**UNIVERSITATEA „PETRU MAIOR”**

**TÎRGU - MUREŞ**

**FACULTATEA DE ŞTIINŢE ŞI LITERE**

**Programul de studiu : INFORMATICĂ**

**LUCRARE DE LICENŢĂ**

**INTELIGENȚĂ ARTIFICIALĂ**

**ÎN ROBOTICĂ**

Coordonator ştiinţific:

Prof.univ.dr.Călin Enăchescu

**Absolvent:**

**Sînmărtean Alin-Mircea**

-2017-

|  |
| --- |
| **UNIVERSITATEA “PETRU MAIOR” DIN TG. MUREȘ LUCRARE DE LICENŢĂ****FACULTATEA DE ŞTIINŢE ŞI LITERE Candidat ul (a): Sînmărtean Alin-Mircea** **Programul de studiu: Informatică Anul absolvirii : 2017** |
| **Conducătorul ştiinţific : Viza facultăţii**  **Prof. univ. dr. Călin Enăchescu** |
| **a) Tema lucrării de licenţă/disertație :**  **Inteligență artifială în robotică** |
| **b) Problemele principale tratate :**  **Introducere în inteligența artificială**  **Introducere în Arduino**  **Deplasarea unui robot cu ajutorul rețelelor neuronale**  **c) Bibliografia recomandată :**   * **Enăchescu Călin , Calculul Neuronal ; Editura "Casa Cărţii de Ştiinţă", Cluj-Napoca 2008** * **Neural Networks - A Systematic Introduction – Raul Rojas** * **h**[**ttps://en.wikipedia.org/wiki/Artificial\_neural\_network**](https://en.wikipedia.org/wiki/Artificial_neural_network) * **Introducere in Arduino – Optimus Digital** * **http://www.descopera.org/inteligenta-artificiala/** |
| **d) Termene obligatorii de consultaţii : săptămânal** |
| **e) Locul şi durata practicii :Laboaratoarele UPM** |
| **Primit tema la data de : 03-11-2016** |
| **Termen de predare :23-06-2017** |
| **Semnătura directorului de departament Semnătura conducătorului** |
| **Semnătura candidatului** |

**Cuprins**

[INTRODUCERE 5](#_Toc485568275)

[CAPITOLUL 1 INTRODUCERE ÎN INTELIGENȚA ARTIFICIALĂ 7](#_Toc485568276)

[**1.1. SCURT ISTORIC 7**](#_Toc485568277)

[**1.2. NEURONUL BIOLOGIC 8**](#_Toc485568278)

[**1.3. NEURONUL ARTIFICIAL 10**](#_Toc485568279)

**1.4.** [**REŢELE NEURONALE ARTIFICIALE 11**](#_Toc485568280)

**1.5.** [**PARADIGME DE ÎNVĂȚARE 12**](#_Toc485568281)

**1.6.** [**REȚELE NEURONALE VERSUS PROCESARE CONVENȚIONALĂ 13**](#_Toc485568282)

**1.7.** [**METODA DE ÎNVĂȚARE BACKPROPAGATION 14**](#_Toc485568283)

[CAPITOLUL 2 INTRODUCERE ÎN ARDUINO 18](#_Toc485568284)

[**2.1. INTRODUCERE ÎN PLATFORMA ARDUINO 18**](#_Toc485568285)

[**2.2. SOFTWARE 18**](#_Toc485568286)

[**2.3. HARDWARE 20**](#_Toc485568287)

[**2.4. ARDUINO UNO 21**](#_Toc485568288)

[CAPITOLUL 3 APLICAȚIE PRACTICĂ 23](#_Toc485568289)

[**3.1. PARTEA FIZICĂ 23**](#_Toc485568290)

[**3.2. PARTEA SOFTWARE 28**](#_Toc485568291)

[CAPITOLUL 4 CONCLUZII 36](#_Toc485568292)

[CAPITOLUL 5 BIBLIOGRAFIE 37](#_Toc485568293)

**INTRODUCERE**

Această lucrare îşi propune să prezinte principalele concepte ale calculului neuronal în robotică cu scopul de a crea un robot autonom capabil să se deplaseze pe un traseu aleator.

Am decis să încep cercetarea în acest domeniu deoarece, în ultimii ani, s-a pus un accent tot mai mare pe dezvoltarea vehiculelor autonome, care pot circula fără a fi nevoie de o acţiune a omului şi care ar prezenta mult mai multe avantaje decât condusul clasic întrucât se consideră că ar elimina aproximativ 90% din numărul total de accidente, ar minimiza congestia din trafic precum şi faptul că ar reduce personalul din mijloacele de transport în comun.

Consider că trăim vremuri în care deplasarea cu un automobil este necesară pentru multe persoane şi erorile provocate de şoferi din cauza oboselii sau a neatenţiei se plătesc cu preţuri mari,acestea constânt în accidente cu rănirea sau chiar decesul persoanelor, motiv pentru care cred că o masină, având o formă de „inteligenţă” ar putea avea ca efect reducerea drastică a numărului de accidente şi astfel un pas înainte în societatea noastră.

Lucrarea de faţă prezintă o variantă mult simplificată a acestui proiect grandios însă totodată oferă o direcţie de început în acest domeniu, deoarece este vorba despre deplasarea unui vehicul în cadrul unui traseu delimitat.Astfel, conținutul lucrării este alcătuit din următoarele capitole:

Capitolul 1, Introducere în Inteligenta artificială prezintă,pe lângă o scurtă introducere in acest domeniu, concepte fundamentale legate de neuronul biologic,neuronul artificial, rețele neuronale artificiale,diferența dintre procesarea clasică și cea dată de rețelele neuronale, precum și de algoritmul backpropagation, fiind cel mai cunoscut algoritm folosit în antrenarea perceptronului multistrat.

Capitolul 2,Introducere în Arduino prezintă elemente introductive despre această platformă alături de elementele hardware și software ale acesteia.

Capitolul 3,Aplicație practică prezintă concret în ce a constat aplicația mea realizată cu toate aspectele de implementare, atât cele fizice cât si cele de programare.

Capitolul 4 va prezenta concluziile la care am ajuns în urma cercetării și realizării practice a acestei aplicații.

Capitolul 5 va prezenta bibliografia care m-a ajutat să descopăr informații noi în acest domeniu,pentru a-mi putea duce proiectul la bun sfarsit.

# **CAPITOLUL 1: INTRODUCERE ÎN INTELIGENȚA ARTIFICIALĂ**

## **1.1. SCURT ISTORIC**

Istoria Inteligenței artificiale începe abia în a 2-a jumătate a secolului XX și a început din dorința oamenilor de a minimiza eforturile intelectuale ale acestora și de a maximiza totodată performanțele în activitățile cotidiene, astfel punându-se problema schimbării poziției omului din efectuant în supervizor.Momentul de naștere al inteligenței artificiale este legat de anul 1956 la un workshop ținut în campusul universității Dartmouth unde pe lângă faptul că a fost prezentat un program de demonstrare a logicii propozitiilor, persoanele care au participat au ajuns să fie lideri în acest domeniu, aici amintim numele lui John McCharty, Marvin Minsky, Allen Newell si Herbert Simon. Aceștia au fondat centre mari de cercetare în cadrul M.I.T, C.M.U. și Standford și au primit milioane de dolari din partea guvernelor pentru proiectele lor. Astfel începând cu anul 1960 au apărut primele proiecte de inteligență artificială,fondatorii find foarte optimiști în legătură cu dezvoltarea acestui domeniu, fiind de părere că în maxim 50 de ani calculatoarele vor fi mai inteligente decât oamenii.

Însă, în cele din urmă a devenit evident faptul că aceștia supraestimaseră dificultățile întâmpinate.Fiind supuși unor critici ale lui Sir James Lighthil și fiind presați de Congresul American care dorea ca fondurile să fie investite în proiecte productive, cele două guverne, cel american și cel britanic, vor renunța la proiectele din domeniul inteligenței artificiale.

Sistemele expert au ajutat domeniul să revină pe piață, iar până în anul 1985 piața A.I avea peste 1 miliard de dolari și investiții mari din partea guvernelor însă câțiva ani mai târziu datorită prăbușirii pieței, domeniul și-a pierdut majoritatea investitorilor, astfel progresele au redevenit extrem de lente și inconsistente.Această perioadă a ținut până în anii ’90 moment în care inteligența artificială a început să fie folosită în majoritatea domeniilor datorită puterii de calcul a computerelor.În momentul actual acest domeniu este într-o dezvoltare continuă întrucât cercetările sunt sponsorizate iar puterea de calcul este suficient de mare pentru ca noile teorii să își poată manifesta practicabilitatea.

## 

## **1.2. NEURONUL BIOLOGIC**

Neuronul este unitatea principală a sistemului nervos, fiind o celula cu excitabilitate electrică,însemnând că poate procesa şi transmite informaţii cu ajutorul semnalelor electro-chimice.Omul are în jur de 1011 neuroni în corp, dispuşi în mare parte în creier dar şi în coloana vertebrală, fiecare neuron având aprox. 104 conexiuni cu alţi neuroni.

Cele 3 părţi fundamentale care stau la baza alcătuirii neuronului sunt date de:

-Arbore dentritic: receptează semnalele de intrare ale altor neuroni

-Corpul celular(soma): poate transforma, în anumite condiţii, semnalele de intrare în semnale de ieşire

-Axonul: transmite semnalul de ieşire către alţi neuroni

Sunt mai multe tipuri de neuroni în corpul nostru, dintre care amintim neuronii senzoriali care răspund unor stimuli precum cei vizuali,auditivi, olfactivi, neuronii motorii care se ocupă de contracţia musculară, precum și interneuronii, care conectează mai mulţi neuroni din aceeaşi zonă a creierului.

Comunicația dintre neuroni se realizează cu ajutorul unor legături numite sinapse, care nu reprezintă altceva decât puncte de contact dintre axonul unui neuron și dentritele neuronul cu care se realizează legătura.

„La sinapsă o substanţă chimică, numită neuro-transmiţător, este stocată într-un număr mare de vezicule. Potenţialul de acţiune eliberează conţinutul anumitor vezicule către cleştele sinaptic. Moleculele neuro-transmiţătorului ajung la membrana neuronului post-sinaptic, unde recepţia acestor molecule induc un potenţial de acţiune post-sinaptic (PSP). PSP-urile generate în diferite puncte ale arborelui dendritic difuzează prin atenuare către soma, unde ele sunt integrate. Dacă suma totală a PSP-urilor integrate în limita unui scurt interval de timp depăşeşte un anumit prag (threshold) de aproximativ câteva zecimi de minivolţi, numit nivel de activare, neuronul va deveni activ, generând un potenţial de acţiune de-a lungul axonului.”[1]

„Intervalul de timp dintre momentul emisiei unui semnal la soma neuronului pre-sinaptic şi momentul emisiei unui semnal indus de către neuronul post-sinaptic, este de aproximativ 1-2 msec. De aici rezultă că un neuron poate avea o emisie maximală de circa 500-1000 semnale pe secundă, care într-o reţea neuronală este redusă de circa 3-5 ori.(.Din aceste considerente de dinamică a activităţii neuronale, se poate observa faptul că neuronul biologic este un dispozitiv biologic lent în comparaţie cu dispozitivele electronice construite de om -acestea pot fi chiar de sute de mii de ori mai rapide decât un neuron biologic. Cu toate acestea, orice sistem de calcul bazat pe dispozitive electronice are performanţe inferioare creierului uman constituit din neuroni. Concluzia evidentă este că puterea de calcul al creierului uman nu este datorată vitezei de procesare a neuronilor constitutivi, ci largii interconectări a unor dispozitive biologice lente -neuronii, care efectuează operaţii simple: integrarea semnalelor sosite de-a lungul arborelui dendritic şi emisia unui semnal de-a lungul axonului, dacă semnalul de intrare integrat depăşeşte nivelul de activare.”[1]

În figura. 1.1 se poate observa neuronul biologic alături de componentele sale

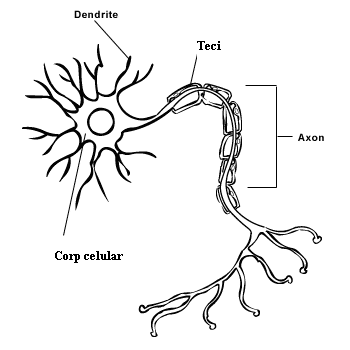


Figura 1.1 Neuronul biologic

## 

## **1.3. NEURONUL ARTIFICIAL**

Neuronul artificial este unitatea de bază a unei reţele neuronale, fiind un model matematic al neuronului biologic.Acesta primeşte unul sau mai multe input-uri, care corespund dentritelor.”Celei de-a i-a cale de intrare în neuronul al j-lea îi corespunde o valoare numerică reală xi, echivalentul semnalului electric din modelul biologic al neuronului. Fiecare mărime de intrare xi este ponderată valoarea numerică reală wji, echivalentul tăriei sinaptice din modelul biologic al neuronului. Produsul xi \* wji reprezintă al i-lea semnal de intrare dentritic în al j-lea neuron artificial.Acestor valori le e aplicata o functie de sumare( ) iar din rezultatul lor se produce un output sau o functie de activare,fiind reprezentat de axonul neuronului biologic.”[1].Cele mai utilizate functii de activare sunt:

-functia liniară f : R → R, f(x) = x

-funcţia treaptă(Heavyside): f : R → {0,1}, f(x)

-funcţia rampă: f : R → [-1, 1], f(x)

-funcţia sigmoidală f : R → (0, 1), f(x) =

-funcţia tangentă hiperbolică: f : R → (-1, 1), f(x) = tanh(x) =

-funcţia signum: f : R →{-1, 1}, f(x) = sgn(x) = [1]

Alegerea funcției de activare depinde de modelul de rețea neuronală ales precum și de complexitatea problemei, pentru problemele simple fiind suficientă funcția liniară.Însă pentru problemele complexe, dorim să introducem conceptul de non-linearitate în rețeaua noastră neuronală, aici fiind mai adecvată o funcție derivabilă, precum funcția sigmoidală.

Din punct de vedere vizual, structura neuronului artificial poate fi observată în figura 1.2, unde este prezent un neuron artificial ce primește input-uri de la alți neuroni, fiecare input având o tărie sinaptică, produsul dintre cei doi fiind constituit dintr-o intrare în neuron.Suma intrărilor este supusă unei funcții de activare și apoi output-ul este trimis mai departe.

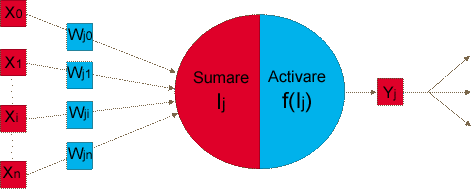


Figura 1.2 Structura unui neuron artificial

„Termenul x0 se numeşte bias,avand o valoare constanta x0 = -1 sau x0 = +1,acest termen e folosit pentru a creşte flexibilitatea reţelei neuronale întrucât permite includerea explicita sau implicita a pragului de activare a neuronului artificial.”[1]

## **1.4. REŢELE NEURONALE ARTIFICIALE**

Mai mulţi neuroni artificiali conectaţi între ei constituie o reţea neuronală. Scopul principal al unei reţele neuronale este de a aduce rezolvări unor probleme complexe, precum recunoașterea șabloanelor, într-un mod asemănător creierului uman.Odată ce reţeaua neuronală a fost antrenată cu mai multe date,ea va fi capabilă să „inveţe” tiparul acestor date şi să prezică un răspuns aferent.

„În mod tipic, neuronii sunt organizaţi pe straturi, iar semnalul va merge de la primul strat(input-ul) către ultimul strat(output-ul)” [2].O reţea neuronală va avea cel puţin două straturi : stratul de intrare,care va avea ca şi funcţie de activare întotdeauna funcţia liniară întrucât preia datele de intrare şi le propagă mai departe în reţea și stratul de ieşire, care va genera un output pe baza informaţiilor primite de la straturile anterioare.

În cazul în care există mai multe straturi,acestea se vor numi straturi ascunse.

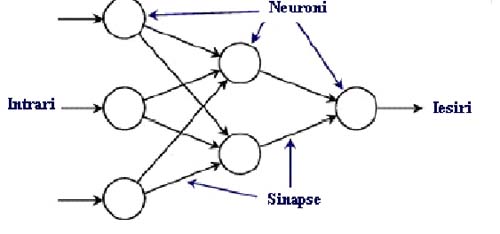


Figura 1.3 Structura unei retele de neuroni artificiali

## **1.5. PARADIGME DE ÎNVĂȚARE**

a) Învăţarea supervizată

Are la bază prezentarea rețelei neuronale datele sub forma perechilor input(x1)-output(y1) de unde capacitatea rețelei ca pornind de la un alt input(x2) să prezică output-ul dorit(y2).Pe baza răspunsului obținut și a răspunsului dorit se obține o eroare ce e propagată înapoi cu scopul diminuării acesteia. Denumirea de învățare supervizată este dată de faptul că perechile input-output inițiale sunt furnizate de un profesor(anternor-supervizor) care știe rezultatele corecte și ajută rețeaua neuronală în corectarea erorii sale.

b)Invatarea nesupervizata

Spre deosebire de învățarea supervizată, în cadrul învăţării nesupervizate nu există profesor (antrenor). Reţeaua neuronală trebuie să fie în stare să “descopere” singură modele, trăsături, corelaţii sau categorii în mulţimea datelor de intrare şi să le codifice sub forma unor date de ieşire . Învăţarea nesupervizată poate fi utilizată doar atunci când în mulţimea datelor de intrare există redundanţă. Fără redundanţă este imposibil de a descoperi vreun model (patern) sau trăsătură în mulţimea datelor de intrare.

Principalele probleme ce pot fi abordate folosind această paradigmă sunt date de probleme de estimări,asocieri, clustering si probabilitati.

c)Învățarea întărită

Este inspirată din psihologia behavioristă şi diferă de paradigmele de invăţare supervizate, întrucât perechile input-output nu sunt prezente. „Mai mult, aceasta se focusează pe învăţarea on-line, ceea ce presupune găsirea unei balanţe între explorarea(unui teritoriu „necunoscut”) şi exploatarea(cunoştinţelor curente)” [3]. Prin învățarea întărită, neuronilor nu li se spune ce anume mai exact trebuie să învețe ci începând sa învețe arbitrar li se va oferi o „recompensă” pentru acțiunile bune, astfel, în mod iterativ aceștia vor încerca sa învețe doar acțiunile care le oferă recompensele cele mai mari.Această paradigmă este studiată în mai multe discipline precum teoria jocurilor, teoria controlului, sisteme multi-agent, algoritmi genetici precum şi în statistică.

## **1.6. REȚELE NEURONALE VERSUS PROCESARE CONVENȚIONALĂ**

Reţelele neuronale diferă de procesarea convenţională întrucât cea din urmă foloseşte o abordare algoritmică, adică urmează un set de instrucţiuni şi paşi pentru a rezolva o anumită problemă.O problemă poate fi rezolvată doar dacă toţi paşii ei de rezolvare sunt cunoscuţi dinainte,ceea ce restricţionează problema la capacitatea noastră de a o înţelege şi de a-i găsi o rezolvare.Cu toate acestea, multe probleme necesită o abordare diferită, întrucât doar o parte din date sunt cunoscute şi cu toate acestea, cu ajutorul capacităţii de generalizare şi predicţie, e nevoie să se ajungă la un rezultat.Principalele avantaje ale calculului neuronal în detrimentul calculului algoritmic sunt date de faptul că cel dintâi are un număr mare de elemente de procesare(neuroni), fiecare efectuând un set mic de operaţii simple şi având numeroase conexiuni cu diverse tării sinaptice.De asemenea un alt exemplu elocvent e dat de faptul că reţelele neuronale învaţă din exemple, acestea nefiind programate să facă un anumit task.

Reţele neuronale şi procesarea convenţională nu sunt, sau cel puţin nu ar trebui să fie într-o competiţie ci se completează reciproc.Unele task-uri sunt mai adecvate pentru folosirea abordării algorimice, iar altele pentru folosirea abordarii retelelor neuronale.”Mai mult chiar, un numar mare de task-uri foloseşte o combinaţie între cele două abordări(spre exemplu o reţea neuronală poate fi supervizată de către un calculator conventional) cu scopul de a obţine o eficienţă maximă.” [4]

## **1.7. METODA DE ÎNVĂȚARE BACKPROPAGATION**

Algoritmul de „propagare înapoi a erorilor” sau pe scurt backpropagation este, la ora actuală, cel mai cunoscut algoritm de a antrena o rețea neuronală, bazându-se pe metoda gradientului descendent.

Acest algoritm a apărut în anii ’70 dar importanța acestuia nu a fost apreciată întrucât presupunea un volum mult prea mare de calcule la acea vreme, decât odată cu lucrarea scrisă de David E. Rumelhart, Geoffrey E. Hinton si Ronald J. Williams, intitulată „Learning representations by back-propagating errors” în anul 1986, lucrare ce prezintă avantajele utilizării algoritmului backpropagation în raport cu vechile abordări pentru învățarea rețelelor neuronale.

Fiind o generalizare a algoritmului delta, backpropagation face parte din categoria învățării supervizate, întrucât este nevoie de un set de perechi input-output pe baza căruia se va calcula funcția eroare.

Scopul algoritmului este minimizarea functiei de cost, acest lucru făcându-se în 2 faze: propagarea și modificarea tăriilor sinaptice.Ideea de bază constă în compararea output-ului obținut de o rețea neuronală cu output-ul dorit de utilizator, comparație care generează o anumită eroare,aceasta eroare fiind caracteristică fiecarui neuron din stratul de ieșire este propagată înapoi spre straturile ascunse iar tăriile sinaptice se modifică aferent, urmând ca dupa multe iterații, de regulă mii, tăriile sinaptice să fie ajustate în așa fel încât să prezinte nu doar rezultatul corect pentru input-urile date de utilizator, ci și pentru alte date necunoscute.

Principalele probleme pe care dezvoltatorul rețelei neuronale trebuie sa le rezolve , întâlnite în cadrul acestui algoritm sunt date de găsirea funcției de activare, normalizarea datelor și găsirea punctului de minim global.”Pentru găsirea funcției de activare, din moment ce acest algoritm presupune calculul gradientului funcții de cost la fiecare pas, trebuie să garantăm continuitatea si diferențiabilitatea acesteia”[8].Astfel principala funcție de activare folosită este funcția sigmoidală,fiind o funcție reala: σ : R → (0, 1) definită de expresia:

σ (x) =

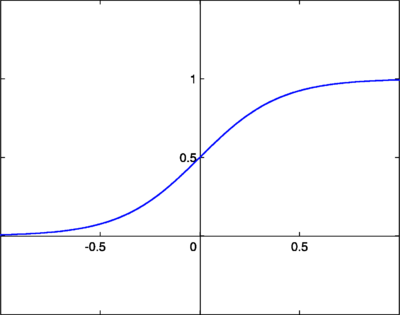
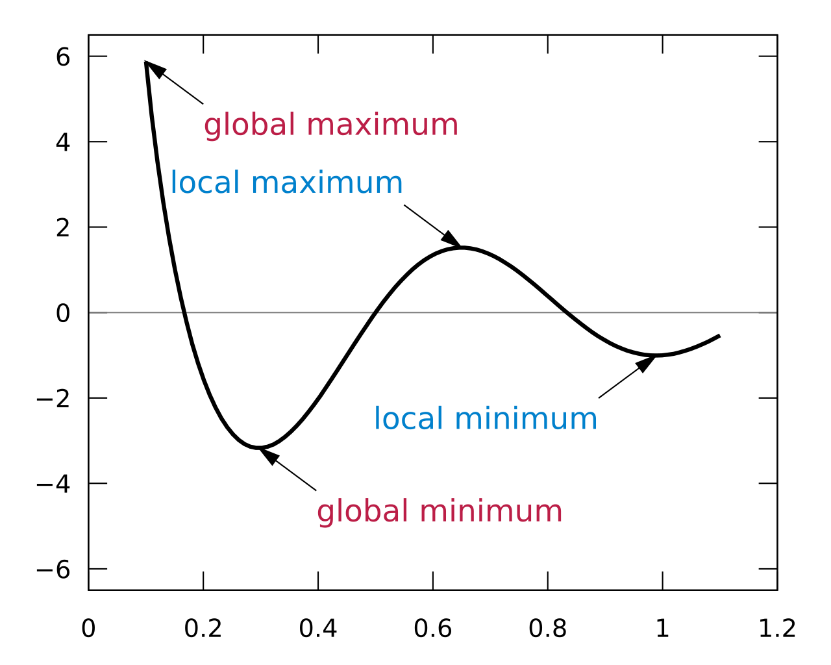


Figura 1.4 Graficul functiei sigmoidale

„Normalizarea reprezintă procesul de scalare a vectorului de ieșire al unui strat al rețelei neuronale astfel încât suma totală a componentelor acestui vector să rămână constantă.”[1].Astfel datele trebuie scalate astfel încât acestea să aibă o dimensiune comparabilă pentru ca rețeaua neuronală să fie capabilă să genereze răspunsurile corecte.În cazul folosirii funcției sigmoidale, datele normalizate trebuie să fie în intervalul (0, 1).

Găsirea punctului de minim global presupune învățarea cu o anumită rată, astfel încât se va ajunge la minimizarea erorii.Principalul obstacol constă în însăși rata de învățare care, dacă este prea mare riscăm să trecem peste punctul de minim, iar la polul opus, dacă este prea mică riscăm să nu mai ajungem la punctul de minim global.



.

Figura 1.5 Graficul punctelor de minim

Algoritmul Backpropagation are mai multe etape de parcurgere, astfel:

Pasul 1: Alegem o rată de învățare,setăm eroarea egală cu 0 și inițializăm tăriile sinaptice cu valori aleatoare cuprinse între -1 și 1 astfel:

∈ (-1, 1)

Pasul 2: Aplicăm pe stratul de intrare vectorul de intrare xµ și introducem în rețeaua neuronală ieșirea corectă zµ.Astfel stratul de intrare y0 va fi de forma:

= , j = 1,...,n

Pasul 3: Se calculează ieșirea rețelei neuronale după următoarele formule:

= fact() , unde fact este funcția de activare aplicată funcției de integrare =

Pasul 4: Se calculează eroarea pătratică astfel:

Eµ = 1/2 \* , unde zi = vectorul dorit, iar yi = vectorul obtinut

Pasul 5: Propagam eroarea inapoi pe straturile anterioare si actualizam tariile sinaptice:

Δ=ηyk-1 , unde η este rata de învățare

=  + Δ

Pasul 6: Daca µ < Emax(unde Emax reprezintă numărul maxim de epoci folosite în învățare) sau E >ε (ε este valoarea prestabilită a erorii admise) atunci GOTO Pasul 2, altfel STOP.[1]

# 

# **CAPITOLUL 2 INTRODUCERE ÎN ARDUINO**

## 

## **2.1. INTRODUCERE ÎN PLATFORMA ARDUINO**

„Arduino este cea mai simplă soluţie pentru a dezvolta aplicaţii electronice, fiind o platformă uşor de utilizat: o plăcuţă uşor de programat şi de integrat în aplicaţii, open-source, foarte bine documentată şi cu o comunitate bogată”[5].Scopul iniţial al platformei Arduino a fost de a introduce în electronică persoanele care au o experienţă puţină sau chiar deloc în domeniul ingineriei sau al programării, însă cu timpul s-a diferenţiat de contra-candidaţii săi prin robusteţe şi simplitate întrucât nu este nevoie de încă o parte hardware pentru a încărca codul sursă pe placă, ci doar folosind un cablu USB.

„De remarcat faptul că în ultimii ani, în jurul acestei platforme s-a dezvoltat o sumedenie de dispozitive, module şi senzori cu care utilizatorul poate lua foarte multe informaţii din mediul din jur.Aici putem menţiona drivere pentru motoare pas cu pas, fiind motoare ce se pot roti sub unghiuri discrete bine definite(paşi)”[5],module bluetooth ce permit comunicarea cu diverse dispozitive care au această tehnologie, shield-uri Ethernet pentru comunicarea prin internet, display-uri LCD, senzori de temperatură, umiditate, culoare, capacitiv, ultrasonic. Acestea precum şi faptul că softul este gratis şi plăcile sunt relativ ieftine au dus la crearea unei comunităti bogate de utilizatori care au ajutat la crearea şi dezvoltarea codurilor pentru o groază de proiecte Arduino.

## **2.2. SOFTWARE**

Arduino IDE (Integrated Development Environment) este o aplicație cross-platform scrisă în Java şi care îşi are originile în mediul de dezvoltare PROCESSING şi în proiectul Wiring.Acesta foloseşte o versiune simplificată a limbajului C++ simplificând programarea în acest mediu însă păstrând totodată posibilitatea de a dezvolta aplicații complexe, acestea culminând cu lansarea în spațiu a unui satelit(ArduSat).

Softul Arduino utilizează conceptul de „Sketches”(schițe), fiind „un loc standard de stocare a programelor(schițelor)”[6], structura programului fiind constituită din două funcții majore,setup() fiind o funcție care se rulează o singură dată la începutul programului și are ca scop inițializarea pinilor și începutul comunicației seriale, și funcția loop() care rulează în mod repetat până la oprirea alimentării plăcuței.



Figura 2.1 Prezentarea unei „schite” in Arduino

## **2.3. HARDWARE**

Din punct de vedere hardware, plăcuța Arduino conține un microcontroller din familia Atmel AVR de 8, 16 sau 32 biți, pini analogici și digitali care permit conectarea cu diverse componente hardware precum și circuite pentru alimentare.Microcontrolerul instalat pe Arduino vine pre-programat cu un bootloader care simplifică încărcarea programelor pe memoria flash a cipului, în comparație cu alte dispozitive care necesită programatoare externe. „Acest aspect face Arduino o soluție simplă, permițând programarea de pe orice computer ordinar.”[7]

În prezent există mai multe variante de plăci Arduino, începând de la clasicul Arduino Uno, la Arduino Mega, Lilypad, Leonardo, Titan, Ethernet etc.

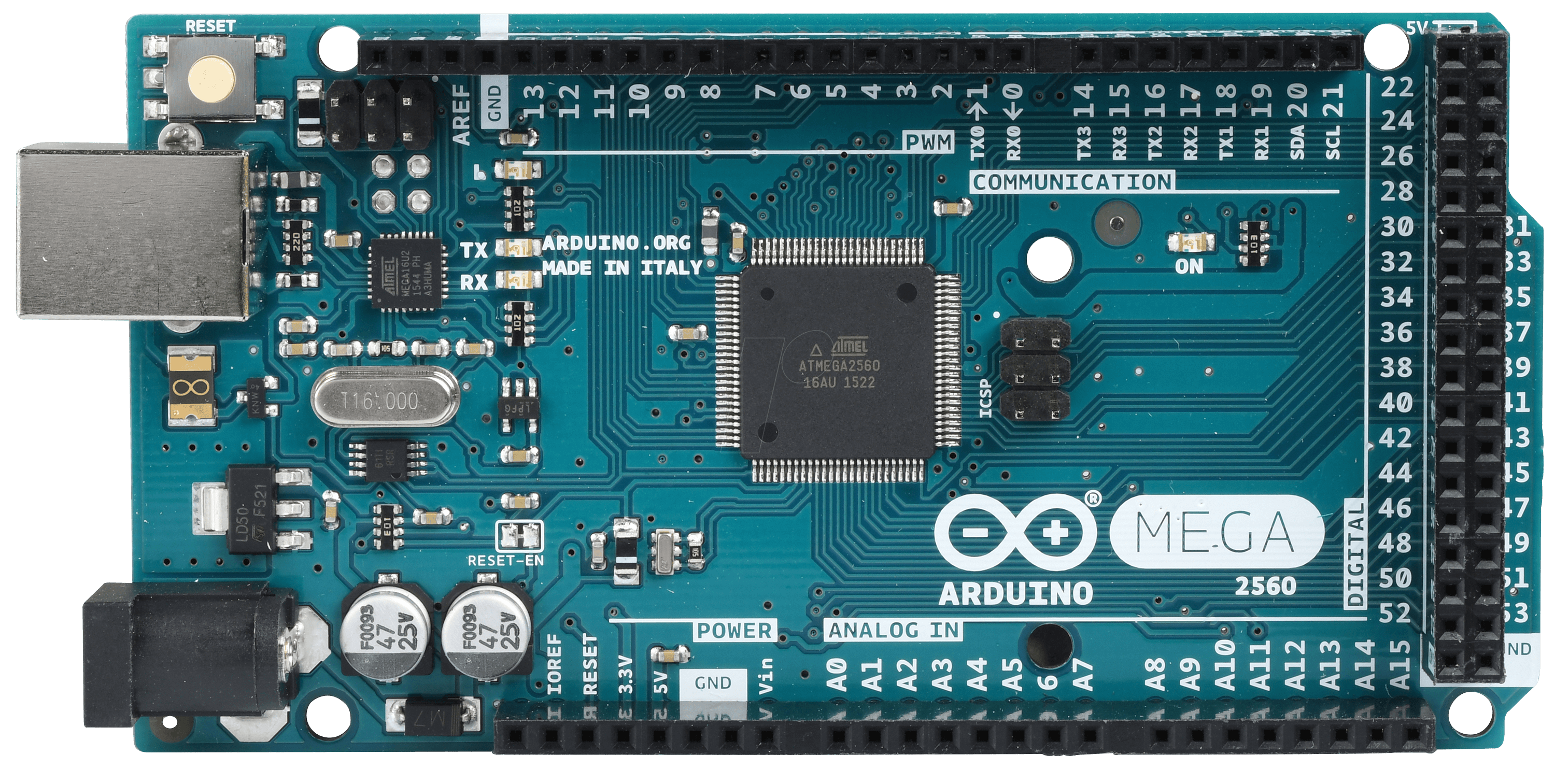


Figura 2.2 Arduino Mega

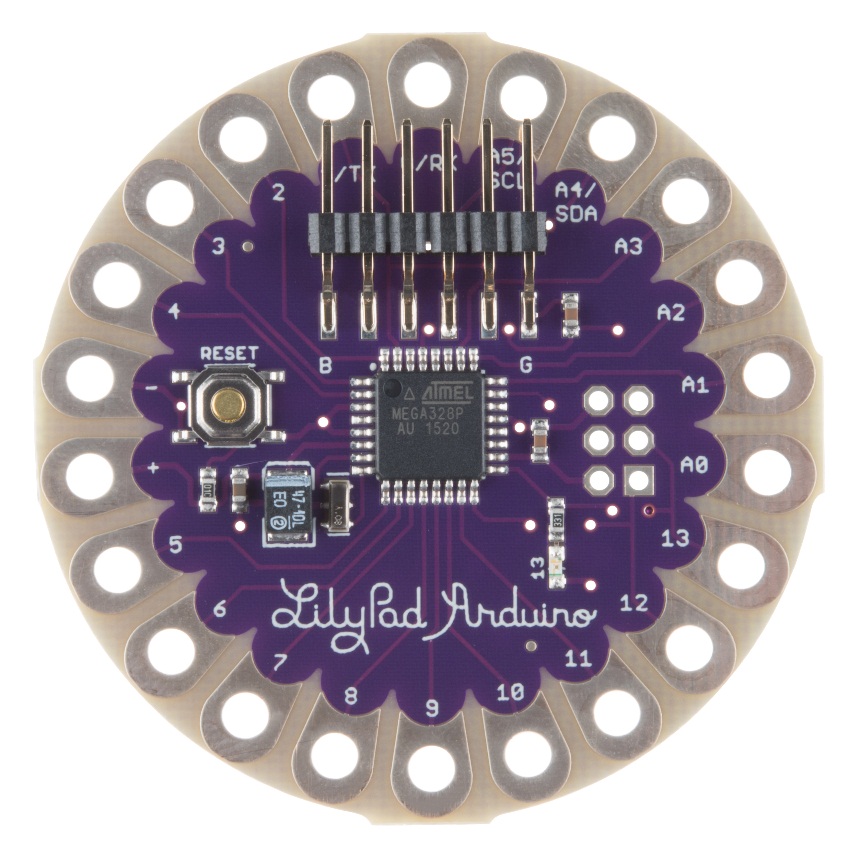


Figura 2.3 Arduino Lilypad

## **2.4. ARDUINO UNO**

Arduino Uno marchează prima versiune a Arduino IDE 1.0 și este una dintre cele mai populare plăci de dezvoltare, popularitate câstigată în special pe baza raportului calitate/preț întrucât oferă 14 pini digitali, dintre care 6 pot fi utilizați ca Output-uri PWM, 6 pini analogici, un cristal de 16 Mhz, o conexiune USB și un buton de reset, suficient pentru a dezvolta de la cele mai simple aplicații, până la unele mai complexe.

**Specificatii:**

|  |  |
| --- | --- |
| Microcontroller | [ATmega328P](http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf) |
| Voltajul de operare | 5V |
| Voltajul de Input (recomandat) | 7-12V |
| Voltajul de Input (limită ) | 6-20V |
| Pini Digitali I/O | 14 (dintre care 6 oferă Output PWN) |
| Pini Digitali PWM I/O | 6 |
| Analog Input Pins | 6 |
| Curent continuu/ pin I/0 | 20 mA |
| Curent continuu / pin 3.3V | 50 mA |
| Memorie Flash | 32 KB (ATmega328P) dintre care 0.5 folosiți de Bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| frecvență | 16 MHz |
| Lungime | 68.6 mm |
| Lățime | 53.4 mm |
| Greutate | 25 g |

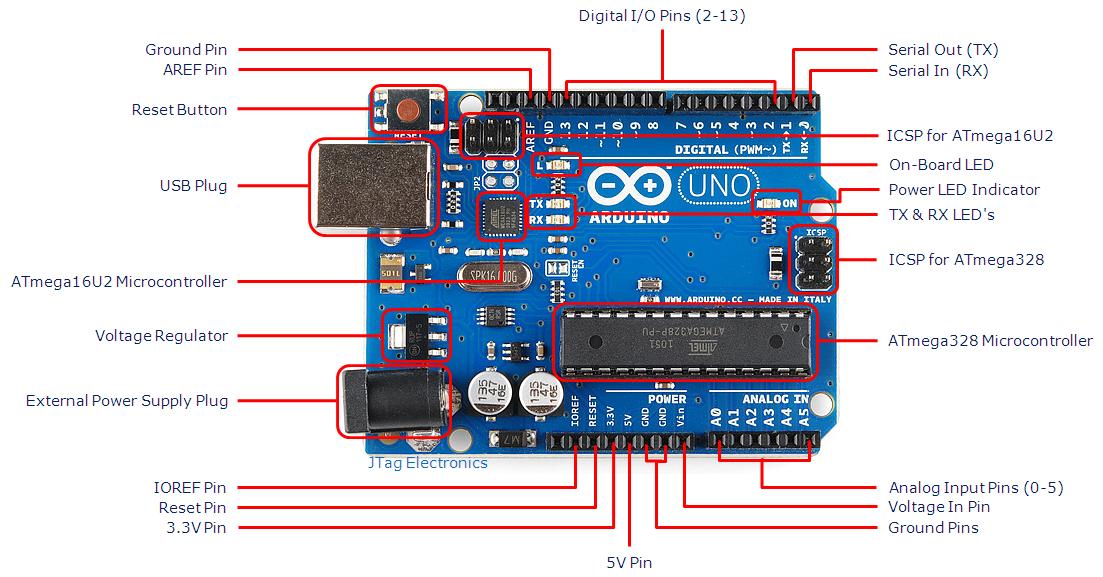
****

Figura 2.4 Arduino Uno

# **CAPITOLUL 3 APLICAȚIE PRACTICĂ**

Ca aplicație practică în acest domeniu am realizat un robot care, cu ajutorul unei rețele neuronale modeste, este capabil să urmărească un traseu, demonstrând astfel că rețelele neuronale pot realiza lucruri practice în acest domeniu al roboticii.Lucrarea a avut 2 părți: implementarea fizică a robotului, precum și implementarea algoritmului cu care acesta este capabil să se deplaseze în cadrul traseului.

## **3.1. PARTEA FIZICĂ**

Începutul părții fizice a constat din montarea șasiului, alături de placa Arduino Uno, precum și senzorii și modulele necesare pentru a dezvolta un robot capabil să se deplaseze.

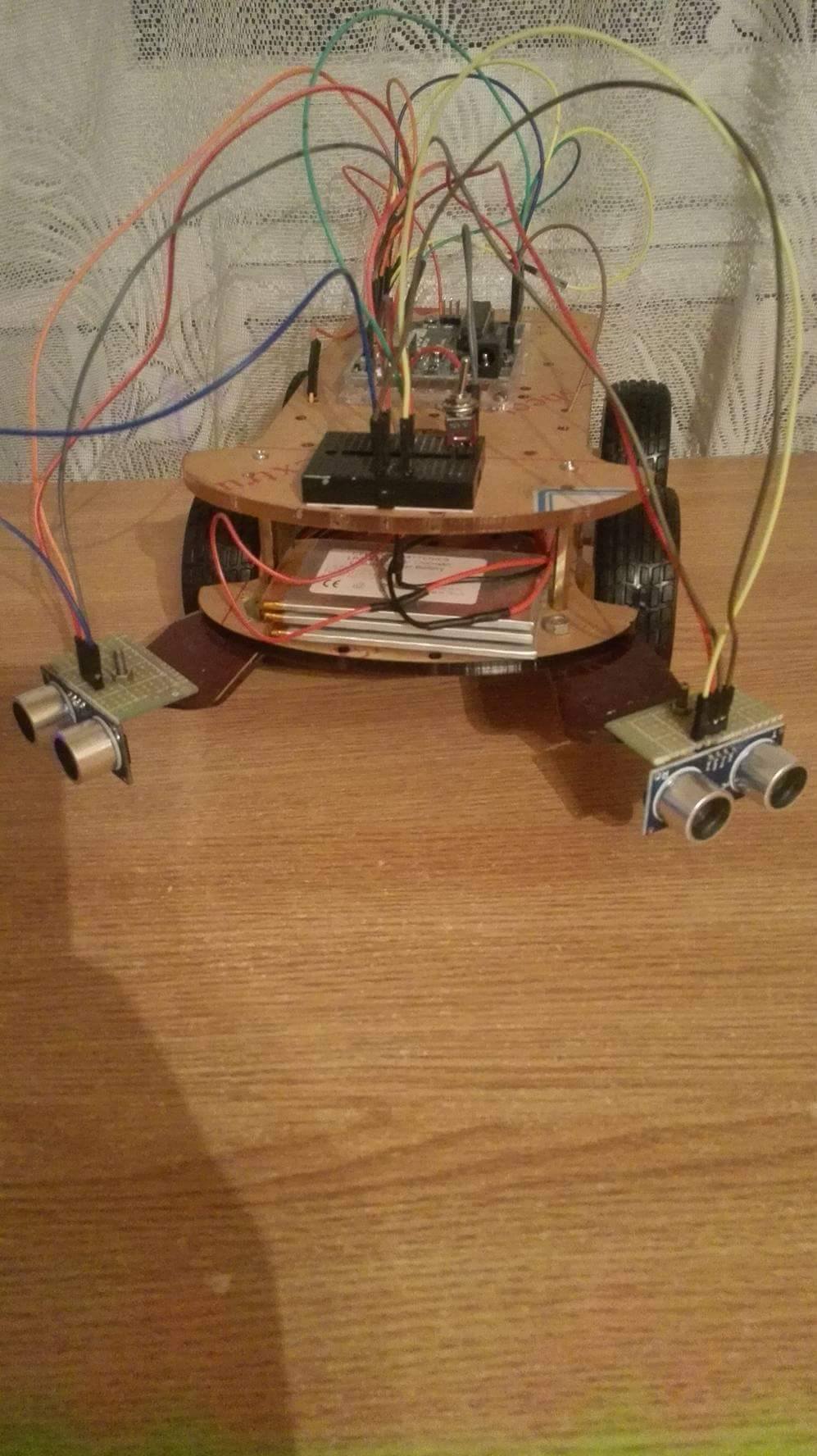


Figura 3.1 Implementarea fizică

Pentru a controla viteza și direcția motoarele cu curent continuu a robotului, este nevoie de un motor shield L298N, care este un amplificator de putere care poate lua puterea din Arduino sau de la baterii și să o transmită motoarelor.Conectarea acestuia cu Arduino se face prin modulele:

Modul 5V (sau Vcc) - Arduino 5V

Modul GND - Arduino GND

Modul 12V – către sursa de putere externă,suportă până la 35V.

Modul output 1 & 2 - motor A

Modul output 3 & 4 – motor B

Module IN1,IN2,IN3,IN4,ENA,ENB - pini digitali Arduino

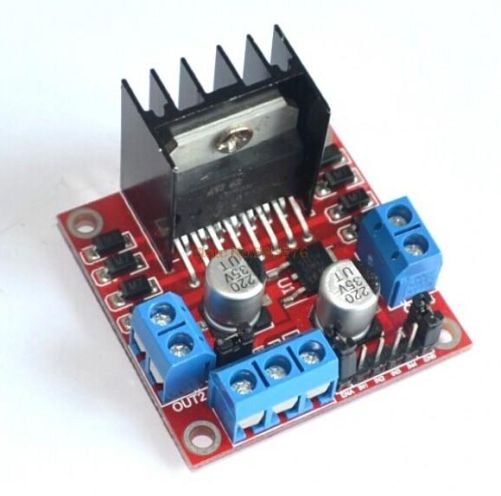


Figura 3.2 Motor Shield L298N

Întrucât Arduino oferă o tensiune de ieșire de 5V, aceasta era prea mică pentru a putea face robotul să se miște,datorită greutății acestuia.Astfel am decis să conectez 3 baterii LiPo, oferind o tensiune totală de 11.1 V .



Figura 3.3 Baterie LiPo

Pentru pornirea și oprirea alimentării robotului am folosit un switch(figura 3.4) iar pentru a facilita conectarea circuitelor electrice am folosit un breadboard(figura 3.5).



Figura 3.4 Switch on/off



Figura 3.4 Breadboard

De asemenea am folosit și 2 senzori ultrasonici HC-SR04,aceștia emițând un semnal ultrasonor la 40 000 Hz care călătorește în direcția transmisă până când întâlnește un obstacol, moment în care se reflectă înapoi la senzor.Pe baza duratei de timp dintre momentul în care unda a fost trimisă până când a fost receptată, precum și cunoscând viteza sunetului, se poate stabili cu exactitate distanța dintre senzor și obiect.



Figura 3.5 Senzor Ultrasonic

Așa cum se observă în figura 3.5, senzorul are 4 pini de conectare cu Arduino,ceea ce face simplă conectarea acestuia la circuit,astfel pinul VCC este legat la pinul 5V de la Arduino, GND la Arduino GND, Trig si Echo sunt legați la pini digitali, Trig fiind pinul de transmisie iar Echo pinul de recepție.

Pentru a genera semalul ultrasonic, pinul Trig trebuie menținut în starea HIGH timp de 10 uS,astfel se vor trimite 8 cicluri de semnale ultrasonice, ce vor fi receptate de pinul Echo după cum se poate vedea în figura. următoare:

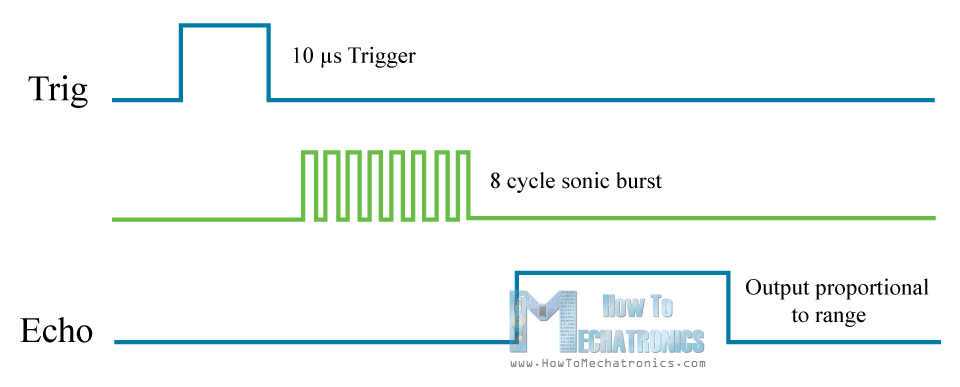


Figura 3.6 Schema de functionare a senzorului ultrasonic

Viteza aproximativă a sunetului în aer uscat, este dat de formula:c = 331.5 + 0.6 \* [temperatura aerului în grade Celsius].Cunoscând asta, aflarea distanței se poate afla din formula vitezei v = d/t având viteza sunetului fiind de aproximativ 340m/s sau 0,034cm/uS , iar timpul fiind aflat odată cu recepția semnalului, însă o altă metodă mai folosită de a afla distanța este împărțirea timpului la 29.1,acesta fiind distanța care a fost parcursă de la trimiterea semnalului până la recepția acestuia,însă pentru a afla distanța de la senzor la un obiect se recurge la formula:

Distanța = (timpul / 29.1) / 2

Cu aceste componente s-a putut realiza partea hardware a proiectului, schema electrică a robotului fiind in figura următoare:

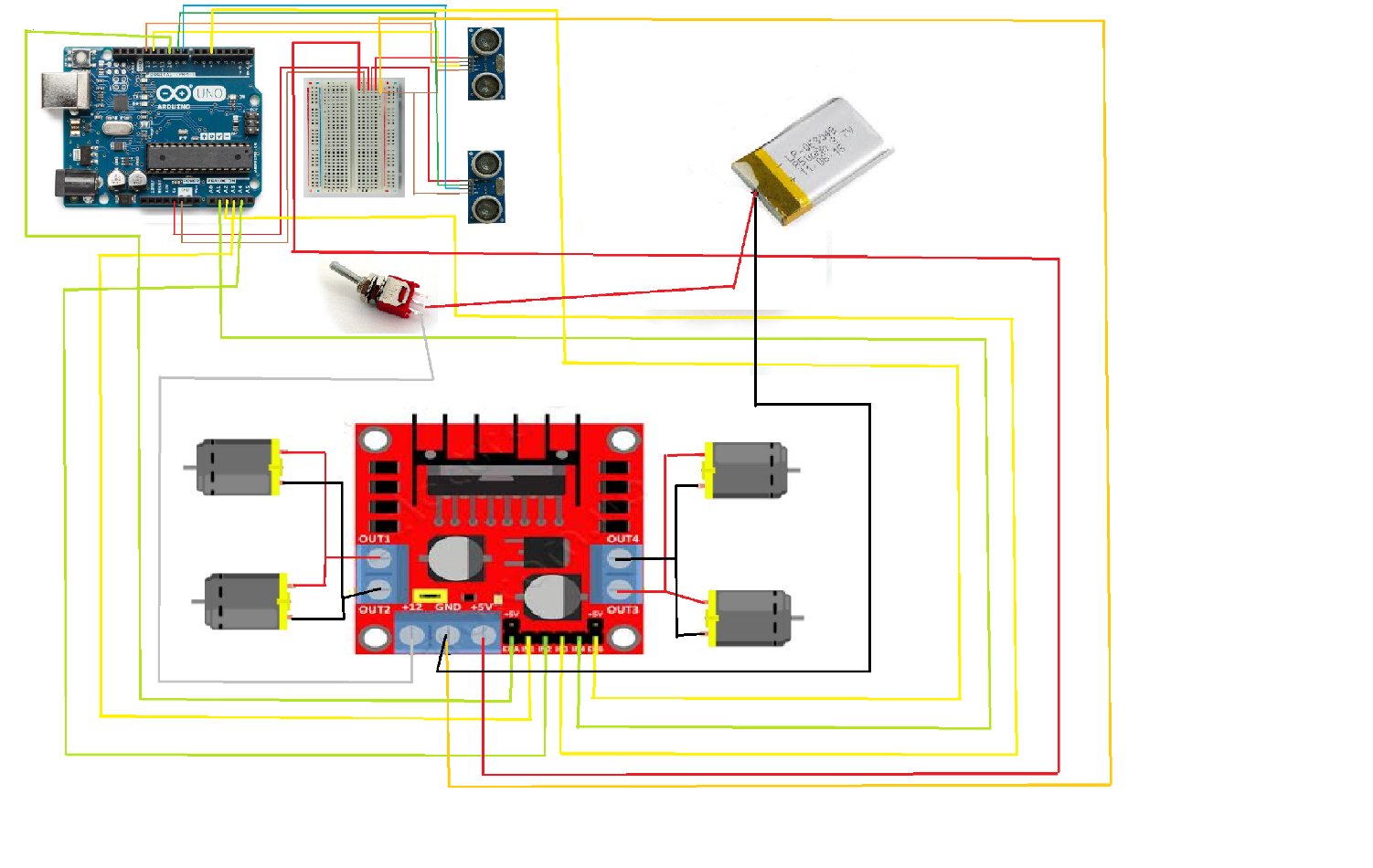


Figura 3.7 Schema electrică

## **3.2. PARTEA SOFTWARE**

Ca implementare software am folosit mediul Arduino IDE, unde am construit o rețea neuronală capabilă să direcționeze robotul pe traseu.Rețeaua neuronalăeste alcatuită din 3 straturi, un strat de intrare format din 2 neuroni, un strat ascuns format din 4 neuroni și un strat de ieșire format din 2 neuroni,după cum se poate vedea in figura. Următoare



Figura 3.8 Structura retelei neuronale

Astfel pe stratul de intrare, valorile corespunzătoare celor 2 neuroni sunt date de valorile celor 2 senzori ultrasonici iar pe stratul de ieșire valorile sunt date de perechea 0 1 pentru mersul la dreapta,1 0 pentru mersul la stanga și 1 1 pentru mersul înainte.

Mulțimea de antrenament este una modestă, de doar 25 de date, suficiente pentru a oferi rețelei neuronale capacitatea de a generaliza informațiile.Datele de intrare au fost normalizate pentru a oferi valori cuprinse între 0 și 1 cu formula: Dn = Val / (Vmin + Vmax), unde Dn reprezintă datele normalizate, Val fiind valoarea distanței în cm a senzorului ultrasonic, iar Vmin și Vmax fiind Valoarea minimă și valoarea maximă, în cazul de față Vmin fiind = 2 iar Vmax = 40. Datorită faptului că Arduino nu permite preluarea datelor din fișiere excel, aceste date au trebuit să fie stocate sub forma matricilor astfel:

float Input[elementNr][InputNodes] = {

//to the right

{ 0.089, 0.225 }, { 0.120, 0.175 },{ 0.074, 0.35 }, { 0.150, 0.15 }, { 0.062, 0.275 }, { 0.075, 0.210 },{ 0.078, 0.185 },

//to the left

{ 0.34, 0.076 },{ 0.27, 0.052 }, { 0.175, 0.042 }, { 0.225, 0.04 }, { 0.30, 0.120 }, { 0.180, 0.08 }, { 0.265, 0.110 },

//forward

{ 0.5, 0.70 }, { 0.75, 0.48 },{ 0.37, 0.30 },{ 0.15, 0.79 },{ 0.25, 0.15 },{ 0.33, 0.16 },{ 0.75, 0.41 },{ 0.125, 0.126 },{ 0.130, 0.260 },{0.143, 0.2 },{ 0.136, 0.127 }

};

const byte Target[elementNr][OutputNodes] = {

//go right

{ 0, 1 }, { 0, 1 }, { 0, 1 },{ 0, 1 },{ 0, 1 },{ 0, 1 },{ 0, 1 },

//go left

{ 1, 0 },{ 1, 0 },{ 1, 0 }, { 1, 0 },{ 1, 0 },{ 1, 0 },{ 1, 0 },

//go forward

{ 1, 1 }, { 1, 1 }, { 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 },{ 1, 1 }

};

Antrenarea rețelei se face pe baza mulțimii de antrenament, aceste date fiind parcurse până când eroarea scade sub pragul setat de 0.3, deoarece această eroare este neglijabilă pentru aplicația noastră.Pentru a vizualiza procesul de invățare precum și scăderea erorii, am decis să transmit output-ul curent, output-ul dorit, precum si eroarea o dată la 1000 de cicluri.

Unul dintre cele mai mari riscuri în implementarea rețelei neuronale este că, în timpul învățării aceasta să ajungă într-un punct de minim local care este mai mare decât punctul de minim global.În această situație se poate ajunge prin 2 moduri: fie rata de învățare este prea mică și rețeaua nu ajunge în punctul de minim global decât dupa zeci de mii de cicluri, fie rata de invățare este prea mare iar rețeaua tinde să "sară" peste acel punct de minim global. Pentru a reduce aceste riscuri am folosit o rată de invățare medie: int learningRate = 0.2, precum si bias-ul ,care este un termen extra adăugat fiecărui strat cu scopul de a face rețeaua mai robustă, fiind mai tolerantă la modificări prin posibilitatea de a modifica curba pe axa x, fără a mai fi nevoie ca aceasta să treacă prin origine.Astfel pentru bias am folosit o variabilă Accum care va avea o tărie sinaptică, astfel:

float Accum;

/\* Compute hidden layer activations \*/

for( i = 0 ; i < HiddenNodes ; i++ ) {

//hiddenWeights[InputNodes] adds an extra node at the end, which is the bias

Accum = HiddenWeights[InputNodes][i] ;

for( j = 0 ; j < InputNodes ; j++ ) {

//the extra neuron is multiplied with a weight

Accum += Input[p][j] \* HiddenWeights[j][i] ;

}

Hidden[i] = 1.0/(1.0 + exp(-Accum)) ;

}

/\* Compute output layer activations and calculate errors \*/

for( i = 0 ; i < OutputNodes ; i++ ) {

Accum = OutputWeights[HiddenNodes][i] ;

for( j = 0 ; j < HiddenNodes ; j++ ) {

Accum += Hidden[j] \* OutputWeights[j][i] ;

}

Output[i] = 1.0/(1.0 + exp(-Accum)) ;

}

De asemenea am folosit conceptul de Momentum, care este folosit pentru optimizarea unei rețele neuronale și precizează ce parte din iterația precedentă va afecta iterația curentă.Astfel invățarea va fi mai eficientă și se va reduce considerabil posibilitatea rețelei de a se bloca într-un punct de minim local.

Cele 2 valori ale neuronilor din stratul de intrare sunt puse într-un vector cu o linie și 2 coloane iar citirea senzorilor precum și calculul pentru obținerea unei decizii aferente și mișcarea robotului sunt urmatoarele:

float sensorArray[1][2];

//first sensor normalised data

sensorArray[1][0] = distanceB / 40;

//second sensor normalised data

sensorArray[1][1] = distance / 40;

for( i = 0 ; i < HiddenNodes ; i++ ) {

Accum = HiddenWeights[InputNodes][i] ;

for( j = 0 ; j < InputNodes ; j++ ) {

Accum += sensorArray[1][j] \* HiddenWeights[j][i] ;

}

Hidden[i] = 1.0/(1.0 + exp(-Accum)) ;

}

for( i = 0 ; i < OutputNodes ; i++ ) {

Accum = OutputWeights[HiddenNodes][i] ;

for( j = 0 ; j < HiddenNodes ; j++ ) {

Accum += Hidden[j] \* OutputWeights[j][i] ;

}

improvisedTarget[i] = 1.0/(1.0 + exp(-Accum)) ;

}

if (improvisedTarget[0] < 0.5 && improvisedTarget[1] > 0.5 ) {

//go right

digitalWrite(dir1MotorA,HIGH);

digitalWrite(dir2MotorA,LOW);

analogWrite(pinTuratieA,95);

digitalWrite(dir1MotorB,HIGH);

digitalWrite(dir2MotorB,LOW);

analogWrite(pinTuratieB,95);

Serial.print("right");

} else if (improvisedTarget[0] > 0.5 && improvisedTarget[1] < 0.5 ) {

digitalWrite(dir1MotorA,LOW);

digitalWrite(dir2MotorA,HIGH);

analogWrite(pinTuratieA,95);

digitalWrite(dir1MotorB,LOW);

digitalWrite(dir2MotorB,HIGH);

analogWrite(pinTuratieB,95);

Serial.print("left");

} else if (improvisedTarget[0] > 0.5 && improvisedTarget[1] > 0.5 ) {

//mers inainte

digitalWrite(dir1MotorA,LOW);

digitalWrite(dir2MotorA,HIGH);

analogWrite(pinTuratieA,95);

digitalWrite(dir1MotorB,HIGH);

digitalWrite(dir2MotorB,LOW);

analogWrite(pinTuratieB,95);

Serial.print("forward");

}

}

 Astfel se poate observa schema de functionare a robotului

Figura 3.9 Schema functionare robot

Precum si diagrama de stare:

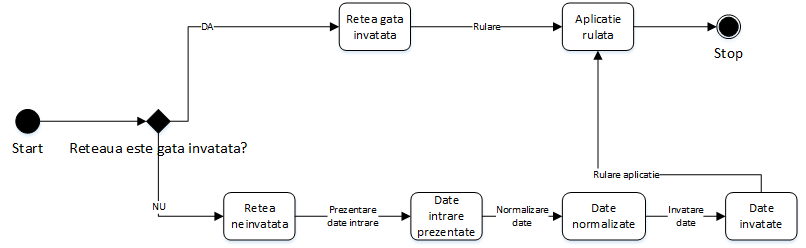


Figura 3.10. Diagrama de Stare

# **CAPITOLUL 4 CONCLUZII**

În opinia mea domeniul rețelor neuronale este unul foarte vast și cu o largă aplicabilitate practică.Acestea au fost concepute din dorința ca un calculator să înceapă să abordeze problemele precum noi oamenii însă acest domeniu este înca nou iar descoperirile sunt încă la inceput.Cu toate acestea consider că pe viitor se vor implementa aplicații mult mai complexe și în curând ne vom afla într-o nouă "revoluție tehnologică".

În lucrarea de față am reusit să demonstrez că rețelele neuronale pot fi folosite cu succes și pe plăcuțe modeste precum este Arduino Uno, în ciuda faptului că eram oarecum limitat de memoria fizică a acestuia.Cu aceste aspecte putem considera că pe viitor rețelele neuronale se vor dezvolta mult mai mult în acest domeniu al roboticii iar în curând limitarea memoriei nu va mai constitui o problemă.

Această lucrare se poate dezvolta pe viitor punând obstacole în cadrul traseului și astfel robotul fiind capabil să evite și obstacolele pe lângă mersul pe drum.Astfel, aplicația ar aborda un scenariu mult mai real, fiind un mini-prototip pentru mașinile autonome care sunt capabile să se deplaseze fără vreun control al șoferului.Consider că pe viitor mașinile nu vor mai fi controlate decât în cazuri exceptionale, realizându-se o automatizare în acest domeniu,având bineînteles numeroase beneficii atâtpentru om cât și pentru societate.

# **CAPITOLUL 5 BIBLIOGRAFIE**

[1] Enăchescu Călin , Calculul Neuronal ; Editura "Casa Cărţii de Ştiinţă", Cluj-Napoca 2008

[2] <https://en.wikipedia.org/wiki/Artificial_neural_network>

[3] <https://en.wikipedia.org/wiki/Reinforcement_learning>

[4][https://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol4/cs11/report.html#Neural%20networks%20versus%20conventional%20computers](https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Neural networks versus conventional computers)

[5] Introducere in Arduino – Optimus Digital

[6] <https://www.arduino.cc/en/guide/environment>

[7] <https://ro.wikipedia.org/wiki/Arduino>

[8]Neural Networks - A Systematic Introduction – Raul Rojas

[9]http://arduinobasics.blogspot.ro/2011/08/

[10] http://www.descopera.org/inteligenta-artificiala/