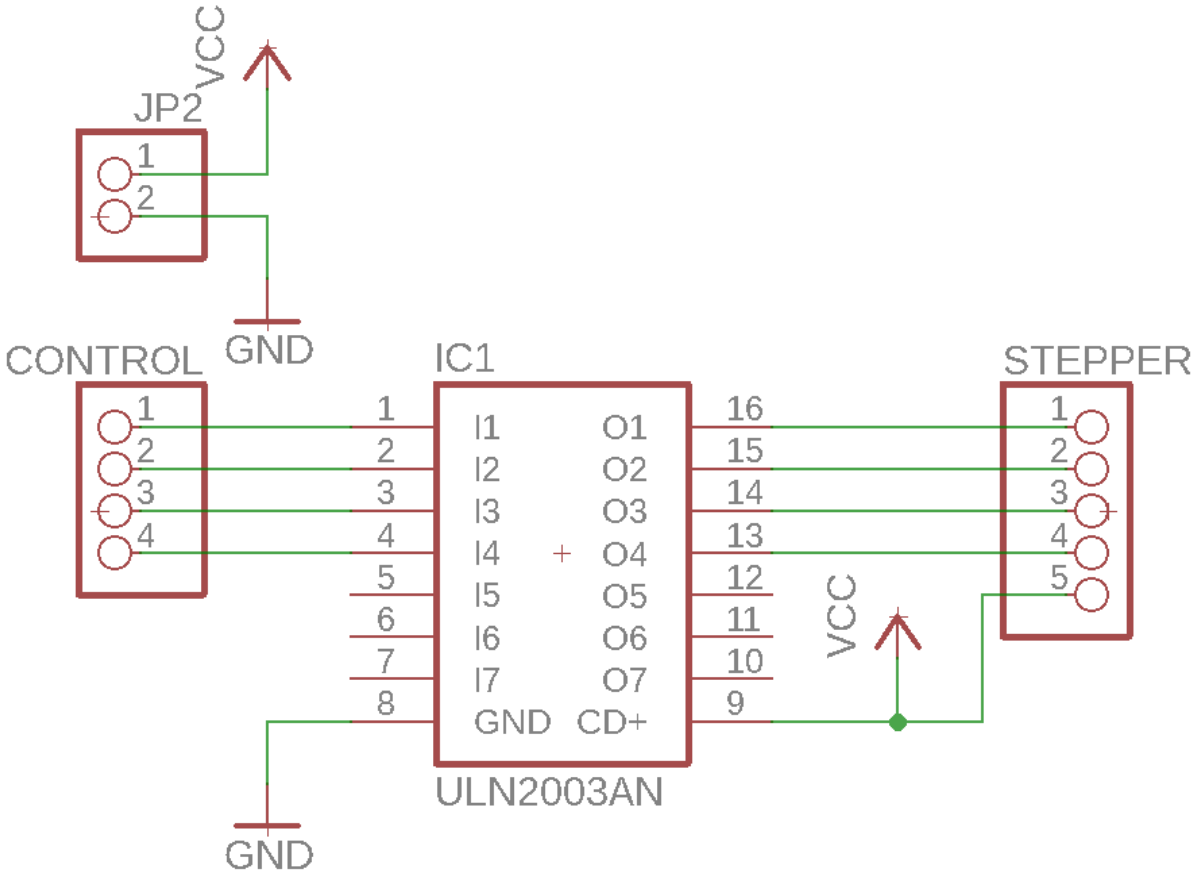


Figura 26.

Specificatii:

- Tensiune de alimentare recomandată: 5V;
- Număr de faze: 4;
- Reducție aproximativă: 1:64;
- 64 pași/rezoluție;
- Rezistență/fază: 50 ohmi;
- Cuplu minim: 34.3 mN*m;
- Grad izolație: A.

Rolul motorului Stepper în această aplicație este de a inclina panoul către cea mai puternică sursă de lumină.

Pentru comanda am folosit un driver ce se bazează pe circuitul integrat ULN2003, ce conține mai mulți tranzistori Darlington în configurație.

3.3.3 Integratul ULN2003AN

ULN2003AN este un circuit integrat format dintr-un sir de 7 tranzistori NPN Darlington, disponibil în mai multe forme pe 16 pini. Are capacitatea de a susține pe fiecare ieșire un curent de maxim 500mA și voltaj de pana la 50V.

Tranzistorul Darlington:

Un tranzistor Darlington este format din două tranzistoare obișnuite legate în serie, pentru a putea controla un curent cât mai mare, cu un curent de comandă cât mai mic.

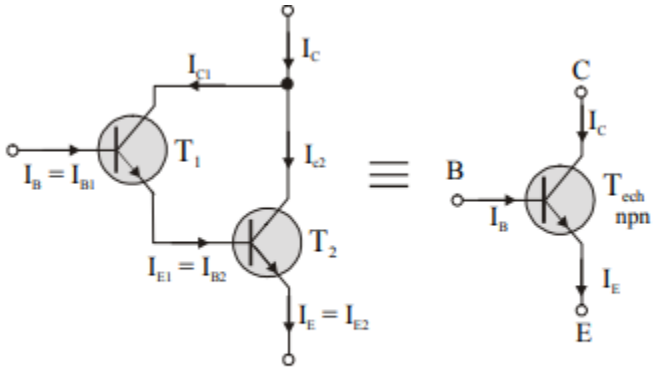


Figura 27.

Schema interna (fiecare legătură a tranzistorului Darlington):

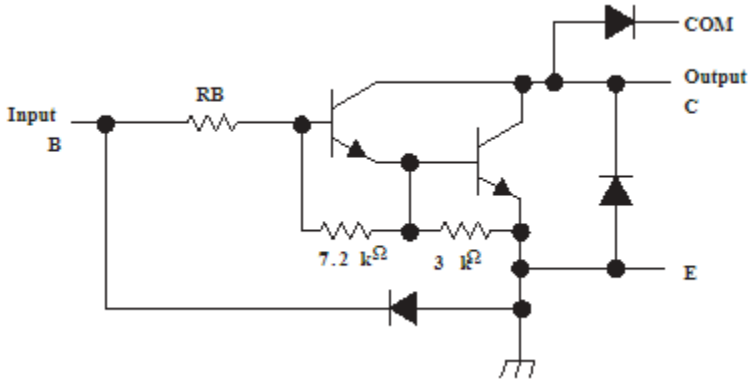


Figura 28.

Aplicații ale integratului ULN2003AN:

- acționări de motorașe și steppere;
- amplificarea ieșirilor microcontrolerelor de la 3.3 sau 5V la pana 50V;
- acționarea releelor și contactoarelor;
- comutarea de lămpi de informare și avertizare;

Diagramă logica:

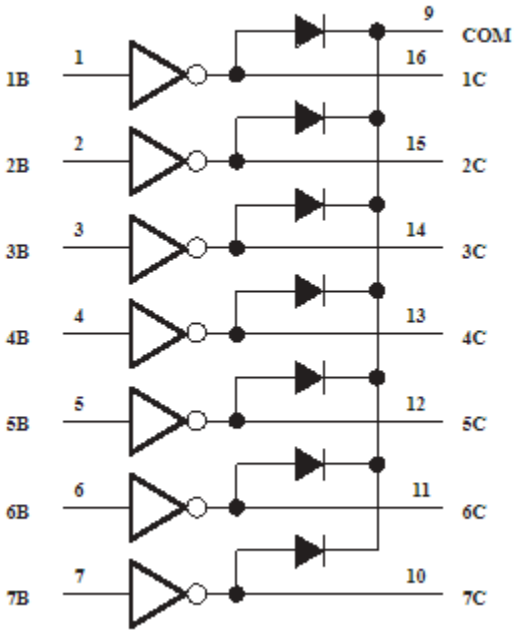


Figura 29.

Cei 16 pini ai integratului sunt distribuiți astfel:

- 7 pini de intrare;
- 7 pini de ieșire;
- 1 pin Ground ;
- 1 pin COM;

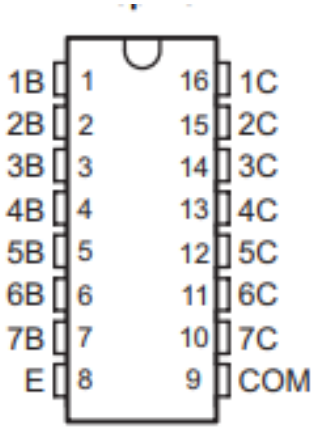


Figura 30.

Cod sursă de control pentru driver-ul ULN2003AN:

```
Void init_unipolar_control(void)
{
	UNIPOLAR_01_DDR |= ((1 << UNIPOLAR_01_STEP_1) | (1 << 
UNIPOLAR_01_STEP_2) | (1 << UNIPOLAR_01_STEP_3) | (1 << 
UNIPOLAR_01_STEP_4));
	UNIPOLAR_01_CLEAR_STEP_1;
	UNIPOLAR_01_CLEAR_STEP_2;
	UNIPOLAR_01_CLEAR_STEP_3;
	UNIPOLAR_01_CLEAR_STEP_4;
	UNIPOLLAR_01_CURRENT_STEP = 4;
}
void unipolar_01_step_forward(unsigned char current_step)
{
	switch(current_step)
	{
		case 1:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_SET_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 2;
			break;
		}
		case 2:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_SET_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 3;
			break;
		}
```

```
                case 3:
                {
                        UNIPOLAR_01_CLEAR_STEP_1;
                        UNIPOLAR_01_CLEAR_STEP_2;
                        UNIPOLAR_01_CLEAR_STEP_3;
                        UNIPOLAR_01_SET_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 4;
                        break;
                }
               case 4:
                {
                        UNIPOLAR_01_SET_STEP_1;
                        UNIPOLAR_01_CLEAR_STEP_2;
                        UNIPOLAR_01_CLEAR_STEP_3;
                        UNIPOLAR_01_CLEAR_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 1;
                         break;
                }
        }
        _delay_ms(3);
}
```

3.4 Unitatea centrala: microcontroller-ul

3.4.1 Generalități

ATmega32A este un microcontroler CMOS cu putere redusă bazat pe 8 biți și cu o arhitectură RISC îmbunătățită. Acesta atinge transferuri apropiate de 1MIPS/MHz prin executarea unor instrucțiuni într-un singur ciclu de ceas și astfel ajută la optimizarea dispozitivului pentru consumul de energie față de viteza de procesare.

Figura 31.

Caracteristici:

Număr pini	44
Tip circuit integrat	Microcontroler AVR
Capacitate memorie EEPROM	1kB
Capacitate memorie SRAM	2kB
Capacitate memorie Flash	32kB
Număr canale PWM	4
Număr timere 8biti	2
Număr timere 16biti	1
Temperatură de lucru	-55...125°C
Frecventă funcționare	16MHz
Tensiune alimentare	2.7...5.5V
Adaptoare A/D 10biti	8
Număr comparatoare	1

Interfață:

- I2C
- JTAG
- SPI
- UART

Caracteristici circuite integrate:

- watchdog
- rezonator 32kHz pentru RTC

Arhitectură avansată RISC:

- 131 Instrucțiuni puternice - majoritatea ciclurilor cu un singur ceas;
- 32 × 8 registre de lucru cu scop general;
- Operație complet statică;
- Până la 16MIPS la 16MHz;
- On-chip multiplicator de cicluri;

Segmente mari de memorie nevolatilă de rezistență:

- 32Kbyte de memorie de program Flash auto-programabilă în sistem;
- 1024 biți EEPROM;
- 2Kbytes SRAM intern
- Cicluri de scriere / ștergere: 10.000 bliț / 100.000 EEPROM;
- Reținerea datelor: 20 de ani la 85 ° C / 100 de ani la 25 ° C

Interfața JTAG:

- Programare Flash, EEPROM, siguranțe și blocare prin interfața JTAG;

I / O și pachete:

- 32 linii I / O programabile;
- PDIP cu 40 de pini, TQFP cu 44 de conductori și QFN / MLF cu 44 de cadre;

Consum de energie la 1MHz, 3V, 25 ° C:

- Activ: 0.6mA;
- Mod inactiv: 0,2 mA;
- Mod de pornire: <1µA;

3.4.2 Funcții pini:

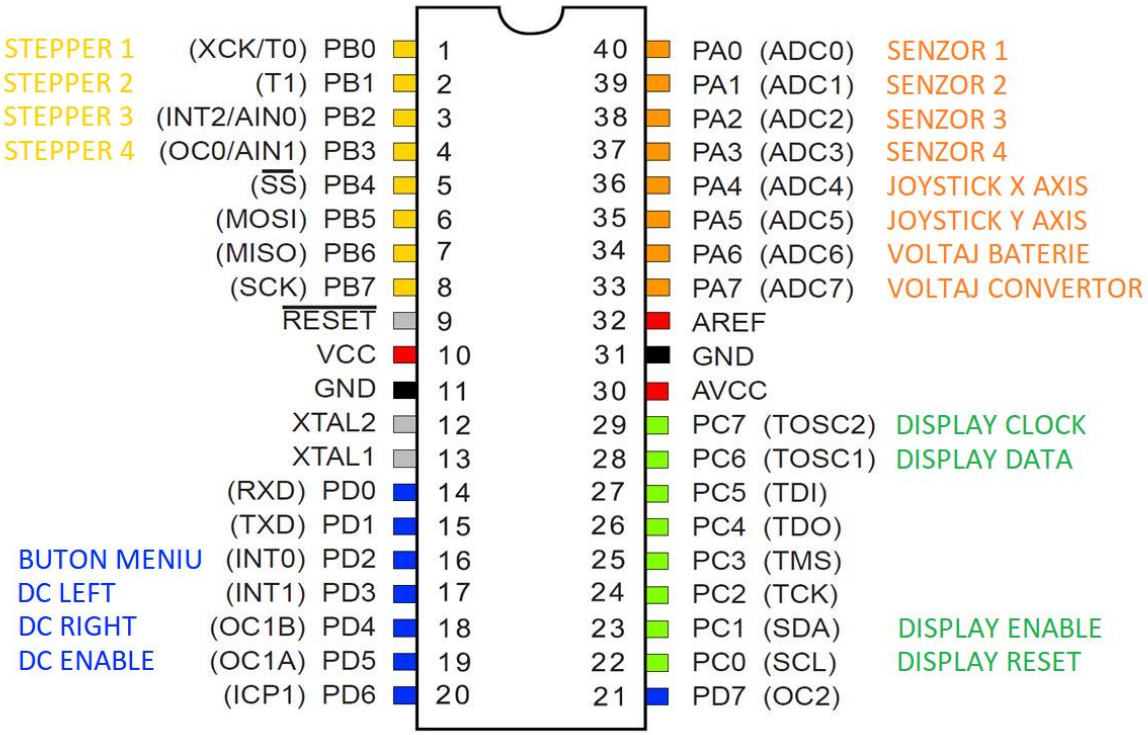


Figura 32.

Vcc - Tensiune de alimentare digitală.
GND - Masă.
PortA (PA7:PA0)
- Portul A servește ca intrări analogice la convertorul A /D.
- Portul A servește și ca port I / O bidirecțional pe 8 biți, dacă convertorul A / D nu este utilizat.

Port B (PB7:PB0)
Port C(PC7:PC0)
PortD (PD7:PD0)
- acestea sunt porturi I / O bidirecționale pe 8 biți cu rezistențe pull-up pentru fiecare bit.

RESET - Resetare intrare. Dacă nivelul scade sub lungimea minimă a impulsului, se va genera un semnal de reset chiar dacă ceasul nu funcționează.
XTAL1 - intrarea în circuitul de funcționare a ceasului intern.
XTAL2 - ieșire de la amplificatorul oscilator inversor.
AVCC - pinul de tensiune de alimentare pentru portul A și convertorul A / D.
AREF - pinul analogic de referință pentru convertorul A / D.

Schema bloc:

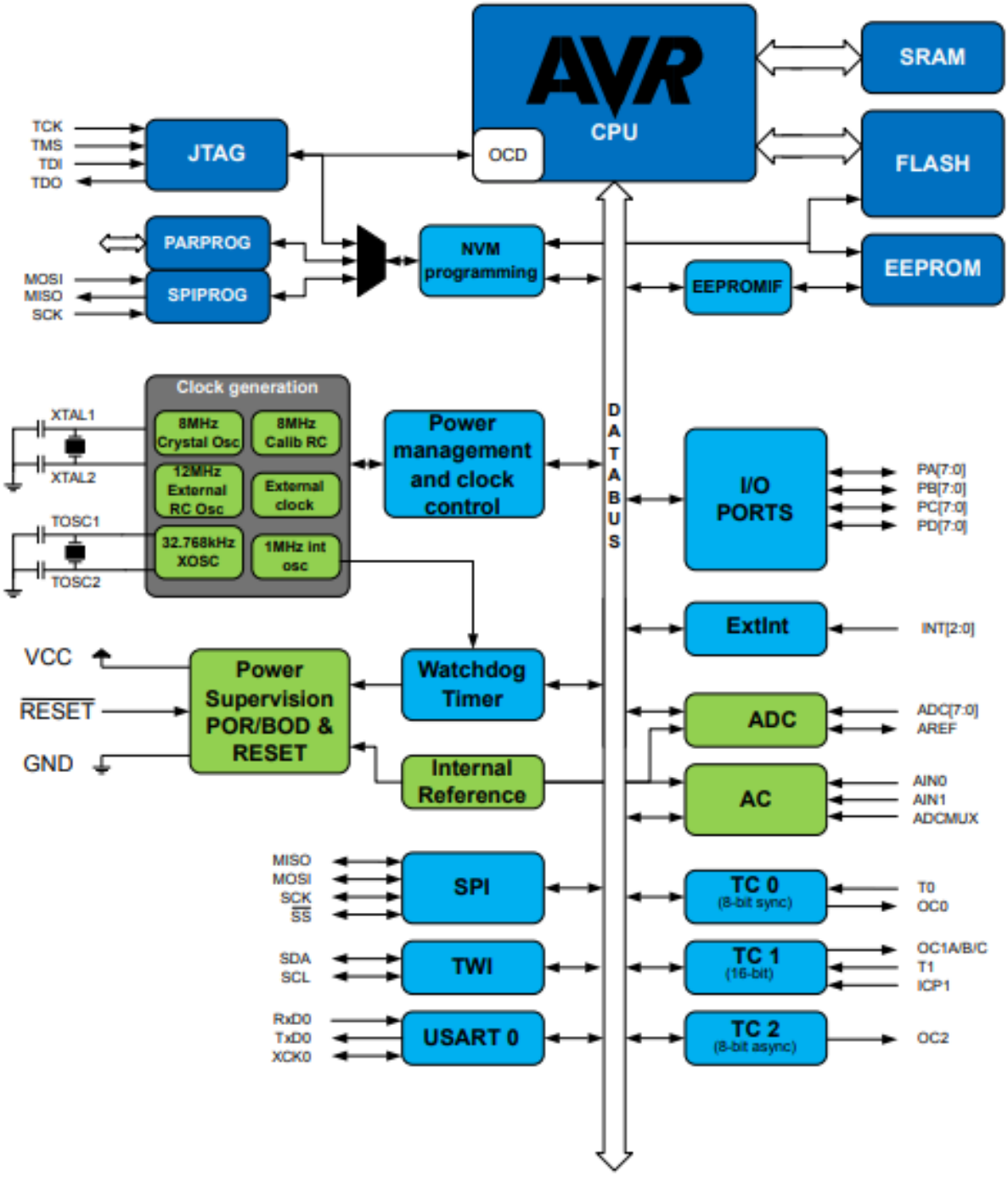


Figura 33.

3.4.3 Schematic:

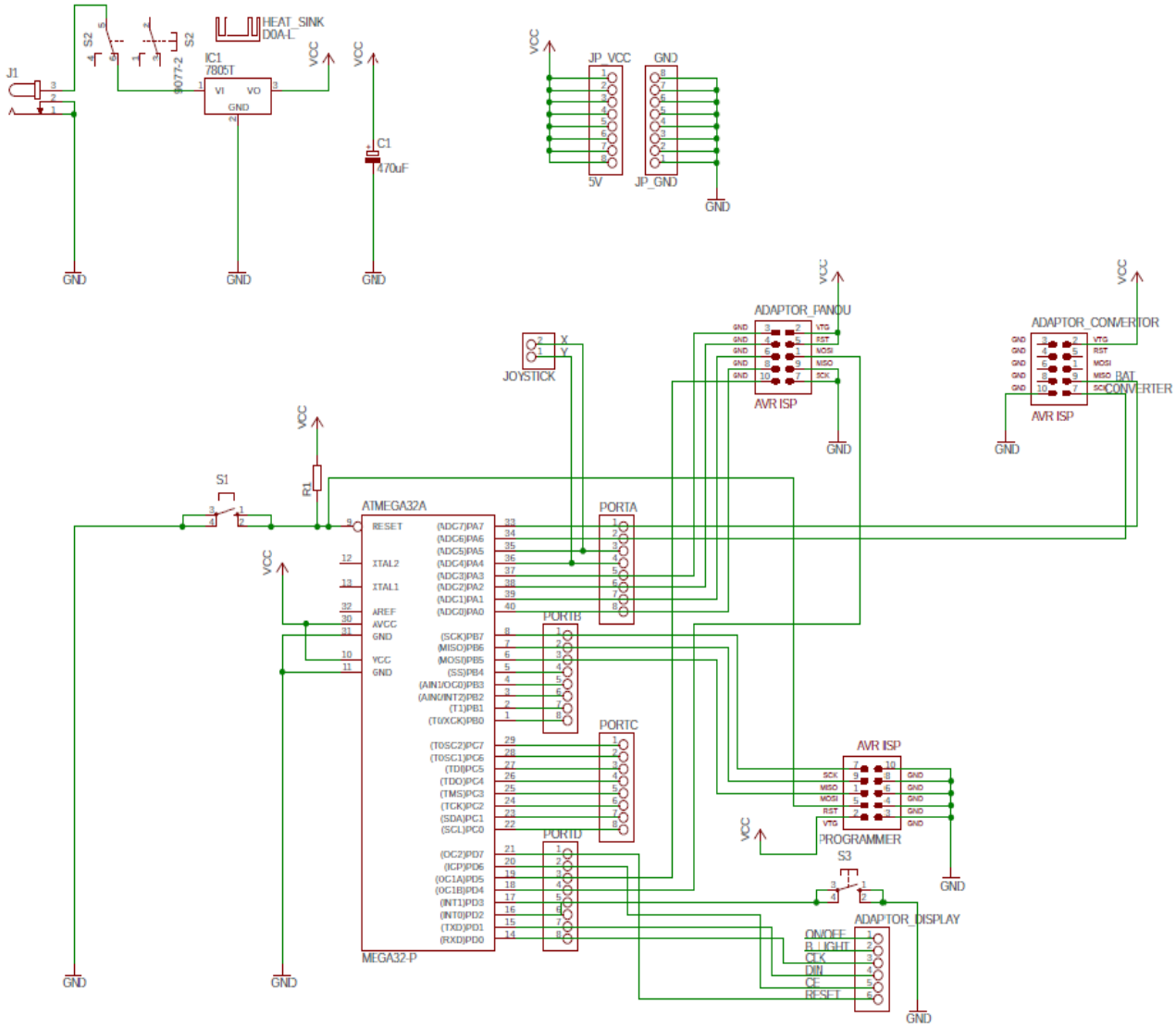


Figura 34.

Capitolul 4. Analize și rezultate

4.1 Simulări control directie motoare

4.1.1 Motor DC

Dupa cum știm, motorul de curent continuu este controlat cu ajutorul integratului L293D, reprezentat de o punte H.

Circuitul permite controlul direcției, pornire/oprire, controlul vitezei si frână:

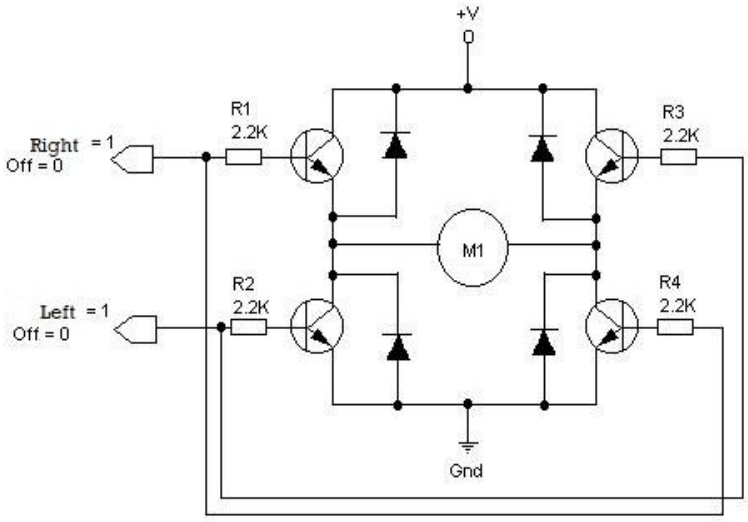


Figura 35.

Tranzistoarele sunt comutatoare electronice, ele permit să activați tensiuni mari utilizând un curent foarte mic. Fiecare pereche de tranzistoare este conectată la un pin de pe microcontroler și controlează polaritatea curentului alimentat de motor.

Diodele din circuit au rolul de a proteja tranzistorul.Când motorul își schimba direcția, bobinele de sârmă din interior acționează ca un generator si produc curent. Acest curent se deplasează inapoi in circuit, la tranzistor. Deoarece tranzistorul permite trecerea curentului intr-un singur sens, curentul care se deplasează în sens invers în circuit poate distruge tranzistorul. Astfel intra în acțiune dioda care permite curentului să ocolească tranzistorul și să ajungă în siguranță înapoi la baterie.

În figurile 36, 37 sunt reprezentate semnalele care se aplică integratului L293D pentru controlul direcțiilor motorului.

Tranziția din „0” logic in „1” logic aplicată pinilor Right și Enable va face ca motorul să se miște spre dreapta, idem și pinilor Left si Enable pentru mișcarea spre stanga.

Unde: „0” logic = 0V;

„1” logic = 5V;

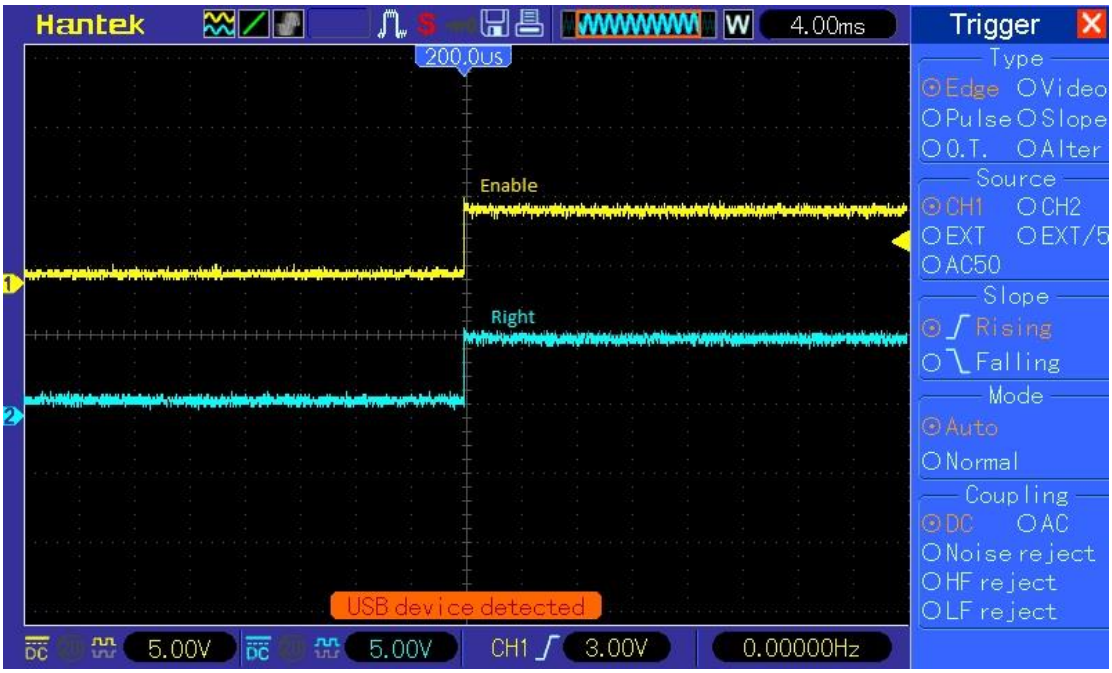


Figura 36.

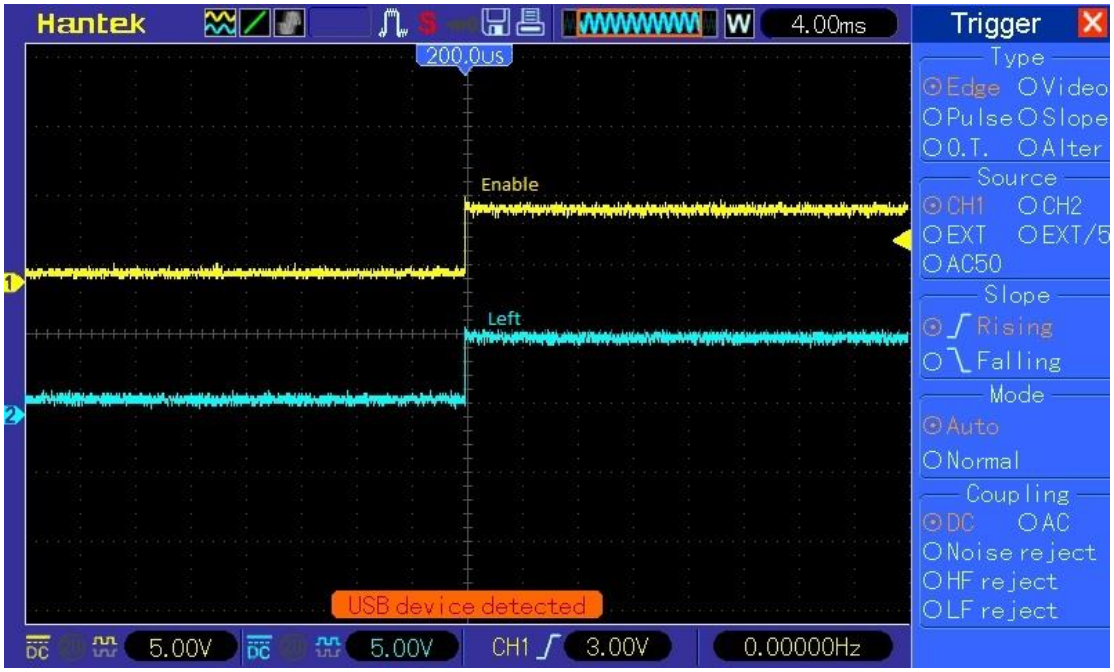
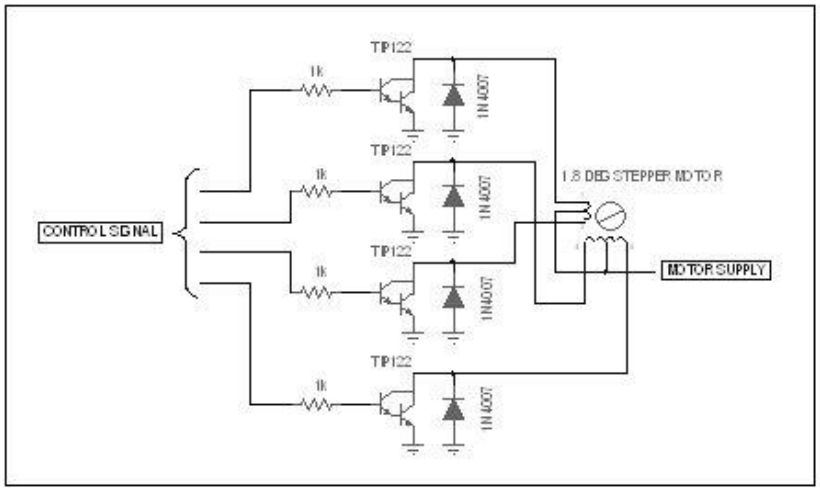


Figura 37.

4.1.2 Motor Stepper

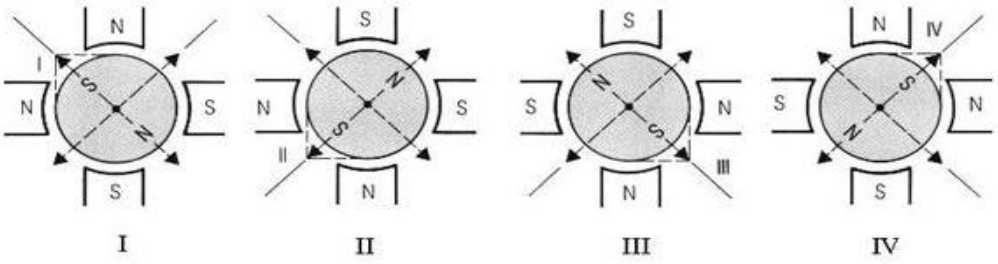
Motorul pas cu pas este controlat cu ajutorul integratului ULN2003AN care este si driver pentru motor. ULN2003AN este format din tranzistoare Darlington.

s

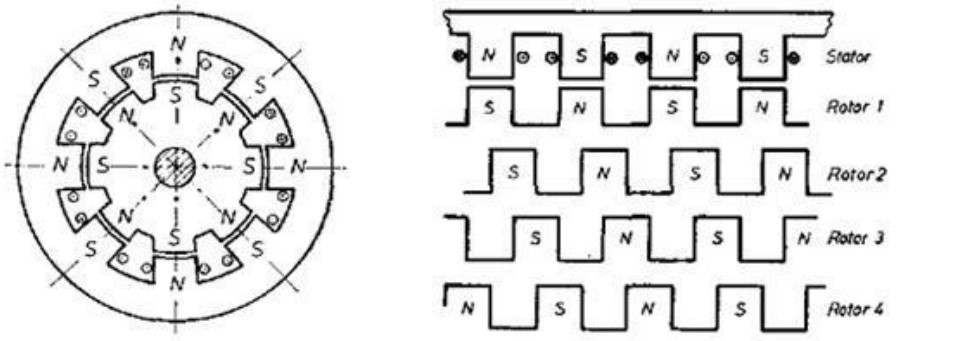


Un motor pas cu pas permite obţinerea unor rotaţii cu exact acelaşi unghi, corespunzător unui pas. Indiferent de principiul de funcţionare al motorului, comanda acestuia se realizează prin comutarea succesivă a fazelor înfăşurărilor.

Motorul pas cu pas in timpul unui rotatii complete:



Semnale de comanda:



In figurile 38, 39 sunt reprezentate semnale activate pe rand si aplicate pe intrarea integratului ULN2003AN pentru controlul inainte/inapoi al motorului.

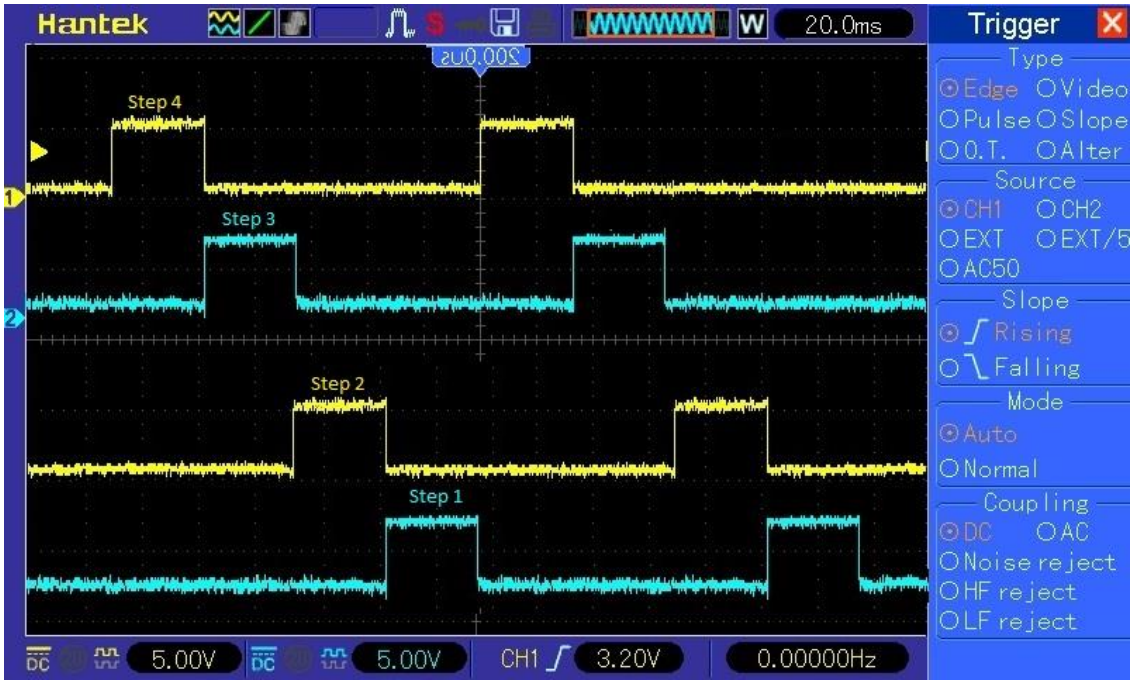


Figura 38.

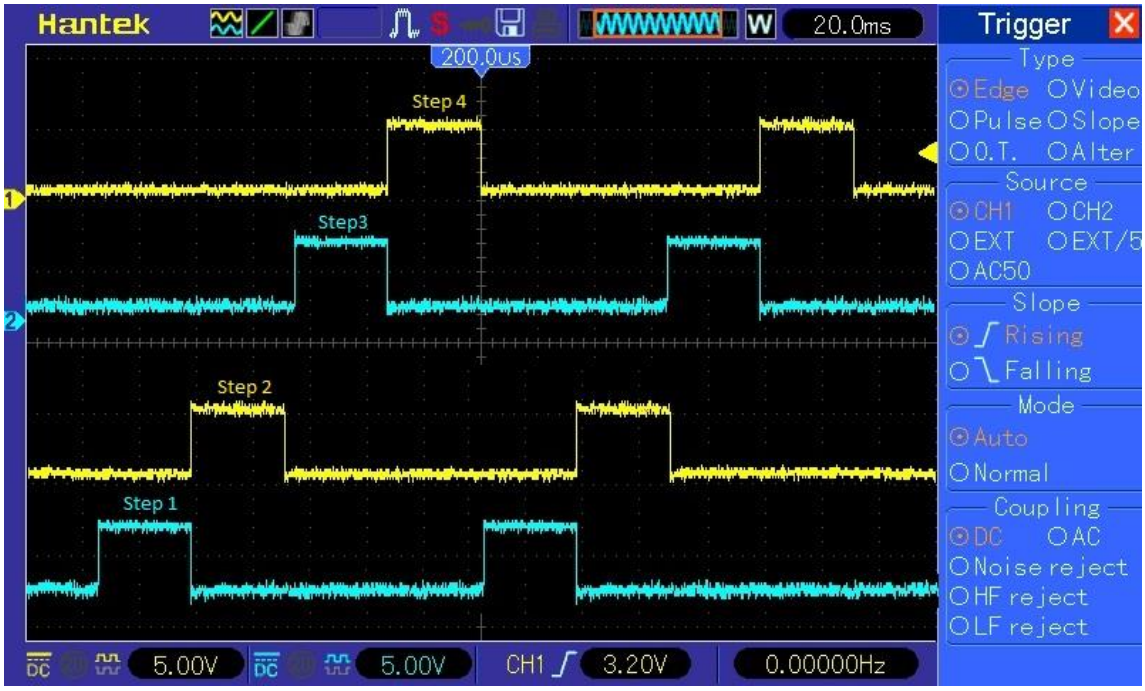


Figura 39.

4.2 Preluarea luminii

In figura 40 avem fotorezistentele cu ajutorul carora captam lumina si formam divizorul rezistiv impreuna cu potentiometrele si rezistentele fixe din figura 41.

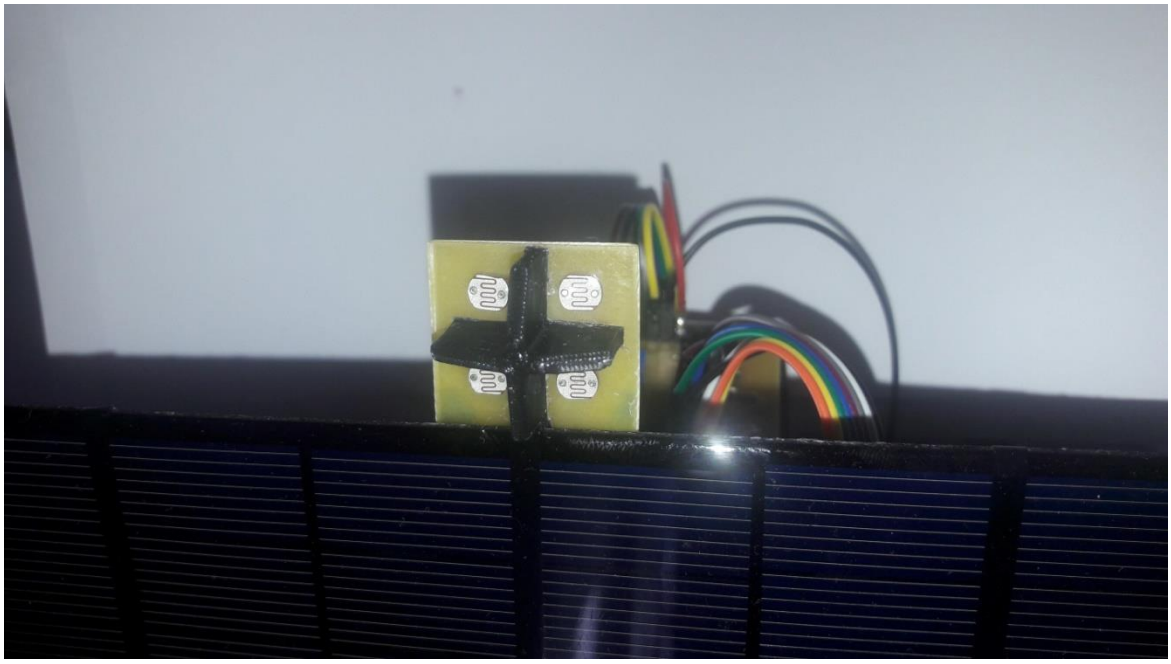


Figura 40.

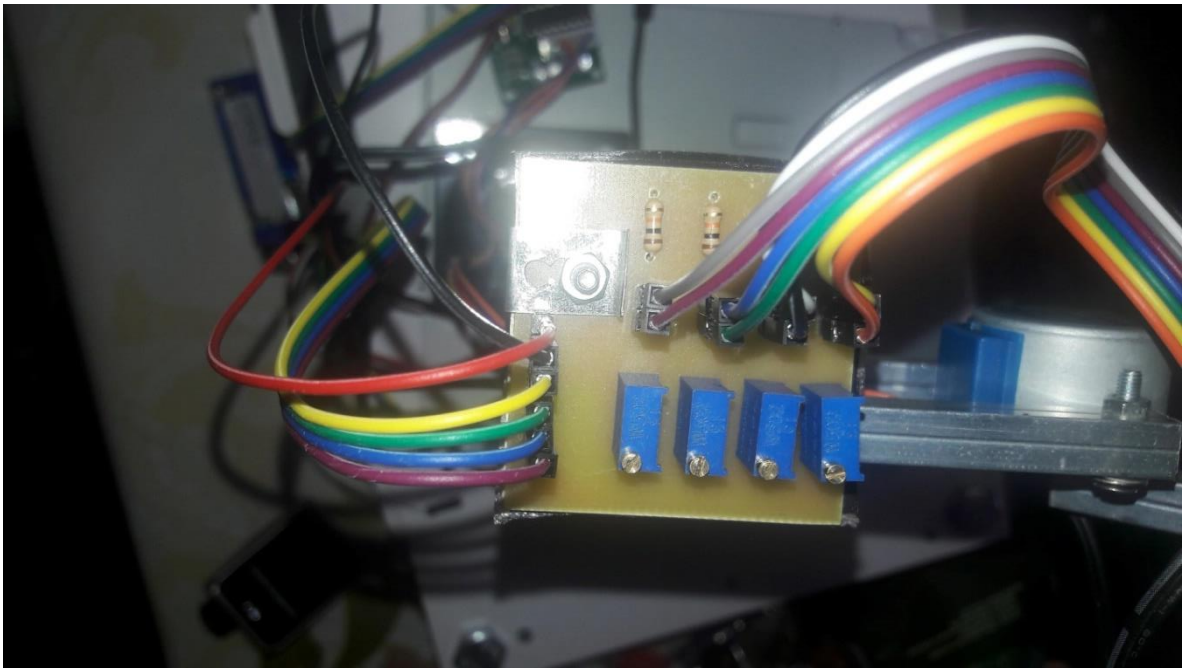


Figura 41.

In figura 41 este prezentata partea practica a proiectului in care se pot vedea componentele de baza:

- Bateria;
- Convertorul;
- Panoul solar;
- Cutia de comanda in care se afla microcontrolerul;
- Display;
- Butoane;
- Joystick;

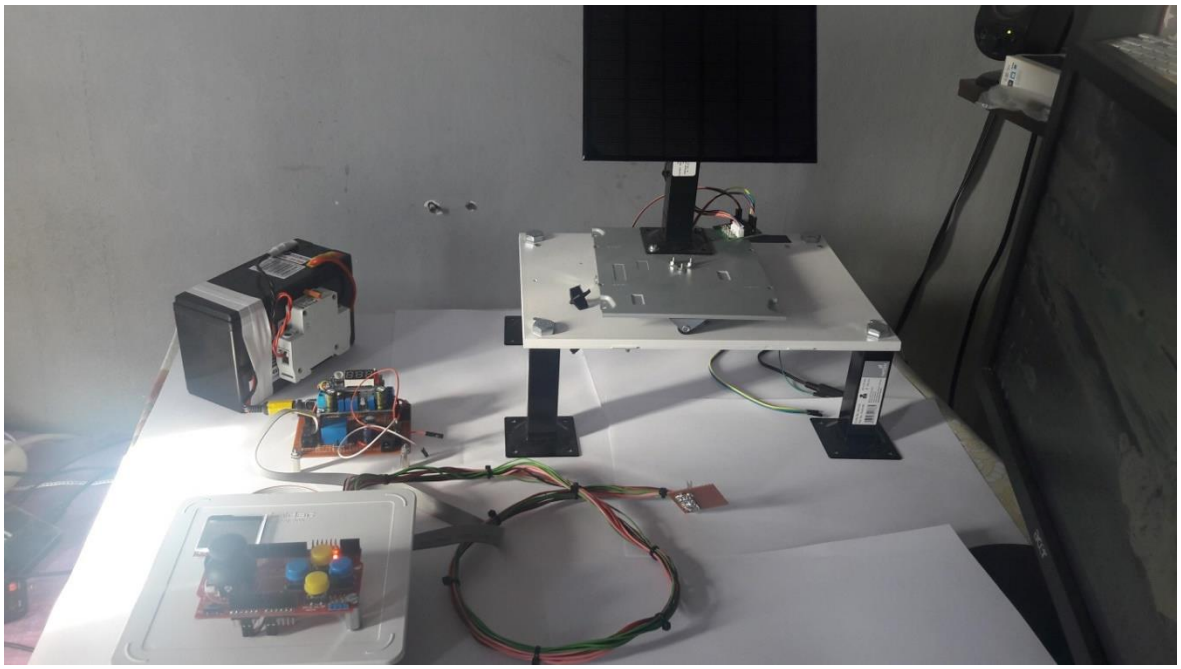


Figura 41.

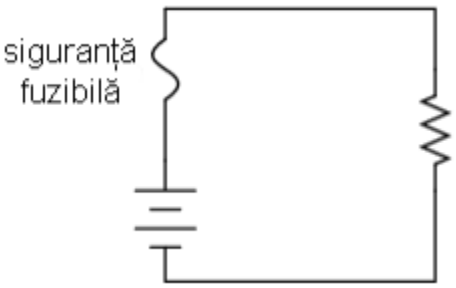
4.3 Măsuri de siguranță

Ca masură de siguranță am folosit o siguranță fuzibilă cu rol de protecție la supracurent. O siguranță fuzibilă nu este altceva decat un fir conductor scurt, proiectat astfel încât, in situația unui curent excesiv, acesta să se separe prin topire, deschizând circuitul.

Mod de conectare:

Siguranțele fuzibile se conectează in serie cu elementul de circuit care necesită o protectie la supracurent, astfel încât, în cazul deschiderii circuitului prin topirea siguranței, curentul prin toate componentele să scadă la zero.

Schema:



Capitolul 5

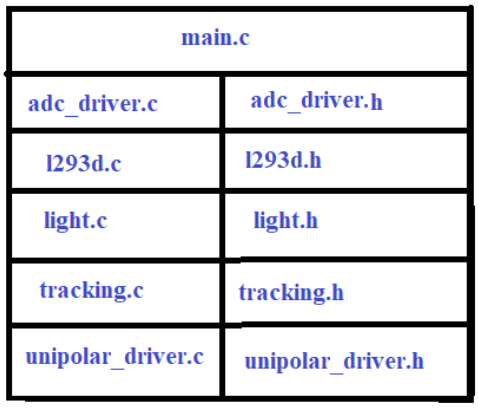
Concluzie

Odată cu avansarea tehnologiei în sfera modulelor fotovoltaice, a dezvoltării unor celule cu randamente de conversie ce vor atinge,sau chiar depăși pragul de 50%, utilizarea panourile fotovoltaice prezintă încă aspecte ce merită să fie luate în considerare.

Proiectul dezvoltat are ca scop principal încărcarea unei baterii de 12V cu ajutorul unui panou solar, formând astfel un circuit închis mai exact un sistem autonom. Conversia energiei solare in energie electrică reprezintă o tehnologie de viitor.

În urma realizarii acestei lucrări am dezvoltat noi cunostinte, atât pe partea teoretică cât si pe partea practică. Am învatat să folosesc diferite aplicații precum Vectary, cu ajutorul căreia am construit scutul, Eagle, folosit pentru creearea schemelor, Atmel Studio, pentru scrierea, construirea și depanarea codului de asamblare.

Ca și parte software lucrarea este împărțita în mai multe module, fiecare modul fiind construit dintr-un fisier .c si un fisier .h, acestea fiind integrate intr-o funcție principală numită “main.c”.



Daca te întrebi „La ce se poate folosi această aplicatie în viața de zi cu zi ?”

O parte benefică a acestei aplicații este aceea ca in urma procesului de convesie putem încarca telefonul, sau avem mereu o baterie de rezervă. Astfel am instalat un port USB care ne dă la iesire o tensiune de 5V, necesară încarcarii unui telefon.

Bibliografie

http://www.uvl.ro/index.php?option=com_content&view=article&id=7:controler-panou-solar&catid=3:proiecte&Itemid=53
http://energie-verde.ro/produse/panouri-fotovoltaice-2/
http://www.solar-depot.ro/Controlere-de-Incarcare
https://www.sistemepanourisolare.ro/blog/35_istoria-energiei-solare
http://www.cineainventat.ro/panoul-solar/
https://www.scribd.com/document/149873225/Lucrare-de-Diploma-panouri-fotovoltaice
http://www.creeaza.com/tehnologie/electronica-electricitate/ENERGIA-SOLARAPANOURI-SOLARE282.php
https://www.microchip.com/avr-support/atmel-studio-7
https://www.wikipedia.org/
https://toppanourisolare.ro/panouri-solare/functionarea-panourilor-solare.html
http://www.naturenergy.ro/efectul-fotovoltaic-160221.htm#.WtYekohubIU
https://www.techwalla.com/articles/what-is-a-display-driver
https://en.wikipedia.org/wiki/Display_driver
https://en.wikipedia.org/wiki/Input_device
https://en.wikipedia.org/wiki/Hardware_interface_design
https://en.wikipedia.org/wiki/Sensor
http://www.tehnium-azi.ro/page/articole_articles/_/articles/notiuni-teoretice-din-electronica/Rezistenta_electrica_rezistorii
https://ro.wikipedia.org/wiki/Rezistor
http://www.creeaza.com/tehnologie/electronica-electricitate/Potentiometrul851.php
http://microcontrollerslab.com/analog-to-digital-adc-converter-working/
http://gts-automatizari.ro/ro/motoare/55-motoare-de-curent-continuu.html
http://www.electricalc.ro/blog/38-masina-de-curent-continuu
https://ro.wikipedia.org/wiki/Motor_electric
http://www.qreferat.com/referate/mecanica/Motorul-pas-cu-pas939.php
https://www.optimusdigital.ro/ro/motoare-motoare-pas-cu-pas/101-driver-uln2003-motor-pas-cu-pas-de-5-v-.html?gclid=CjwKCAjw3cPYBRB7EiwAsrc-uXt1xXrSbzt4DZ2y0OrbYtTtmeriB1suibg7dPHHM9IPcMNPxVupwhoCRVgQAvD_BwE7
https://www.omega.com/prodinfo/stepper_motors.html
http://blog.tehnicale.ro/tag/uln2003an/
https://www.microchip.com/wwwproducts/en/ATmega32A
http://www.electronicsteacher.com/robotics/robotics-tutorial/advanced-robotics/controlling-dc-motors.php

ANEXĂ

adc_driver.c

```
/*
 * adc_driver.c
 *
 * Created: 18-Oct-17 9:51:05 AM
 *  Author: ScorpionIPX
 */

#include <avr/io.h>

void ADC_init(void)
{
	DDRA = 0x00;
	// AREF = AVcc
	ADMUX = (1<<REFS0);

	// ADC Enable and prescaler of 128
	// 16000000/128 = 125000
	ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

// read ADC value
uint16_t ADC_get_value(uint8_t ch)
{
	// select the corresponding channel 0~7
	// ANDing with '7' will always keep the value
	// of 'ch' between 0 and 7
	ch &= 0b00000111;  // AND operation with 7
	ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

	// start single conversion
	// write '1' to ADSC
	ADCSRA |= (1<<ADSC);

	// wait for conversion to complete
	// ADSC becomes '0' again
	// till then, run loop continuously
	while(ADCSRA & (1<<ADSC));

	return (ADC);
}
```

adc_driver.h

```
/*
 * adc_driver.h
 *
 * Created: 18-Oct-17 9:58:32 AM
 *  Author: uidq6025
 */


#ifndef ADC_DRIVER_H_
#define ADC_DRIVER_H_

#include <avr/io.h>

#define ADC_MAX 1023
#define ADC_HALF 512

void ADC_init(void);
uint16_t ADC_get_value(uint8_t ch);

#endif /* ADC_DRIVER_H_ */
```

l293d.c

```
/*
 * l293d.c
 *
 * Created: 22-Apr-18 17:49:49
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <util/delay.h>
#include "l293d.h"

void init_l293d_control(void)
{
	L293D_DDR |= ((1 << L293D_HB2_DIRECTION_LEFT) | (1 << L293D_HB2_DIRECTION_RIGHT) |
(1 << L293D_HB2_ENABLE));
	L293D_PORT &= ~((1 << L293D_HB2_DIRECTION_LEFT) | (1 << L293D_HB2_DIRECTION_RIGHT)
| (1 << L293D_HB2_ENABLE));
}

void l293d_hb2_rotate_left(void)
{
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_LEFT;
	L293D_SET_HB2_ENABLE;
}

void l293d_hb2_rotate_right(void)
{
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_RIGHT;
	L293D_SET_HB2_ENABLE;
}

void l293d_hb2_stop(void)
{
	L293D_CLEAR_HB2_ENABLE;
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
}
```

l293d.h

```
/*
 * l293d.h
 *
 * Created: 22-Apr-18 17:50:02
 *  Author: ScorpionIPX
 */


#ifndef L293D_H_
#define L293D_H_

#include <avr/io.h>

#define L293D_PORT PORTD
#define L293D_DDR DDRD

#define L293D_DEAD_TIME_MS 1  /* delay time to wait before changing H bridge current
direction */

#define L293D_HB2_DIRECTION_LEFT PORTD3
#define L293D_HB2_DIRECTION_RIGHT PORTD4
#define L293D_HB2_ENABLE PORTD5

#define L293D_SET_HB2_DIRECTION_LEFT (L293D_PORT |= (1 << L293D_HB2_DIRECTION_LEFT))
#define L293D_SET_HB2_DIRECTION_RIGHT (L293D_PORT |= (1 << L293D_HB2_DIRECTION_RIGHT))
#define L293D_SET_HB2_ENABLE (L293D_PORT |= (1 << L293D_HB2_ENABLE))

#define L293D_CLEAR_HB2_DIRECTION_LEFT (L293D_PORT &= ~(1 << L293D_HB2_DIRECTION_LEFT))
#define L293D_CLEAR_HB2_DIRECTION_RIGHT (L293D_PORT &= ~(1 << L293D_HB2_DIRECTION_RIGHT))
#define L293D_CLEAR_HB2_ENABLE (L293D_PORT &= ~(1 << L293D_HB2_ENABLE))


void init_l293d_control(void);
void l293d_hb2_rotate_left(void);
void l293d_hb2_rotate_right(void);
void l293d_hb2_stop(void);

#endif /* L293D_H_ */
```

light.c

```
/*
 * light.c
 *
 * Created: 28-Oct-17 7:02:05 PM
 *  Author: ScorpionIPX
 */

#include <avr/io.h>
#include "light.h"
#include "adc_driver.h"

#define FILTLER_RANK 15

int get_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = ADC_get_value(sensor);
	adc_value = percentage_value(adc_value);
	return adc_value;
}

int get_filtered_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = 0;
	for(char i = 0; i < FILTLER_RANK; i++)
	{
		adc_value += ADC_get_value(sensor);
	}
	adc_value /= FILTLER_RANK;
	adc_value = percentage_value(adc_value);
	return adc_value;
}

int percentage_value(int raw_value)
{
	raw_value = raw_value*((long)100)/1023;
	raw_value = 100 - raw_value;
	return raw_value;
}
```

light.h

```
/*
 * light.h
 *
 * Created: 28-Oct-17 7:02:15 PM
 *  Author: ScorpionIPX
 */


#ifndef LIGHT_H_
#define LIGHT_H_

#define LS_UP_LEFT    0
#define LS_UP_RIGHT   1
#define LS_DOWN_LEFT  2
#define LS_DOWN_RIGHT 3

#define LS_UP_LEFT_RAW_OFFSET 0
#define LS_UP_RIGHT_RAW_OFFSET   0
#define LS_DOWN_LEFT_RAW_OFFSET  -5
#define LS_DOWN_RIGHT_RAW_OFFSET 0

static const int LS_RAW_OFFSETS[4] = {LS_UP_LEFT_RAW_OFFSET, LS_UP_RIGHT_RAW_OFFSET,
LS_DOWN_LEFT_RAW_OFFSET, LS_DOWN_RIGHT_RAW_OFFSET};

int get_light_intensity(uint8_t sensor);
int get_filtered_light_intensity(uint8_t sensor);
int percentage_value(int raw_value);

#endif /* LIGHT_H_ */
```

main.c

```
#include "global.h"
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "adc_driver.h"
#include "hx1230.h"
#include "hx_8x6_characters.h"
#include "graphics.h"
#include "light.h"
#include "pwm_driver.h"
#include "sg90_driver.h"
#include "tracking.h"
#include "user_interface.h"
#include "state_handler.h"
#include "joystick_driver.h"
#include "monitoring.h"
#include "unipolar_driver.h"
#include "l293d.h"

void uC_init(void);

int main(void)
{
	STATE = STATE_INIT;
	OLD_STATE = STATE_INIT;
	uC_init();

	STATE = STATE_IDLE;

	while (1)
	{
		if(STATE_CHANGED)
		{
			OLD_STATE = STATE; // update state
			go_to_state(STATE);
			_delay_ms(250);
			sei(); // enable interrupts
		}

		switch(OLD_STATE)
		{
			case STATE_TRACKING:
			{
				track();
				break;
			}
			case STATE_MANUAL:
			{
				manual_control();
				break;
			}
			case STATE_MONITORING:
			{
```

```
                        monitor();
                        break;
                }
                default:
                {
                        break;
                }
        }
    }
}

void uC_init(void)
{
    // Wait for system to get fully powered up
    _delay_ms(100);

    // initialize required modules
    ADC_init();
    _delay_ms(50);

    init_user_interface();
    _delay_ms(50);

    init_unipolar_control();
    _delay_ms(50);

    init_l293d_control();
    _delay_ms(50);

    init_hx1230_control();
    _delay_ms(50);
    hx_fill_screen();
    _delay_ms(500);
    hx_clear_screen();
    _delay_ms(50);

    display_title();
    display_idle_state_message();

    sei(); // enable global interrupts
}
```

stand_control.c

```
/*
 * stand_control.c
 *
 * Created: 29-Apr-18 21:55:45
 *  Author: ScorpionIPX
 */

#include "global.h"
#include "l293d.h"
#include "unipolar_driver.h"

#define ROTATE_LEFT_ALLOWED 1
#define ROTATE_RIGHT_ALLOWED 1
#define INCLINE_UP_ALLOWED 1
#define INCLINE_DOWN_ALLOWED 1

void stand_rotate_left(void)
{
	if(ROTATE_LEFT_ALLOWED)
	{
		l293d_hb2_rotate_left();
	}
}

void stand_rotate_right(void)
{
	if(ROTATE_RIGHT_ALLOWED)
	{
		l293d_hb2_rotate_right();
	}
}

void stand_stop_rotation(void)
{
	l293d_hb2_stop();
}

void stand_incline_down(void)
{
	if(INCLINE_DOWN_ALLOWED)
	{
		unipolar_01_step_backward(UNIPOLLAR_01_CURRENT_STEP);
	}
}

void stand_incline_up(void)
{
	if(INCLINE_UP_ALLOWED)
	{
		unipolar_01_step_forward(UNIPOLLAR_01_CURRENT_STEP);
	}
}

void stand_stop_incline(void)
{
```

```
        unipolar_01_clear_steps();
}
```

stand_control.h

```
/*
 * stand_control.h
 *
 * Created: 29-Apr-18 21:56:02
 *  Author: ScorpionIPX
 */


#ifndef STAND_CONTROL_H_
#define STAND_CONTROL_H_

void stand_rotate_left(void);
void stand_rotate_right(void);
void stand_stop_rotation(void);
void stand_incline_down(void);
void stand_incline_up(void);
void stand_stop_incline(void);

#endif /* STAND_CONTROL_H_ */
```

tracking.c

```
/*
 * tracking.c
 *
 * Created: 29-Oct-17 5:24:58 PM
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <avr/io.h>
#include <stdlib.h>
#include <util/delay.h>
#include "tracking.h"
#include "light.h"
#include "hx1230.h"
#include "graphics.h"
#include "stand_control.h"

int light_up_left;
int light_up_right;
int light_down_left;
int light_down_right;

int up_intensity_average;
int down_intensity_average;
int left_intensity_average;
int right_intensity_average;

int up_down_movement_gradient_request;
int left_right_movement_gradient_request;

void track(void)
{
	light_up_left = get_filtered_light_intensity(LS_UP_LEFT);
	light_up_right = get_filtered_light_intensity(LS_UP_RIGHT);
	light_down_left = get_filtered_light_intensity(LS_DOWN_LEFT);
	light_down_right = get_filtered_light_intensity(LS_DOWN_RIGHT);

	display_light_sensor_data(LS_UP_LEFT, light_up_left);
	display_light_sensor_data(LS_UP_RIGHT, light_up_right);
	display_light_sensor_data(LS_DOWN_LEFT, light_down_left);
	display_light_sensor_data(LS_DOWN_RIGHT, light_down_right);

	up_intensity_average = light_up_left + light_up_right;
	up_intensity_average >>= 1;

	down_intensity_average = light_down_left + light_down_right;
	down_intensity_average >>= 1;

	left_intensity_average = light_up_left + light_down_left;
	left_intensity_average >>= 1;

	right_intensity_average = light_up_right + light_down_right;
	right_intensity_average >>= 1;

	hx_set_coordinates(42, 2);
```

```
		hx_write_char('0' + (up_intensity_average / 10) % 10);
		hx_write_char('0' + up_intensity_average % 10);

		hx_set_coordinates(42, 6);
		hx_write_char('0' + (down_intensity_average / 10) % 10);
		hx_write_char('0' + down_intensity_average % 10);

		hx_set_coordinates(6, 4);
		hx_write_char('0' + (left_intensity_average / 10) % 10);
		hx_write_char('0' + left_intensity_average % 10);

		hx_set_coordinates(78, 4);
		hx_write_char('0' + (right_intensity_average / 10) % 10);
		hx_write_char('0' + right_intensity_average % 10);

		up_down_movement_gradient_request = up_intensity_average - down_intensity_average;
		left_right_movement_gradient_request = left_intensity_average -
right_intensity_average;

		if(abs(up_down_movement_gradient_request) > INCLINE_TRACKING_TOLERANCE)
		{
			if(up_down_movement_gradient_request > 0)
			{
				stand_incline_up();
			}
			else
			{
				stand_incline_down();
			}
		}
		else
		{
			stand_stop_incline();
		}

		if(abs(left_right_movement_gradient_request) > ROTATE_TRACKING_TOLERANCE)
		{
			if(left_right_movement_gradient_request > 0)
			{
				stand_rotate_right();
			}
			else
			{
				stand_rotate_left();
			}
		}
		else
		{
			stand_stop_rotation();
		}
		_delay_ms(40);
}
```

unipolar_driver.c

```
/*
 * unipolar_driver.c
 *
 * Created: 17-Apr-18 18:55:52
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <avr/io.h>
#include <util/delay.h>
#include "unipolar_driver.h"

void init_unipolar_control(void)
{
	UNIPOLAR_01_DDR |= ((1 << UNIPOLAR_01_STEP_1) | (1 << UNIPOLAR_01_STEP_2) | (1 << 
UNIPOLAR_01_STEP_3) | (1 << UNIPOLAR_01_STEP_4));
	UNIPOLAR_01_CLEAR_STEP_1;
	UNIPOLAR_01_CLEAR_STEP_2;
	UNIPOLAR_01_CLEAR_STEP_3;
	UNIPOLAR_01_CLEAR_STEP_4;
	UNIPOLLAR_01_CURRENT_STEP = 4;
}

void unipolar_01_step_forward(unsigned char current_step)
{
	switch(current_step)
	{
		case 1:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_SET_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 2;
			break;
		}
		case 2:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_SET_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 3;
			break;
		}
		case 3:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_SET_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 4;
			break;
		}
		case 4:
```

```
		{
			UNIPOLAR_01_SET_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 1;
			break;
		}
	}
	_delay_ms(3);
}

void unipolar_01_step_backward(unsigned char current_step)
{
	switch(current_step)
	{
		case 1:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_SET_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 4;
			break;
		}
		case 2:
		{
			UNIPOLAR_01_SET_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 1;
			break;
		}
		case 3:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_SET_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 2;
			break;
		}
		case 4:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_SET_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 3;
			break;
		}
	}
	_delay_ms(3);
}

void unipolar_01_clear_steps(void)
{
	UNIPOLAR_01_CLEAR_STEP_1;
```

```
        UNIPOLAR_01_CLEAR_STEP_2;
        UNIPOLAR_01_CLEAR_STEP_3;
        UNIPOLAR_01_CLEAR_STEP_4;
}
```

unipolar_driver.h

```
/*
 * unipolar_driver.h
 *
 * Created: 17-Apr-18 18:56:06
 *  Author: ScorpionIPX
 */


#ifndef UNIPOLAR_DRIVER_H_
#define UNIPOLAR_DRIVER_H_

void init_unipolar_control(void);
void unipolar_01_step_forward(unsigned char current_step);
void unipolar_01_step_backward(unsigned char current_step);
void unipolar_01_clear_steps(void);

#define UNIPOLAR_01_DDR DDRB
#define UNIPOLAR_01_PORT PORTB

#define UNIPOLAR_01_STEP_1 PORTB0
#define UNIPOLAR_01_STEP_2 PORTB1
#define UNIPOLAR_01_STEP_3 PORTB2
#define UNIPOLAR_01_STEP_4 PORTB3

#define UNIPOLAR_01_SET_STEP_1 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_1)
#define UNIPOLAR_01_SET_STEP_2 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_2)
#define UNIPOLAR_01_SET_STEP_3 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_3)
#define UNIPOLAR_01_SET_STEP_4 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_4)

#define UNIPOLAR_01_CLEAR_STEP_1 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_1))
#define UNIPOLAR_01_CLEAR_STEP_2 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_2))
#define UNIPOLAR_01_CLEAR_STEP_3 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_3))
#define UNIPOLAR_01_CLEAR_STEP_4 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_4))

#endif /* UNIPOLAR_DRIVER_H_ */
```