**UNIVERSITATEA “ALEXANDRU IOAN CUZA” FACULTATEA DE INFORMATICĂ**



**LUCRARE DE LICENŢĂ**

**Smart Car**

**Absolvent: Coordonator științific:**

**Bartic Andrei Drd. Crăciun Vlad**

Sesiunea: *iulie, 2017*

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” FACULTATEA DE INFORMATICĂ**

**Smart Car**

# Bartic Andrei

# Sesiunea: *iulie, 2017*

**Coordonator științific**

**Drd. Crăciun Vlad**

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Smart Car*” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
* codul sursă, imaginile etc. preluate din proiecte *open*-*source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
* rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, *data*

Absolvent *Prenume Nume*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Smart Car*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date detest etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

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

Iaşi, *data*

Absolvent *Prenume Nume*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Cuprins**

Introducere ……………………………………………………………………4

Capitolul 1. Platforma Arduino ……………………………………………….7

* 1. Hardware …………………………………………………………..13
  2. Cinci mituri despre Arduino……………………………………….19
  3. Mediul de dezvoltare și tehnologiile utilizate……………………..20

1.4 Alte aplicații ale platformei Arduino................................................21

Capitolul 2. Prezentare generală a proiectului ………………………………...21

2.1 Interacțiunea cu mediul……………………………………………22

2.2. Mișcarea…………………………………………………………..24

2.3 Comunicația……………………………………………………….29

2.4 Concepte și echipamente………………………………………….30

Capitolul 3. Descrierea proiectului……………………………………………33

3.1 Implementare……………………………………………………...33

3.2 Aplicația desktop………………………………………………….38

3.3 Proiectare hardware ……………………………...……………….39

3.4 Librarii folosite……………………………………………………43

Concluziile lucrarii……………………………………………………………43

Bibliografie…………………………………………………………................45

**Introducere**

Creierul uman este o mașinarie incredibilă care poate realiza un numar impresionant de calcule pe secunda. Această capacitate impresionantă de care dispunem este des folosită in timpul uneia dintre cele mai provocatoare activitați din punct de vedere neurologic, in care oamenii sunt implicați aproape zi de zi și anume, conducerea unei mașini.

Numeroase zone ale creierului actionează impreună pentru a procesa in timp real informatiile preluate in timpul șofatului. Aceste procese complexe ar putea trece neobservate de către conducatorul auto, insă funcționarea lor neîntrerupta este cruciala. Diferența dintre viață și moarte poate fi la o distantă de numai 100 de miimi de secundă in timpul de reacție. La viteze mari, această perioadă poate fi crucială atunci cand vine vorba de evitarea unui obstacol si implicit a unui accident care în multe cazuri s-a dovedit a fi fatal.

Din acest motiv, o invenție revoluționara ce are capacitatea de a reduce aceste pericole si potențialul de a salva milioane de vieți pe șosele sunt mașinile autonome, care intr-un viitor foarte apropiat ar putea deveni o realitate curenta. Mulți oameni au auzit de aceasta inovație, insa nu mulți își dau seama cât de repede se apropie și cât de mult va schimba această invenție societatea.

Se estimează că accidentele de automobile la nivel mondial in 2016 au costat viața mai mult de 1.1 milioane de oameni, in timp ce mai mult de 31 de milioane de persoane au fost ranite. Odată ce această tehnologie va deveni un lucru obișnuit și mașinile autonome vor fi omniprezente, aceste cifre vor scădea la o mica fracțiune din cât reprezinta astazi.

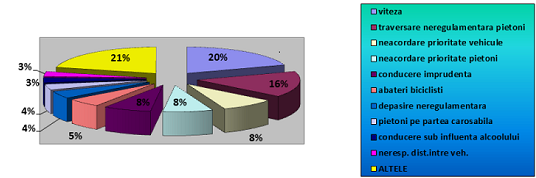


Fig. 1: Accidentelele de circulație rutieră și cauza producerii lor

După cum vedem, există nenumărate cauze ale producerii accidentelor rutiere. O masină autonomă nu numai ca ar ajuta la scăderea numărului de accidente dar ar putea preveni multe situații în care s-ar putea produce un accident. De multe ori, ca și conducatori auto suntem influențați de diverse lucruri nesemnificative cum ar fi telefonul, dialogul purtat de ceilalți pasageri sau de ce nu decorul. Acestea sunt doar cateva exemple de greșeli care pot fi evitate cu ajutorul unui sistem automatizat ce ar putea controla mașina in locul nostru și implicit ar evita o tragedie.

Impactul social va fi chiar mult mai mare, la o proporție care este foarte greu de cuantificat în prezent. Mașinile autonome vor face ca serviciul de folosire în comun a mașinilor sa fie mult mai ușor și mai eficient.

Acest lucru va însemna o mai mică poluare a mediului prin reducerea consumului de combustibili, blocaje mai puține în trafic, mai puține ore petrecute pe șosele și de ce nu mai puțin timp alocat găsirii unui loc de parcare.

Companii precum Google, Uber sau Tesla au ajuns destul de departe in dezvoltarea acestui proiect, cei de la Uber susținînd ca vor începe testele unei mașini autonome in 2016.

Piața mașinilor autonome ramane, totuși, una plina de controverse.

Recent, Uber a anunțat ca suspenda activitatea flotei de vehicule autonome, în așteptarea rezultatelor unei anchete ce vizeaza un accident, in Arizona, in care a fost implicat unul din vehiculele sale. Funcționalitatea de pilot automat a început testarea la sfarșitul anului 2016 in San Francisco, unde a fost foarte repede interzisă. Uber a mutat aceste operațiuni în Arizona și in Pittsburgh, unde totul a mers bine pana cand una dintre mașini a fost implicata într-un accident. In ciuda faptului ca mașina autonomă circula regulamentar, șoferul celeilalte mașini fiind de vina pentru ca nu a acordat prioritate, Uber a decis sa sisteze aceste activitați continuând însa dezvoltarea software-ului.



Fig. 2: Mașina autonomă Uber implicată în accident

După cum vedem, incă o dată eroarea umană și-a făcut simțita prezența, în acest accident fiind vinovat șoferul din cealaltă mașina, nicidecum mașina autonomă.

În viziunea mea, acesta nu ar fi un simplu proiect în care doar o mașină autonomă este implicata. O întreagă armată de oameni ar trebui să creeze un mediu cât mai favorabil pentru ca interacțiunea dintre mașină și mediul înconjurator sa fie una cât mai precisa. Un exemplu ar fi străzile care trebuie sa fie cât mai corect delimitate si semnalizate pentru ca atunci când senzorii acelei mașini sunt puși în funcțiune să citeasca niste date clare, lipsite de erori. Deci infrastructura ar putea da bătăi de cap unui asemenea proiect mai ales dacă este testat intr-o zonă rurala unde infrastructura lipsește cu desăvarșire uneori.

Am ajuns la concluzia că societatea în care trăin nu mai poate face față provocarilor de zi cu zi fară o mână de ajutor din partea tehnologiei. În contextul în care ne referim strict la mașina autonomă cred că ar fi timpul sa își facă simțita prezența și să ne ofere o mană de ajutor.

Influențat de tendințele multor companii de a crea o așa zisă mașină autonomă, îmi propun prin intermediul acestei lucrări realizarea unui astfel de automobil folosind platforma Arduino. Voi incerca să creez un mediu cât mai real atunci când va fi testată mașina, de asemenea voi simula situații cât mai reale pe care o mașină obisnuită le poate intâlni.

Capitolul I va conține o amplă descriere a platformei Arduino, fiind prezentată atât partea software a platformei cât și partea hardware. De asemenea mi-am propus să descriu obiectivele acestui proiect precum și o eventuală aplicare a acestuia în viața reală.

Capitolul II va conține o prezentare generală a proiectului și a obiectivelor pe care mi le-am propus să le ating prin intermediul acestui proiect. În acest capitol vor fi prezentate și componentele folosite pentru realizarea proiectului.

In capitolul III vor fi scoase in evidenta detaliile tehnice ale proiectului care vor cuprinde atat partea hardware cât și partea software. Partea hardware va fi prezentata intr-o maniera cât mai succinta pentru a ințelege interacțiunea dintre componentele fizice ale acestui proiect și mai ales cum ajută o componentă la crearea obiectivului principal.

Partea software va beneficia de o atentie mai sporita deoarece consider ca este partea cea mai importanta a proiectului. In acest capitol vor fi detaliate si anumite aspecte teoretice pe care am incercat să le ating într-o manieră cât mai succintă, aici am putea vorbi de concepte pe care le intâlnim în programarea orientată pe obiecte, șabloane software sau de ce nu anumite librarii care au ajutat la realizarea unei interfețe grafice sau la punerea în funcțiune a unei componente hardware.

**Capitolul 1**

**Platforma Arduino**

Arduino este o companie open-source care produce atât placuțe de dezvoltare bazate pe microcontrollere, cât și partea de software destinată funcționării și programării acestora. Pe lângă acestea include și o comunitate uriașă care se ocupa de creația si distribuirea de proiecte care au ca scop crearea de dispozitive care pot sesiza și controla diverse activitați sau procese în lumea reala.

Proiectul este bazat pe designul plăcilor cu microcontroller produse de câțiva furnizori, folosind diverse tipuri de microcontrollere. Aceste plăci pun la dispoziția utilizatorului pini I/O, digitali și analogici, care pot fi interfațați cu o gamă largă de plăcuțe numite scuturi (shield-uri) si/sau cu alte circuite. Placile au interfețe de comunicare seriale, inclusiv USB pe unele modele, pentru a încarca programe dintr-un calculator. Pentru programarea microcontrollerelor, Arduino vine cu un mediu de dezvoltare integrat (IDE) bazat pe proiectul Processing, care include suport pentru limbaje de programare ca C și C++.

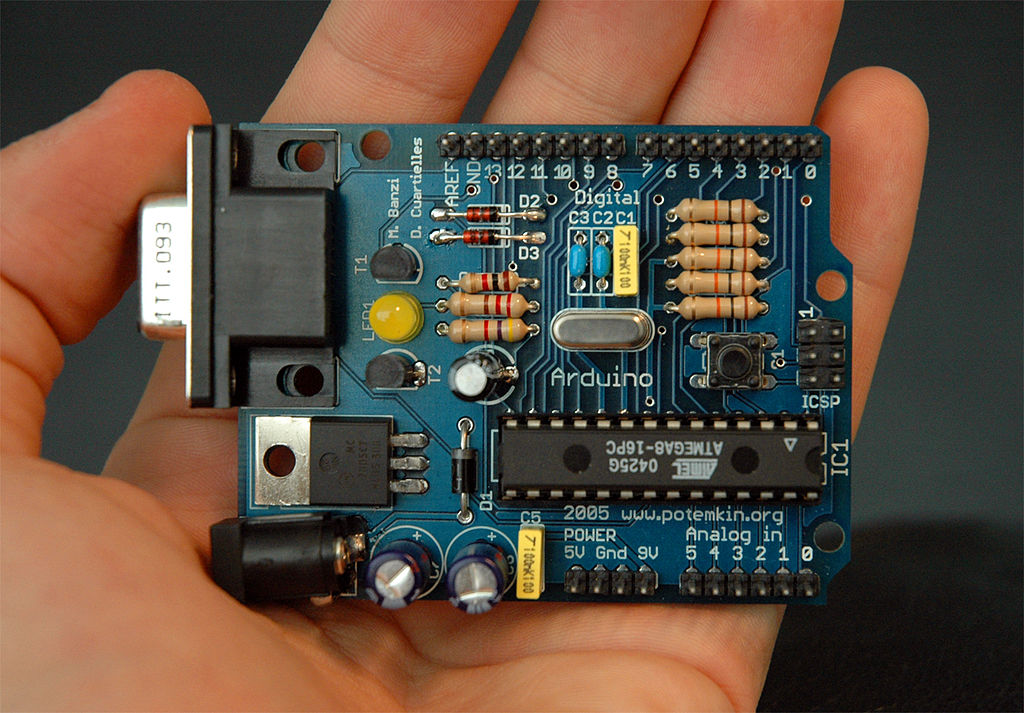


Fig. 3: Una dintre primele plăcuțe Arduino cu interfața de comunicații serială RS-232(stânga sus) și un microcontroller Atmel Atmega8(dreapta jos)

Primul arduino a fost lansat in 2005, având ca țintă asigurarea unei soluții ieftine și simple pentru începători și profesioniști spre a crea dispozitive capabile să interacționeze cu mediul, folosind senzori și sisteme de acționare. Cele mai comune exemple sunt dispozitivele pentru utilizatorii începatori precum roboții simpli, termostatele și/sau detectoarele de mișcare.

Placuțele Arduino sunt disponibile comercial sub formă preasamblată sau sub forma unor kituri de asamblat acasă. Specificațiile schemelor sunt disponibile pentru orice utilizator, permițînd oricui să fabrice plăcuțe Arduino. Arduino Industries estimase la mijlocul anului 2011 că peste 300.000 de placuțe oficiale Arduino au fost produse, iar in 2013 700.000 de placuțe oficiale erau în posesia utilizatorilor.

Arduino a început în 2005 ca un proiect al unui student de la Institutului de Interacțiune a Designului din Ivrea, Italia. La acea vreme studenții folosau o plăcuță de dezvoltare BASIC Stamp care costa 100 de dolari, ceea ce era considerat foarte scump pentru studenți. Massimo Banzi, unul dinete fondatori, era student la Ivrea. Numele „Arduino” provine de la un bar din Ivrea, locul unde o parte din fondatori obișnuiau să se întâlnească. Studentul columbian Hernando Barragan a creat o platformă de dezvoltare Wiring care a servit ca bază pentru Arduino. Dupa finalizarea platformei Wiring, mai multe versiuni, mai light și mai ieftine, au fost create la dispoziția comunitaților open-source. Din echipa inițială Arduino au facut parte Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino si David Mellis.

O plăcuță Arduino este compusa dintr-un microcontroller Atmel AVR de 8,16 sau 32 biți(deși începând cu 2015 s-au folosit microcontrollere de la alți producători) cu componente complementare care facilitează programarea și încorporarea în alte circuite. Un aspect important la Arduino este că acesta dispune de conectori standard, care permit utilizatorului să conecteze plăcuța cu procesorul la diferite module interschimbabile numite shield-uri. Unele shield-uri comunica cu Arduino direct prin pinii digitali sau analogi, dar altele sunt adresabile individual prin magistrala serial I²C permițând utilizarea mai multor module în paralel. Până in anul 2015 plăcuțele Arduino oficiale au folosit cipuri Atmel din seria megaAVR, în special ATmega8, ATmega168, ATmega328, ATmega 1280 și ATmega2560 iar în 2015 au fost adăugate cipuri de la alți producatori. O multitudine de alte procesoare au fost folosite de dispozitive compatibile Arduino. Multe plăcuțe includ un regulator linear de 5 V și un oscillator cu cuarț de 16 MHz (sau un rezonator ceramic în unele variante), deși anumite plăcuțe, cum ar fi LilyPad, funcționează la 8MHz și nu necesită regulator, datorită restricțiilor de formă. Un microcontroller instalat pe Arduino vine programat cu un bootloader care simplifică încarcarea programelor pe memoria flash a cipului, în comparație cu alte dispozitive care necesita programatoare externe. Acest aspect face Arduino o soluție simplă, prmițând programarea de pe orice computer ordinar. În present, bootloader-ul optibot este bootloader-ul implicit instalat pe Arduino UNO.

La nivel conceptual, când se folosește mediul de dezvoltare integrat Arduino, programarea tuturor plăcuțelor se face prin conexiune serială. Implementarea acesteia diferă în funcție de versiunea hardware. Unele plăcuțe Arduino au implementate convertoare de nivel logic pentru a realiza între nivelele logice RS-232 și cele TTL. Plăcuțele Arduino din prezent sunt programate prin USB, având integrate cipuri de conversie USB-serial, cum ar fi FTDI FT232. Unele modele UNO, mai noi, folosesc un cip AVR separat, programat să funcționeze ca un convertor USB-serial, care poate fi reprogramat printr-un port ICSP dedicat. Alte variante, cum ar fi Arduino Mini și versiunea neoficială Boarduino folosesc adaptoare detașabile USB-serial, cabluri, Bluetooth sau alte modele.

Placuța Arduino are expuși mulți dintre pinii de intrare/ieșire ai microcontrolerului, pentru ca aceștia sa fie folosiți de alte circuite. Diecimila, Diemilanove si UNO ofera 14 pini digitali de intrare/ieșire, dintre care 6 pot produce semnale PWM și 6 intrări analogice care, de asemenea, pot fi folosite ca intrări/ieșiri digitale. Acești pini sunt accesibili prin partea superioară a plăcuței, prin intermediul unor barete mama cu pasul între pini de 2.54 mm.

Plăcuțele originale Arduino erau produse de compania italiană Smart Projects. O parte dintre plăcuțele cu brandul Arduino au fost proiectate de companiile americane SparkFun Electronics și Adafruit Industries. 16 versiuni de hardware Arduino au fost produse în scop commercial până la această dată.

Programele Arduino pot fi scrise în orice limbaj de programare cu un compilator capabil să producă un cod mașină binar. Atmel oferă un mediu de dezvoltare pentru microcontrollerele sale, AVR Studio și mai nou Atmel Studio. Proiectul Arduino ofera un mediu integrat de dezvoltare(IDE), care este o aplicație cross-platform, scrisă in Java. Aceasta își are originile în mediul de dezvoltare pentru limbajul de programare Processing și în proiectul Wiring. Este proiectat pentru a introduce programarea în lumea artiștilor și a celor nefamiliarizați cu dezvoltarea software. Include un editor de cod cu funcții ca evidențierea sintaxelor, potrivirea acoladelor și spațierea automată și oferă mecanisme simple cu un singur click, pentru a compila și a încărca programele în plăcuța Arduino. Un program scris în IDE pentru Arduino se numește sketch.

Arduino IDE suportă limbaje de programare C și C++ folosind reguli speciale de organizare a codului. Arduino IDE oferă o librărie software numită Wiring, din proiectul Wiring, care ofera multe proceduri comune de intrare și ieșire. Un sketch tipic Arduio scris în C/C++ este compus din doua funcții care sunt compilate și legate cu un ciot de program main(), într-un program executabil cu o execuție ciclică.

* Setup(): o funcție care este rulată o singură dată la începutul programului, când se inițializează setarile.
* Loop(): o funcție apelată in mod repetat până la oprirea alimentării cu energie a plăcuței.

După compilarea și legarea cu GNU toolchain inclus, de asemenea, în IDE, mediul de dezvoltare Arduino trimite comandă catre programul avrdude pentru a converti codul executabil într-un fișier text codat hexazecimal, care poate fi încărcat în placa Arduino de un program de încarcare.

Un program Arduino tipic pentru un programator începător face ca un LED să se aprindă intermitent. Acest program este încărcat pe placă, în mod normal, de către producator.

Arduino are o platformă hardware open-source: referințe de design pentru Arduino sunt distribuite sub licența Creative Commons Attribution Share-Alike 2.5 și sunt disponibile pe situl Arduino. Schemele și fișierele de producție sunt și ele disponibile.

Codul sursă pentru IDE este disponibil sub GNU General Public Licence.

Pinii dispuși pe o platformă Arduino pot fi configurați ca fiind input sau output. Majoritatea pinilor analogi prezenți pe o platformă Arduino pot fi configurați și utilizați în aceeași maniera ca pinii digitali.

Arduino setează în mod implicit pinii ca fiind input deci nu trebuie să folosim funcția pinMode() si să setăm starea unui pin daca vrem să il folosim ca și input. Atunci când trecem un pin dintr-o stare în alta este nevoie de foarte puțin curent iar acest lucru îi poate face utili în indeplinirea sarcinilor ca implementarea unui senzor tactil, citirea semnalului generat de un fotorezistor sau citirea unui senzor analog.

Oricum am seta acel pin, neavând ceva conectat la el, acesta va primi semnale random deoarece captează semnale electrice din mediul înconjurator (zgomot).

In interiorul unui ATmega avem incorporați niste rezistori care pot fi accesați din software. Acești rezistori încorporați sunt accesați atunci când setăm pinMode() ca fiind INPUT\_PULLUP. Aceasta rezistență practic inversează comportamentul pinului când îl setăm ca fiind input, unde HIGH înseamnă că senzorul este oprit iar LOW înseamnă ca senzorul este pornit. Valoarea acestui rezistor depinde în funcție de microcontroller-ul utilizat. Majoritatea AVR-urilor au valoarea acelei rezistențe cuprinsă între 50kΩ și 150kΩ.

Acești rezistori sunt controlați de aceeași regiștri care controlează daca un pin este HIGH sau LOW.

Pinii configurați ca fiind OUTPUT pot furniza o cantitate substanțială de current la alte dispozitive. Pinii ATmega pot furniza current electric de până la 40mA altor dispozitive electrice. Există suficient current pentru a activa un LED sau a activa mai mulți senzori, dar nu destul curent pentu a alimenta mai multe relee sau motoare electrice.

Scurtcircuitele realizate pe un pin Arduino sau incercarea de a trece o cantitate mare de curent prin el(Motor DC) poate avaria sau distruge tranzistorul atașat pinului sau chiar distruge întreg cipul. De multe ori vom ramîne cu un pin nefuncțional, ceilalți pini fiind în continuare utilizabili.Din acest motiv este bine să conectam pinii de output la alte dispozitive cu ajutorul unui rezitor (470Ω-1k), asta daca nu avem nevoie de întregul curent disponibil pentru acea aplicație.

Funcția pinMode() ne ajută să trecem un pin din starea INPUT în starea OUTPUT. Cum am discutat și mai sus, este posibil cu ajutorul acestei funcții să activăm rezistorul intern cu ajutorul modului INPUT\_PULLUP. Adițional, modul INPUT poate dezactiva rezistența interioară.

Funcția digitalWrite() este una din cele mai folosite funcții din libraria Arduino.h. Aceasta ne ajută să inițializam un pin digital cu HIGH sau LOW. Daca pinul a fost configurat ca fiind OUTPUT cu ajutorul funcției pinMode(), voltajul acesteia va fi setat cu o valoare corespunzătoare de 5V sau 3.3V, atunci când folosim o placă alimentată la 3.3V. Deci dacă funcția are setat parametrul HIGH, atunci pinul la care facem referința va genera 5V iar daca este setat parametrul LOW atunci prin acel pin nu va fi generat nici un voltaj, 0V(ground). Dacă pinul este configurat ca și INPUT, digitalWrite() va permite semnal HIGH sau LOW cu pullup intern pe pinul de intrare. Se recomandă să setăm pinMode() înainte de a folosi acel pin digital ca și input pentru a activa rezistorul intern pull-up.

O altă funcție utilizată este digitalRead(). Această funcție citește valoarea de la un pin digital, valoarea fiind HIGH sau LOW. Dacă pinul nu este conectat la nici un dispozitiv extern, acesta poate capta zgomotul din mediul înconjurător și poate întoarce fie HIGH fie LOW într-un mod aleator.

Pentru folosirea semnalelor de tip PWM vom folosi funcția analogWrite(). Această funcție este folosită pentru a transmite valori analogice unui pin. De exemplu putem aprinde un led în mod treptat sau putem realiza un sistem de viteze pentru un motor electric. După ce apelăm funcția analogWrite(), pinul va genera un semnal stabil cu o valoare dată în funcție de parametrul pe care l-am setat atunci când am apelat funcția, acest semnal fiind transmis pană cand primim un apel la o alta funcție de scriere/citire asupra pinului. Frecvența semnalelor PWM pe majoritatea pinilor este de aproximativ 490 Hz. Pe Arduino Uno și pe plăci similare, pinii 5 și 6 au o frecvență de aproximativ 980 Hz. Pinii 3 și 11 de pe Arduino Leonardo rulează de asemenea cu o frecvență de 980 Hz. Pe majoritatea plăcilor Arduino, în special cele cu ATmega168 și ATmega328, această funcție ruleaza pe pinii 3,5,6,9,19,11. Pe Arduino Mega, funcția PWM rulează pe pinii 2-13 și 44-46. Versiunile mai vechi de Arduino avînd microcontrollere ATmega8 suportă analogWrite() doar pe pinii 9,10,11. Arduino Due suporta analogWrite pe pinii 2-13 plus pinii DAC0 și DAC1.

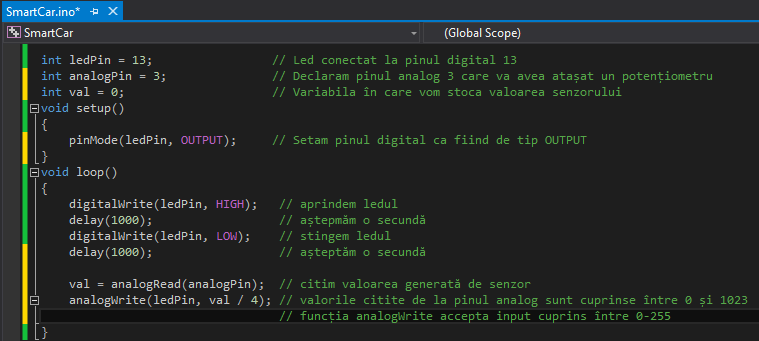
****

Fig. 4: Setarea unui pin și utilizarea funcțiilor digitalWrite/analogWrite/analogRead

Pulse width modulation sau PWM, este o tehnică prin care obținem semnale analogice cu mijloace digitale. Controlul digital este folosit pentru a crea un semnal care variază între pornit și oprit. Acest mod de tratare a semnalelor poate simula voltaje doar de tip minim/maxim ceea ce ne poate dezavantaja în unele aplicații. Pentru a obține valori variabile analogice, nu trebuie decât să schimbam lățimea pulsului generat de către un pin.

Dacă repetam acțiunea pornit/oprit de mai multe ori cu un led de exemplu, rezultatul va fi un voltaj cuprins între 0 și 5V, controlând intensitatea ledului sau de ce nu viteza unui motor electric.

Într-un semnal PWM, frecvența este constantă, cee ace diferă este fracțiunea de timp în care acel semnal este pornit, variind între 0 și 100%.

Semnalul PWM este necesar atunci cand vrem sa facem anumite componente electronice mult mai interactive, avînd la dispoziție o multitudine de variante de funcționare a unui periferic avînd ca și input un voltaj diferit de cel maxim, in cazul nostrum voltajul maxim generat de un pin Arduino fiind de 5V.

In cele ce urmeaza voi prezenta cîteva exemple in care semnalul PWM îsi face simțită prezența. Unul din cele mai comune exemple este folosirea semnalului PWM împreună cu un led. Diferența de voltaj la o anumită perioadă de timp va face ca ledul să fie aprins cu o anumită intensitate. În termeni mai tehnici, aici este vorba de o ieșire analogă filtrată de un pin digital care va produce voltaje cuprinse între 0% și 100%. Un alt exemplu de întrebuințare este impreună cu un difuzor. Pentru producerea diferitelor sunete, difuzorul are nevoie de un semnal PWM. Controlul vitezei unui motor electric este și el o bună întrebuințare a acestei tehnologii. In acest proiect, de exemplu a fost utilizat semnalul PWM pentru a simula un sistem de 6 viteze. La fiecare viteză se oferă pinului analog o valoare pe care o aflăm în funcție de viteza la care suntem.

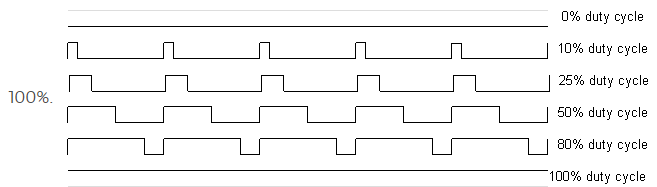
****

Fig. 5: Semnale PWM

Un semnal PWM poate fi implementat manual, fară folosirea funcțiilor din librăria Arduino.h. Aprinderea unui led în mod repetat la o anumita perioada de timp poate fi vazută ca un semnal PWM. Această tehnică este avantajoasă deoarece putem folosi orice pin digital ca fiind PWM. În plus avem control absolut asupra frecvenței și a ciclului de funcționare. Un dezavantaj al acestei tehnici este atunci când o întrerupere survine în timpul execuției programului, cauzând afectarea semnalului PWM.

ATmega168P/328P au 3 timere PWM care controlează 6 pini PWM de output. Manipulând timerii regiștrilor în mod direct, putem obține mai mult control decât atunci când folosim funcția analogWrite. ATmega328P are timere cunoscute ca și Timer 0, Timer 1 și Timer 2. Fiecare timer are 2 regiștri de comparere care controlează semnalul PWM pentru cele doua semnale de ieșire. Atunci când timerul ajunge la valoarea comparată de registru, outputul este transmis către pin. Cele două ieșiri pentru fiecare timer vor avea în mod normal aceeași frecvență, dar pot avea un ciclu de funcționare diferit.

Pe piață există o gamă foarte variată de sisteme de dezvoltare bazată pe microcontroller, avantajele pe care le are Arduino față de aceste sisteme fiind destul de numeroase incât să ne convingă să achiziționăm acest produs în defavoarea altuia.

Datorită simplității și accesibilitații pe care o oferă userului atunci când folosim această platformă, Arduino este folosit în foarte multe proiecte. Partea software este ușor de folosit de către începători dar în același timp destul de avansată pentru cei ce vor să se aventureze în proiecte mult mai sofisticate. Rulează pe diferite platrorme cum ar fi Mac, Windows sau Linux. Studenții și chiar profesorii folosesc această platformă pentru a construi proiecte având un buget redus la dispoziție pentru a demonstra diferite principii sau de ce nu pentru a se aventura în lumea roboticii. Designerii și arhitecții construiesc prototipuri interactive, muzicienii și artiștii folosesc Arduino pentru a experimenta noi instrumente muzicale.

Arduino este un instrument cheie pentru a învăța lucruri noi. Oricine, copii, adulți, artiști sau programatori pot incepe un proiect Arduino urmând un set de instrucțiuni pas cu pas sau împărtășind idei cu alți utilizatori Arduino.

Există multe alte microcontrollere și platforme de microcontrollere disponibile pentru partea hardware a unui proiect. Parallax Basic Stamp, Netmediațs BX-24, Phidgets MITțs Handyboard, și multe altele oferă funcționalități similare. Toate acestea ofera un control asupra multor detalii pe care trebuie sa le luăm în calcul atunci când vrem să programăm un microcontroller. Arduino deasemenea simplifică lucrul cu un microcontroller, dar oferă în același timp avantaje pentru profesori, studenți și de asemenea pentru cei ce vor să se aventureze într-un proiect de acest gen.

În cele ce urmeaza voi oferi câteva avantaje concrete pe care platforma Arduino le oferă. În primul rând, această platformă nu este costisitoare, potrivită pentru cei ce vor să inceapă un proiect cu buget limitat. În comparație cu alte platforme de acest gen, Arduino este destul de acceptabil ca și preț.

Putem spune că Arduino este de tip multiplatformă(cross-platform) deoarece mediul de dezvoltare Arduino rulează pe diferite sisteme de operare cum ar fi Windows, Macintosh OSX și Linux. Majoritatea microcontrollerelor sunt limitate la un singur sistem de operare cum ar fi Windows.

Mediul de dezvoltare este simplu, clar și bine structurat, ușor de folosit pentru începători dar destul de flexibil pentru userii mai avansați care pot transforma un proiect simplu intr-unul complex. Partea software a proiectului Arduino este open source, putând fi extinsă de către programatorii experimentați. Limbajul poate fi extins prin librarii C++, iar cei ce vor să ințeleaga detaliile tehnice se pot aventura mult mai departe în fișierele AVR C, limbajul de programare pe care se bazeaza Arduino. Similar, putem adăuga cod AVR-C direct în proiectele noastre pentru a le extinde funcționalitațile. Partea hardware a proiectului, mai exact planurile pentru plăcile Arduino sunt publicate sub licența Creative Commons, deci cei ce vor sa iși facă propria versiune Arduino pot lua acele schite și să le extindă dupa bunul plac. Chiar și cei neexperimentați pot construi propriul modul pentru a ințelege principiile dupa care a fost făcut și a salva banii dați pe un modul gata făcut.

**1.1 Hardware**

Intelegerea arhitecturii hardware a platformei Arduino ne ajută să facem urmatorul pas în dezvoltarea proiectelor ce se bazeaza pe această platformă. O bună intelegere ne va ajuta să încorporăm un microcontroller în proiectarea unui produs final, și să omitem anumite aspecte de care nu avem nevoie în proiectul nostru.

Placa PCB Arduino folosește componente SMD (Surface Mount Device).Dispozitivele sau mai bine zis componentele electrice SMD se folosesc în electronică și reprezintă clasa componentelor montate direct pe suprafața plăcii cu cablaj imprimat folosindu-se micile lor suprafețe lipite cu cositor. Sunt cele mai răspândite componente electrice și datorită dimensiunolor lor mici fac să crească calitatea electrică a întregului circuit, mai ales la frecvențe mari și să se economisească mult spațiul necesar pentru plantarea componentelor pe placa de cablaj imprimat. Acest tip de componente nu prezintă pini de inserție în placă. Componentele SMD se lipesc direct, pe una din fețele plăcii cu cablaj imprimat (sau pe ambele) cu ajutorul cositorului.

Înainte să ințelegem partea hardware a platformei, trebuie să avem o viziune generală asupra sistemului care stă la baza funcționării acestei platforme. Dupa ce codul este compilat folosind Arduino IDE, trebuie sa îi facem upload pe microcontroller folosind o conexiune USB. Din cauza că microcontrollerul nu are un transceiver USB, avem nevoie de o “punte” pentru a converti semnalele între interfața serial (interfața UART) de microcontroller și semnalele gazdă USB. Puntea este reprezentată de Atmega 16U2, care are un transciever USB și o interfață serială (UART interface).



Fig.6: Componentele SMD montate pe o placă Arduino

Pentru a alimenta placa Arduino, putem folosi USB-ul ca și o sursă de curent. O alternativă ar fi să folosim DC jack. Pentru a face reset plăcii Arduino, trebuie să apasăm singurul buton care se află pe plăcuță. Altă variantă de a face reset este atunci cand deschidem interfața serială din Arduino IDE.

Este foarte important să ințelegem ca placa Arduino include un microcontroller și acest microcontroller este cel care executa instrucțiunile din program. Daca vom ști acest lucru, nu vom mai face greseala pe care multi utilizatori Arduino o fac: “Arduino este un microcontroller”.

Este important să ințelegem ca placa Arduino include un microcontroller, iar acest microcontroller este cel ce execută instrucțiunile dintr-un program. Daca vom ști asta, nu vom mai folosi vreodată fraza: “Arduino este un microcontroller”.

ATmega328 este un microcontroller utilizat împreună cu placa Arduino UNO R3 ca și controller principal. ATmega328 este un MCU din familia AVR. Este un dispozitiv de 8 biți, ceea ce înseamnă că arhitectura de date de autobus și registrele interne sunt proiectate să se ocupe de 8 semnale de date paralele.

ATmega328 are trei tipuri de memorie:

* **Memoria Flash**: 32kb memorie nevolatilă. Este folosită pentru stocarea aplicației, ceea ce explică de ce nu trebuie sa facem upload la cod de fiecare data cand deconectăm placa Arduino de la sursa de curent.
* **Memoria SRAM:** 2kb memorie volatile. Este folosită pentru a stoca variabilele folosite de aplicație in timp ce programul rulează.
* **Memoria EEPROM:** 1kb memorie nevolatilă. Este folosita pentru a stoca datele necesare atunci cînd placa ramane fara curent, și a relua aceste date atunci cand placa este conectată la o sursă de curent

|  |  |
| --- | --- |
| Parametri | Valori |
| Flash | 32 kbytes |
| RAM | 2kbytes |
| Cantitate Pini | 28 |
| Frecvență maximă de funcționare | 20 Mhz |
| CPU | 8-bit AVR |
| Numărul de variabile Canale | 16 |
| Pini maximi de E/S | 26 |
| Întreruperile externe | 24 |

Fig. 7: Detalii tehnice ATmega328p

MCU este un pachet DIP-28, ceea ce înseamnă ca are 28 pini dispuși în paralel. Acești pini includ pinii pentru I/O precum și cei de alimentare. Majoritatea pinilor sunt multifuncționali, ceea ce înseamnă că același pin poate fi folosit în diverse moduri în funcție de cum este configurat în software. Acest lucru reduce necesarul de pini, deoarece nu este necesar câte un pin separat pentru fiecare funcție. Poate de asemenea să facă designul mult mai ușor și flexibil deoarece un pin I/O poate oferi diferite tipuri de funcționalități.

De asemenea sunt disponibile și alte pachete ATmega328 cum ar fi TQFP-32 SMD (Surface Mount Device).

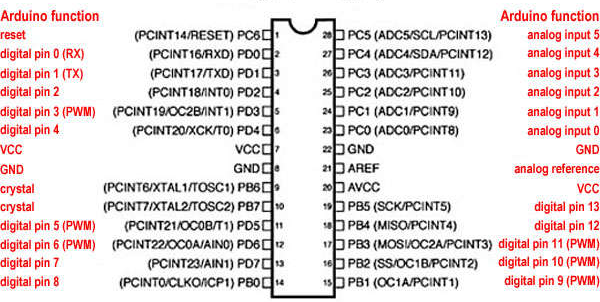
MCU suportă voltaje cuprinse între 1.8 și 5.5 V. Cu toate acestea, există restricții cu privire la frecvența de lucru, de exemplu, dacă doriți să utilizați frecvența maximă (20 MHz), aveți nevoie de o tensiune de alimentare de cel putin 4.5 V.

Există 3 porturi ce delimitează 3 familii de pini: PORTC, PORTB, PORTD. Toți pinii acestor porturi pot fi folosiți în scop general I/O sau pentru funcțiile alternative indicate în schema de mai jos.

Există de asemenea unii pini care sunt marcați ca fiind PWM. Acești pini sunt marcați cu “~” pe placa Arduino.

Atmega168 este aproape identic cu Atmega328, pinii fiind compatibili. Diferența dintre ele este că Atmega328 are mai multă memorie (32 kb flash, 1kb EEPROM și 2kb RAM) în comparație cu Atmega168 (16kb flash, 512 bytes EEPROM și 1kb RAM).

ATmega328 are șase pini analogi conectați la header-ul analog de pe placa Arduino. O greșeală comună este să ne gandim la pinii analogi ca fiind strict A/D. În realitate, ei pot fi folosiți și ca A/D dar și ca I/O. Mai exact, atunci când un pin este folosit ca fiind analog, mapeaza voltajele de la 0 la 5 V cu valori între 0 și 1023 ceea ce ne conduce la un rezultat de 4.9 mV per unitate. Comunicația serial realizată prin pinii RX/TX utilizează niveluri de logică TTL (5V sau 3.3V în funcție de placă). Nu trebuie să conectăm acești pini direct la un port serial RS232 deoarece acestea funcționează la o tensiune de +/- 12V și poate deteriora placa Arduino.



.

Fig.8: Descrierea pinilor de pe un ATmega328

Comunicarea Serială este utilizată pentru comunicația dintre placa Arduino și un calculator sau alte dispozitive. Putem de asemenea să folosim acest tip de comunicație pentru a crea o comunicație bidirecțională între două plăci Arduino. Toate plăcile Arduino au cel puțin un port serial (cunoscut de asemena ca UART sau USART). Acestea comunică prin pinii digitali 0(RX) și 1(TX), precum și cu un PC prin USB. Daca folosim pinii 0 și 1 pentru comunicarea serial nu ii putem folosi și pentru I/O. ATmega are un singur modul UART(Universal Asynchronous Receiver/Transmitter). Pinii RX/TX sunt conectați la un circuit convertor USB-UART și de asemena sunt conectați la pinii 0 și 1 în antetul digital.

Serial Peripheral Interface (SPI) este un protocol sincron de date seriale utilizat de microcontrollere pentru a comunica cu unul sau mai multe dispozitive periferice rapid pe distanțe scurte. Acesta poate fi, de asemenea, utilizat pentru comunicarea dintre două placi Arduino.

Cu o conexiune SPI există întotdeauna un dispozitiv master (de obicei un microcontroller) care controlează dispozitivele periferice. De obicei există trei linii comune tuturor dispozitivelor:

- MISO (Master in Slave out) – Linia Slave pentru trimiterea de date la master.

- MOSI (Master out Slave in) – Linia de master pentru a trimite date la periferice.

- SCK (Serial Clock) – Impulsurile de ceas care sincronizează transmisia de date generate de catre master.

- SS (Slave) -pinul de pe fiecare dispozitiv pe care master-ul îl poate utiliza pentru a activa și dezactiva anumite dispozitive.

Atunci când pinul de select al unui slave este low, el comunica cu masterul. Atunci cand este high, acesta ignoră masterul. Acest lucru ne permite să avem mai multe dispozitive SPI care împart aceiași pini MISO,MOSI și SCK.

Care este viteza maximă pe care o poate suporta dispozitivul SPI? Acest lucru este controlat de către primul parametru din SPI Settings. Dacă folosim un cip cotat la 15 MHz, atunci vom folosi 15000000. Arduino va folosi automat cea mai bună viteză, care este mai mică sau egală cu numărul folosit în SPI Settings.

Comunicarea de tip I2C este foarte populara și des folosită de către dispozitivele electrice deoarece poate fi implementată foarte ușor în designul circuitelor electrice care necesită o comunicare intre un master si multiple dispozitive slave sau de ce nu multiple dispozitive master. Implementarea ușoară se datorează faptului că avem nevoie de doar doua fire pentru comunicarea cu pană la 128 dispozitive când folosim 7 biți de adresare sau 1024 dispozitive când se utilizează 10 biți de adresare.

Cum este posibilă comunicarea dintre atâtea dispozitive cu doar doua fire? Fiecare dispozitiv are un id preselectat sau mai bine spus o adresă unică de dispozitiv astfel încît master-ul poate alege cu ce dispozitiv va comunica. Cele 2 fire sau linii sunt numite Serial Clock (SCL) și Serial Data (SDA). Linia SCL este semnalul ceasului care sincronizează transferul datelor intre dispozitivele de pe magistrala I2C și este generat de dispozitivul master. Cealaltă linie este linia SDA care transmite datele.

Semnalul de date este transferat în secvențe de 8 biți. Deci după o condiție specială de pornire apare prima secvență de 8 biți care ne indică adresa slave-ului care va primi datele. Fiecare secvență de 8 biți este urmată de un bit de confirmare. Bitul de confirmare este urmat de o alta secvență de adresare dar de data aceasta este adresată regiștrilor interni sau dispozitivelor slave. Dupa secvența de adresare urmeaza secvența de date, urmată și ea de o condiție specială de stop.

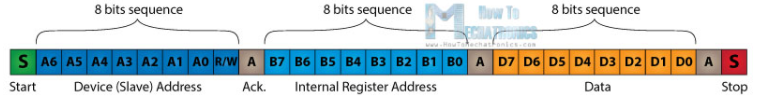


Fig. 9: Secvența de biți folosită de catre protocolul I2C

Întorcându-ne la partea electronică. Secțiunea microcontroller-ului este formată din urmatoarele componente:

* **ATmega328-PU:** Microcontroller-ul
* **IOL și IOH (Digital):** Aceste headere sunt headerele digitale pentru pinii de la 0 la 13 plus GND, AREF, SDA, SCL. Rx și TX de la puntea USB sunt conectați prin pinii 0 și 1.
* **AD:** Header-ul pentru pinii analogi.
* **Oscilator 16 Mhz:** Conectat cu XTAL2 și XTAL1 de MCU
* **Pin pentru reset:** Acesta este conectat la o rezistență de 10k pentru a ajuta la prevenirea resetărilor parazite în medii zgomotoase. Pinul reset are integrat un rezistor dar conform designului AVR și a schițelor de proiectare: “În cazul in care mediul este zgomotos, poate fi insufucienta iar resetul se poate declanșa foarte ușor”. Declanșarea are loc atunci cand utilizatorul apasă pe butonul de reset sau in cazul in care reset-ul este declanșat de la portul USB.
* **Pinul 13:** Este conectat la pinul SCK de la MCU și de asemenea este conectat la un LED. Placa Arduino folosește un buffer (LMV358) pentru a dirija ledul.
* **ICSP(In-Cirtuit Serial Programming):** Este folosit pentru a programa microcontroller-ul ATmega328p folosind un programator extern. Este conectat la interfața ISP (In System Programming) folosind pinii SPI.

În mod normal nu trebuie să folosim acest tip de programare a microcontroller-ului deoarece bootloader-ul manevrează programarea microcontroller-ului din interfața UART care este conectat prin punte la USB.

Rolul punții USB-UART este de a converti semnalele de interfață USB la interfața UART, pe care ATmega328 le înțelege folosind ATmega16U2 cu un emițator receptor USB intern. Acest lucru se face cu ajutorul firmware-ului special încarcat la ATmega16U2.

Din punctul de vedere al proiectarii electronice, această secțiune este similara cu secțiunea microcontroller-ului. MCU-ul are un header ICSP, un cristal extern cu condensatori (CL) și un condensator de filtrare Vcc.

Pentru sursa de alimentare avem opțiunea de a utiliza USB-ul sau mufa DC. Acum vine întrebarea: “Dacă le conectăm pe amândoua în același timp, care va fi sursa de alimentare?”. Regulatorul de 5V este NCP1117ST50T3G iar Vin al acestui regulator este conectat la portul jack DC prin intermediul diodei M7, versiunea SMD a faimoasei diode 1N4007. Această diodă oferă protecție atunci când polaritatea este inversată. Output-ul regulatorului de 5 V este conectat la tot circiutul care are nevoie de această tensiune, și de asemenea acționează ca și input pentru regulatorul de 3.3V LP2985-33DBVR. O altă sursă a regulatorului de 5V este USBVCC care este conectat la iesirea unui FDN340O, un MOSFET P-canal iar sursa este conectată la net 5V. Poarta tranzistorului este conectată la ieșirea unui op-AMP LMV358 folosit ca și comparator. Comparația se face între 3.3v și Vin/2. Cînd Vin/2 este mai mare, comparatorul va produce un output puternic iar P-channel MOSFET va fi oprit. Daca nu este aplicat nici un Vin, atunci comparatorul este initializat cu GND iar Vout este LOW, astfel încât tranzistorul este pornit iar USBVCC este conectat la 5V.

LP2985-33DBVR este regulatorul de 3.3V. Atat regulatorul de 3.3V cît și cel de 5V este de tip LDO(Low Dropout), cee ace înseamnă ca ele pot regla voltajul chiar dacă este apropiat de output, ceea ce poate fi vazut ca un pas înainte față de regulatoarele mai vechi cum ar fi 7805.

Cum am spus și mai sus, Vin de la portul jack DC este protejat de polaritatea inversată avînd o diodă M7 la input. În schimb, pinul de Vin din header nu este protejat. Acest lucru e din cauza ca este conectat dupa diode M7. O simpla rezolvare la acest lucru ar fi fost sa conecteze pinul înainte de diode pentru a oferi aceeași protecție.

Atunci cînd folosim USB-ul ca și sursă de current, pentru a oferi protecție portului USB, exita un PTC (positive temperature coefficient), mai exact o siguranță (MF-MSMF050-2) în serie cu USBVCC. Aceasta oferă protecție la supracurent, 500mA. Atunci cînd se atinge o limită supracurent, rezistența PTC crește foarte mult. Rezistența scade dupa ce supracurentul este eliminat.

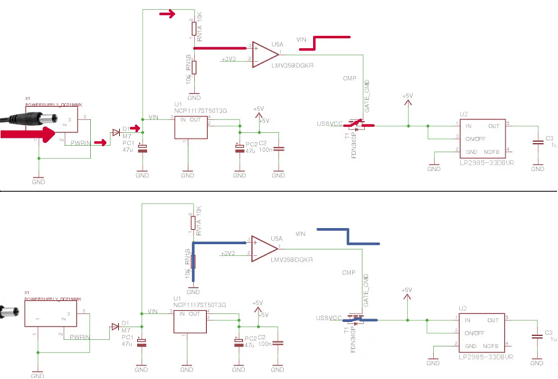
****

Fig.10: Mecanismul de inversare a polaritații

**1.2 Cinci mituri despre Arduino**

În cele ce urmează voi clarifica așa-zisele mituri care și-au facut simțită prezența în jurul acestei platforme, mituri care sunt în totalitate false dar sunt prezentate ca fiind adevărate chiar și la unele prezentări ce au ca scop studiul acestei platforme.

Primul mit, și cel mai controversat dupa părerea mea este: “Arduino folosește propriul limbaj de programare”. În mare parte este adevarat, un fisier Arduino arata în mare parte ca fiind unic din punct de vedere structural, dar de fapt el este cod C++ cu puțină preprocesare. Atunci când realizăm partea software a unui proiect scriem cod C++, care se bazează pe libraria Arduino.h. Funcțiile ca setup(), loop() sau digitalWrite() nu sunt altceva decât funcții C++.

Al doilea mit este acela că pinul 13 al unei plăci Arduino are încorporat o rezistenta. Acest zvon cum că pinul 13 al unei plăci Arduino are o rezistență integrată a fost fondat datorită ledului încorporat în placuță. care este legat la pinul 13 al microcontroller-ului. Într-adevar, există o rezistentă dar nu este folosită pentru semnalul de output generat de către pinul 13 ci este folosită de LED. Singura dată când a fost pusă o rezistență pe acest pin a fost la primele versiuni de Arduino, fiind produse în jur de 200 de exemplare.

Al treilea mit este legat de spațiul lăsat intre pinul 7 si 8. Atunci când luam prima dată în mană o placuța Arduino Uno, este imposibil să nu observăm acel spațiu care a fost lăsat între două barete de pini, mai exact între pinii 7 și 8. La un moment dat, pe situl web Arduino a aparut această întrebare iar raspunsul a fost unul intuit de unii utilizatori cum că acel spațiu a fost o eroare de proiectare. Apoi cei de la Arduino au dezvăluit că acest spațiu a fost pus intenționat pentru ca un shield să fie așezat pe partea corecta. Citind mai multe păreri, această decizie de a lăsa acel spațiu nu a facut decat să facă atașarea unui shield mult mai grea.

**1.3 Mediul de dezvoltare și tehnologiile utilizate**

Proiectul a fost dezvoltat folosind mediile de dezvoltare Visual Studio, Arduino Ide și pentru codul Java a fost folosit Eclipse. Ca și limbaje de programare folosite predomină C++, fiind codul pentru microcontrollere, dar a fost folosit si Java pentru realizarea interfeței care ajută la trimiterea comezilor de la PC și realizează interacțiunea cu utilizatorul. De asemenea proiectul este realizat în totalitate conform principiilor OOP.

Interfața aplicației desktop a fost realizată folosind familia de pachete de clasă cunoscută sub numele Swing. Această librarie a fost creată pentru a se realiza interfețe grafice mai bogate și mai diversificate decât cele posibile în AWT. Din această familie fac parte pachetele javax.swing, javax.swing.event și altele. Remarcăm că Swing nu înlocuiește, ci completează AWT cu noi pachete de clase. Multe din clasele AWT sunt necesare și atunci când se lucrează în Swing. Obiectele grafice din swing nu au numai aceeași funcționalitate, ci și același aspect pe orice platformă. Mai mult, acest aspect poate fi ales de către programator dintre mai multe variante posibile. Pentru a se realiza acest obiectiv, autorii Swing au renunțat la ideea de a folosi pentru prezentările grafice metodele native de pe platformele pe care se execută aplicația. Toate obiectele grafice din Swing sunt reprezentate pe ecran și manupulate prin metode scrise în limbajul Java. Aceasta conferă interfeței frafice independență totală față de platformă și posibilitatea diversificării claselor de obiecte folosite, dar prețul plătit este creșterea complexității programelor și a timpului de calcul.

Componentele Swing sunt aproape toate derivate dintr-o singură clasă de bază numită Jcomponnt, care moștenește la rândul ei clasa Container din AWT. Moștenirea clasei Jcomponent dă componentelor posibilitatea de a acea bordere, tooltips, respectiv look-and-feel configurabil. De asemenea tot din clasa Jcomponent sunt moștenite și metodele prin care se pot seta dimensiunile și poziționarea componentelor. Swing îmbogățește considerabil paleta componentelor disponibile în AWT. De asemenea, toate componentele grafice Swing sunt construite conform specificației JavaBeans, ceea ce înseamnă că sunt de fapt componente JavaBeans. Compatibilitatea dă posibilitatea construirii de medii vizuale pentru Java îm care interfața grafică pentru aplicații se construiește interactiv prin așezarea componentelor grafice pe suprafața de lucru. Pentru a folosi componentele Swing trebuie să importăm mai întâi pachetele corespunzătoare.

**1.4 Alte aplicații ale platformei Arduino**

Platforma Arduino are o multitudine de întrebuințări atunci când vrem să realizăm un proiect ce implică lucrul cu parți electronice. Multitudinea de proiecte în care poate fi integrat Arduino se datorează faptului că această platforma este puternic modularizată. Prin modularizare mă refer la multitudinea de module pe care putem să le realizăm, sau să le achiziționăm.

Dea lungul timpului au fost realizate numeroase proiecte de acest gen, în care sunt folosiți niste senzori pentru a realiza o interacțiune cu mediul iar pe baza acelor rezultate să se aplice o acțiune/strategie. Deoarece multă lume a abordat acest tip de proiect, producătorii de componente electronice au pus la dispoziția utilizatorilor niște schelete de roboți pe care putem să îi folosim special la acest tip de proiecte.

Mulți utilizatori abordează acest gen de proiect folosind pentru interacțiunea cu mediul un sonar realizat cu un singur sensor de ultrasunete și un servomotor. Am încercat și eu această abordare dar s-a dovedit ineficientă deoarece citirea datelor durează prea mult din cauza mișcărilor pe care trebuie să le facă servomotorul, ceea ce ar diminua semnificativ viteza cu care se deplasează mașina.

Un alt proiect care mi s-a părut interesant și pe care îl voi aborda intr-un viitor apropiat este un cub 8x8x8 care poate fi asemănat cu un ecran LED. Ce este special la acest cub este că spre deosebire de un ecran LCD care este 2d, Acest cub poate reda o imagine de tip 3D. La un ecran normal de tip 3D trebuie să avem grijă cum folosim acei pixeli pentru a reda o imagine cât mai clară pe când la acest cub ne putem concentra pe felul cum este căzută imaginea în spațiu, fiind realizată o reprezentare clară a obiectului. Pentru fabricarea acestui cum avem nevoie de 512 leduri ceea ce face acest proiect unul de o complexitate destul de ridicată. Cu toate acestea, chiar dacă este un proiect foarte complex, acesta ne ajută să întelegem multe concept hardware și software deci este un proiect educational foarte bun.

**Capitolul 2**

**Prezentarea generală a proiectului**

Proiectul “Smart Car” are ca scop realizarea unui robot care se folosește de un set de senzori pentru a realiza o interacțiune precisă cu mediul pe care îl are la dispoziție. Datele de la senzori ne oferă o viziune clară despre eventualele obstacole care ar putea fi prezente în imprejurimi, evitarea obstacolelor fiind principalul obiectiv al robotului. De asemenea pe baza datelor primite de la senzori putem realiza o cartografiere a mediului înconjurator. Atunci când este întâlnit un obstacol, robotul aplica o strategie pentru a evita acel obstacol. Această strategie generează la rândul ei datele necesare cartografierii mediului și le transmite mai departe către un dispozitiv de stocare, datele de pe acel dispozitiv fiind reinterpretate mai apoi de către un PC.

Controlul robotului este realizat de către o aplicație existentă pe un PC. Această aplicație transmite semnalele către robot prin intermediu unui dispozitiv de tip radio care preia la rândul sau comenzile de la aplicație prin intermediul unui port USB. Există două stări principale pe care robotul le poate avea după ce acesta a fost pornit.

Modul manual, fiind starea inițială a robotului este starea în care robotul este ghidat în totalitate de catre un utilizator prin intermediul aplicației de pe PC. Interacțiunea cu aplicația se realizează prin intermediul dispozitivelor periferice, cum ar fi tastatura sau mouse-ul. Prima sarcină pe care o avem de facut atunci cînd vrem sa începem o comunicare cu robotul este realizarea conexiunii între PC și plăcuța Arduino care are atașat un dispozitiv radio de tip transmitter. După ce placuța a fost atașată la laptop prin portul USB, aceasta va trebui recunoscută de aplicație și totodata va trebui să realizăm conexiunea software care se realizează la nivel de aplicație. După ce am facut aceste lucruri trebuie să ne asigurăm că robotul este alimentat la o sursă de curent și că este plasat într-un mediu în care se poate deplasa. Acum nu mai rămâne decât să folosim aplicația și să transmitem comenzi robotului.

Al doilea mod în care se poate afla robotul este modul automat. Pentru a putea fi setat acest mod, trebuie să ne asiguram ca am trecut prin toate etapele de mai sus și că robotul se află în starea inițială și anume modul manual. Modul automat este setat cu ajutorul aplicației prin apăsarea butonului “A” de pe tastatura. După ce a fost transmisă această comandă, robotul va ști că trebuie să adopte altă strategie, dupa care vor fi activați senzorii care îl ajută să realizeze interacțiunea cu mediul în care se află. Atunci când se află în modul automat, robotul se va deplasa singur pe suprafața în care se află evitând eventualele obstacole pe care le întâlnește în calea sa. De asemenea, atunci cînd robotul se află în modul automat acesta va interpreta datele primite de la senzori și va stoca un anumit traseu pe ca l-a realizat atunci când a evitat obstacolele, ceea ce ne va ajuta să realizăm o eventuală cartografiere a mediului în care s-a deplasat. Aceste date vor fi stocate într-un card de memorie pentru a face posibil transferul datelor la aplicația de pe PC.

Bazele acestui proiect pot fi transpuse într-o aplicație reală benefică în viața de zi cu zi. Așa cum am menționat și în introducerea acestui proiect, o mașină autonomă funcționează după aceleași principii, și anume, interpretează niște date primite de la receptori care în cazul nostru sunt niște senzori cu ultrasunete. Într-un proiect real, cum ar fi o mașină autonomă, principiile de bază sunt în mare parte aceleași doar ca la nivel hardware, componentele sunt mult mai sofisticate, interacțiunea cu mediul înconjurator fiind realizată într-o manieră mult mai detaliată și precisă. Este important să primim niște date cît mai precise si detaliate pentru a evita anumite erori care pot surveni pe baza unor date care pot pot fi interpretate de către program într-un mod greșit, ceea ce ar duce la o catastrofă în unele cazuri.

**2.1 Interacțiunea cu mediul**

Revenind la proiectul nostru, componenta hardware care face posibilă interacțiunea cu mediul este senzorul cu ultrasunete HC-SR04. Acest senzor folosește un sonar pentru a determina distanța pâna la un obiect, asa cum fac și liliecii. Oferă un rezultat precis, cu un mod foarte stabil de citirea a datelor și de asemenea este foarte ușor de utilizat, oferind o rază de acțiune cuprinsă între 2cm și 400cm.

Transmitter-ul de pe acest modul ultrasonic emite o undă într-o anumită direcție și în același timp este pornit un cronometru pentru a vedea cât timp îi ia acelei unde să se întoarcă la emițător.Undele ultrasonice se transmit prin aer și se vor întoarce imediat ce au întâlnit un obstacol în calea sa. În cele din urma, cronometrul se va opri atunci când unda se va întoarce la receptor. Viteza cu care unda se raspândește prin aer este de 340 m/s, iar pe baza informațiilor de la cronometru putem calcula distanța între obstacol și transmitter, cu ajutorul formulei L = C x T. În această formulă, L este distanța măsurată, C este reprezentat de raspândirea undei ultrasonice în aer iar T reprezintă timpul necesar pentru ca acea undă să se întoarcă la receptor.

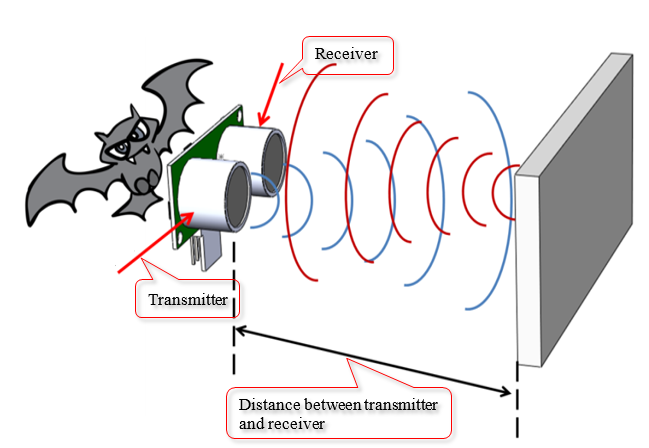


Fig.11: Modul de funcționare al unui senzor cu ultrasunete

Acest tip de senzor emite ultrasunete la o frecvență de 40 000 Hz care este transmisă prin aer, iar în cazul în care întâlnește un obstacol în calea sa, semnalul se va întoarce la senzor. Considerând ca semnalul traversează aerul cu viteza sunetului putem calcula distanța de la senzor la obstacol.

Modulul HC-SR04 are 4 pini, Ground, Vcc, Trig și Echo. Pinii ground și Vcc ai modulului trebuie sa fie conectați la pinii de Ground și 5V prezenți pe placa Arduino, respectiv Echo și trig la oricare pini digitali I/O.

Pentru a genera ultrasunete, trebuie să setam pinul Trig ca fiind activ pentru o perioada de 10 µs. Acest semnal va trimite 8 cicluri de unde care vor traversa aerul cu o viteză supersonică și vor fi receptate mai apoi de pinul echo. Pinul echo va genera ca și output timpul în microsecunde care a fost necesar pentru traversarea undei de la modul la obstacol.

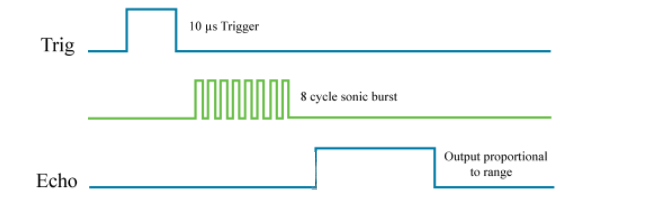


Fig. 12: Semnalul transims/receptionat de catre un modul HC-SR04

De exemplu, daca un obiect este la 10 cm distanță de senzor, iar viteza sunetului este 340 m/s sau 0.034 cm/microsecundă, unda va traversa distanța în 294 microsecunde. Atunci când se va întoarce la modul, acea valoare va fi dublată deoarece unda va trebui să parcurgă de doua ori distanța pentru a ajunge înapoi la modul. Deci în loc sa află, distanța în centimetri, va trebui să multiplicăm timpul recepționat de la echo cu 0.034 si să îl împărțim la 2.

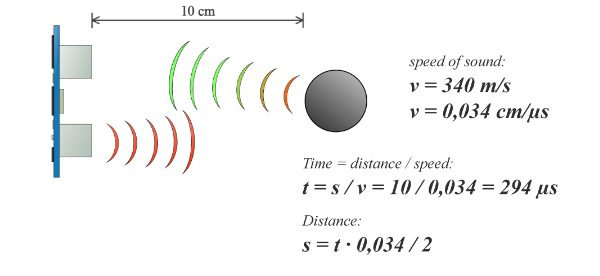


Fig. 13: Calcularea distanței de la modul la obstacol

**2.2 Mișcarea**

Pentru mișcarea robotului sunt folosite 4 motoare electrice care lucrează împreună pentru a realiza mișcări cât mai precise. Având la fiecare roată câte un motor, tracțiuna 4x4

este utilă atunci când roborul întâlnește un mediu mai ostil.

La nivel software există patru comenzi pe care motoarele trebuie sa le îndeplinească, iar în funcție de comanda primită, un motor sau un grup de motoare se mișcă la o anumită viteză. Prima comandă, „goForward” activează toate cele 4 motoare, ajutând robotul să se deplaseze înainte. La această comandă, participă toate cele 4 motoare existente. A doua comandă, “goBackward” îi spune mașinii să se deplaseze în sens invers față de prima comandă. De asemenea pentru îndeplinirea acestei sarcini sunt puse la dispoziție cele 4 motoare. A treia comandă, “goLeft” realizează un viraj la stânga și pune în sarcină doar două motoare, fiind active motoarele din partea opusă în care vrem să virăm. A patra comandă, “goRight”, realizează un viraj la dreapta și la fel ca la comanda precedentă, sunt active motoarele din partea opusă virajului.

Pentru punerea în funcțiune a motoarelor a fost folosit un semnal PWM care ne-a ajutat să scadem sau să marim forța motoarelor atunci când este nevoie. De exemplu atunci când realizăm un viraj este nevoie de toata forța în condițiile în care avem active doar doua motoare din cele 4. Atunci când vrem sa ne mișcăm înainte sau înapoi, putem sa setăm una din cele 6 viteze care sunt simulate cu ajutorul semnalului PWM. O viteză redusă a fost necesară si pentru procesarea datelor de la senzori. Întrucât există o întârziere atunci când sunt procesate datele primite de la senzori, trebuie să scadem viteza cu care se deplasează robotul pentru a avea timp sa îsi dea seama ca acolo există un obstacol și să îl evite.

Un motor electric (sau electromotor) este un dispozitiv electromecanic ce transformă energia electrică în energie mecanică. Transformarea în sens invers a energiei mecanice în energie electrică este realizată cu un generator electric. Nu există diferențe de principiu semnificative între ceke doua tipuri de mașini electrice, același dispozitiv putând îndeplini ambele roluri în situații diferite.

Majoritatea motoarelor electrice funcționează pe baza forțelor electromagnetice ce acționează asupra unui conductor parcurs de curent electric aflat în câmp magnetic. Există însă și motoare electrostatice construite pe baza forăei Coulomb și motoare piezoelectrice.

Fiind construite într-o gamă extinsă de puteri, motoarele electrice sunt folosite la foarte multe aplicații: de la motoare pentru componente electronice până la acționări electrice de puteri foarte mari (pompe, locomotive).

Indiferent de tipul motorului, acesta este construit din două părți componente: stator și rotor. Statorul este partea fixă a motorului, în general exterioară, ce include carcasa, bornele de alimentare, armătura feromagnetică statorică și înfașurarea statorică. Rotorul este partea mobilă a motorului, plasată de obicei în interior. Este format dintr-un ax și o armătura rotorică ce susține înfașurarea rotorică. Între stator și rotor există o porțiune de aer numită întrefier ce permite mișcarea rotorului față de stator. Grosimea întrefierului este un indicator important al performanței motorului.

Motorul de curent continuu a fost inventat în 1873 de Zénobe Gramme prin conectarea unui generator de curent continuu la un generator asemănător. Astfel, a putut observa că mașina se rotește realizând conversia energiei electrice absorbite de la generator. Astfel el a constatat, că generatorul “inițial” era de fapt o mașină electrică reversibilă, care putea lucre ca un convertizor de energie bidirecționali. Motorul de curent continuu are pe stator polii magnetici și bobinele polare concentrate care creează câmpul magnetic de excitație să excite în permanență o forță față de rotor.

În funcție de modul de conectare a înfașurarii d excitație motoarele de curent continuu pot fi clasificate în:

* Motor de excitație indpendentă – unde înfașurarea statorică și înfașurarea rotorică sunt conectate la doua surse separate de tensiune.
* Motor cu excitație paralelă – unde înfașurarea statorică și înfașurarea rotorică sunt legate în paralel la aceeași sursă de tensiune
* Motor cu excitație serie – unde înfășurarea statorică și înfășurarea rotorică sunt legate în serie.
* Motor cu excitație mixtă – unde înfășurarea statorică este divizată în doua înfășurari, una conectată în paralel și una conectată în serie.

Înfașurarea rotorică parcursă de curent va avea una sau mai multe perechi de poli magnetici echivalenși până când polii rotorici se aliniază în dreptul polilor statorici opuși. În același moment, colectorul schimbă sensul curenților rotorici astfel încât polaritatea rotorului se inversează și rotorul va confinua deplasarea până la următoarea aliniere a polilor magnetici.

Pentru acționări electrice de puteri mici și medii sau pentru acționări ce nu necesită camp magnetic de excitație variabil, în locul înfășurarii statorice se folosesc magneți permanenți.

Turația motorului este proporțională cu tensiunea aplicată înfășurarii rotorice și invers proporțională cu câmpul magnetic de excitație, Turația se reglează prin varierea tensiunii aplicată motorului până la valoarea nominalp a tensiunii, iar turații mai mari se obțin prin slăbirea câmpului de excitație. Ambele metode vizează o tensiune variabilă ce poate fi obținută folosind un generator de curent continuu prin înserierea unor rezistoare în circuit sau cu ajutorul electronicii de putere.

Cuplul dezvoltat de motor este direct proportional cu curentul electric prin motor și câmpul magnetic de excitație. Reglarea turației prin slabire de camp se face, așadar, cu diminuare a cuplului dezvoltat de motor. La motoare de serie același curent străbate înfășurarea de excitație și înfășurarea rotorică.

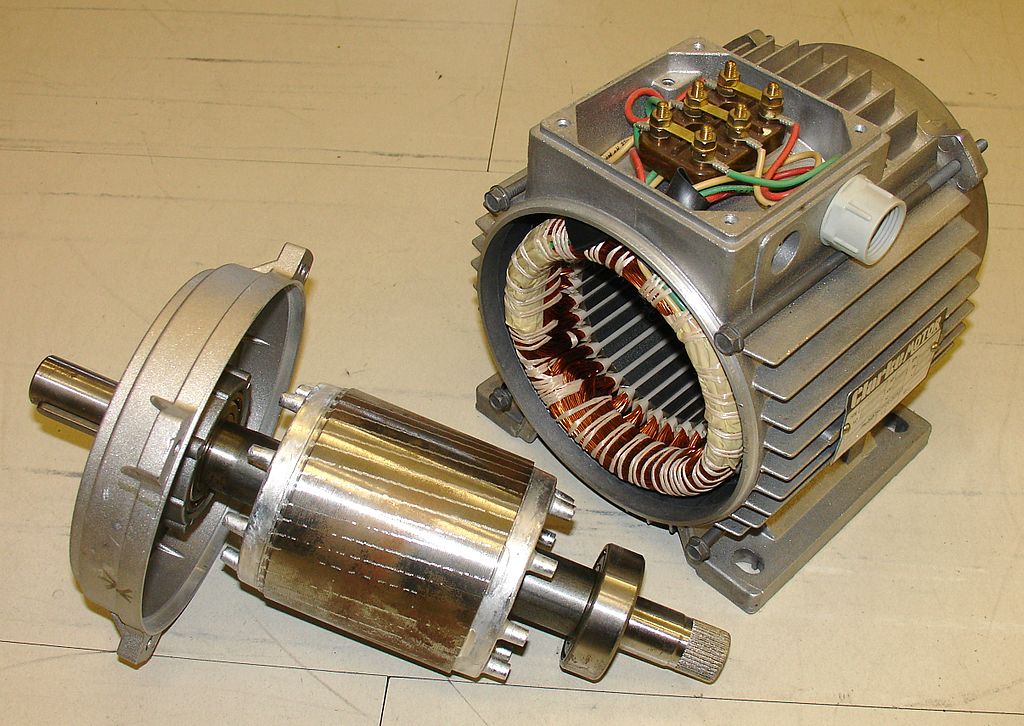


Fig. 14: Rotorul motorului electric(stânga) și statorul(dreapta)

Din această considerație se pot deduce două caracteristici ale motoarelor de serie: pentru încărcări reduse ale motorului, cuplul acestuia depinde de pătratul curentului electric absorbit; motorul nu trebuie lăsat să funcționeze în gol pentru că îm acest caz valoarea intensității curentului electric absorit este foarte redusă și implicit câmpul de excutație este redus, ceea ce duce la ambalarea mașinii până la autodistrugere. Motoarele de curent continuu cu excitație serie se folosesc în tracțiunea electrică urbană și feroviară (tramvaie, locomotive). Schimbarea sensului de rotație se face fie prin schimbarea polaritații tensiunii de alimentare, fie prin schimbarea sensului câmpului magnetic de excitație. La motorul serie, prin schimbarea polarității tensiunii de alimentare se realizează schimbarea sensului ambelor mărimi și sensul de rotație rămâne neschimbat. Așadar, motorul serie poate fi folosit și la tensiune alternativă, unde polaritatea tnsiunii se inversează o data în decursul unei perioade. Un astfel de motor se numește motor universal și se folosește în aplicații casnice de puteri mici și viteze mari de rotație (aspirator, mixer).

Atunci când înfășurarea rotorului este alimentată, în jurul lui se generează un câmp magnetic (poziție relativă a polilor magnetici, de la stânga la dreapta: N-NS-S). Polul nord al rotorului e respins de polul nord al statorului spre dreapta și e atras de polul sud al statorului(din dreapta), producând un cuplu mecanic motor care întreține mișcarea de rotație. Când rotorul ajunge în poziție orizontală, corectorul electric de comutare al sensului curentului continuu prin înfășurarea rotorului, inversând polii câmpului magnetic produs de rotor, se ajunge astfel la poziția relativă a polilor magnetici (N-NS-S) și procesul continuă.

Pentru o mai bună tracțiune și implicit o forță mai mare s-a folosit la fiecare motor câte un reductor. Un reductor are funcția de a modifica parametrii caracteristici puterii mecanice emise de un motor electric – cuplul și turația. Reductoarele reduc, de regulă, rotația arborelui motor, însă în același timp multiplică momentul și cuplul acestuia.

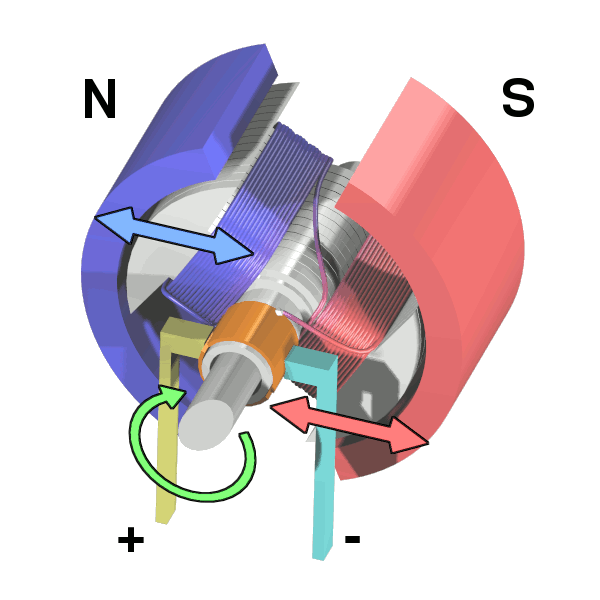


Fig. 15: Reprezentare de principiu al motorului.

Cuplate direct pe motoarele electrice prin flanșe specifice, reductoarele pot fi încadrate în categorii diferite, în funcție de angrenajul lor. Existî astfel reductoare cu angrenaje cilindrice, care oferă una sau mai multe trepte de reducere, după necesarul dorit. Randamentul total al unui astfel de reductor este dat de însumarea randamentelor individuale ale fiecărei trepte.

O a doua categorie de reductoare o reprezintp gama de reductoare melcate. Aceste reductoare își găsesc utilizare în aplicații din cele mai diverse, se pretează la instalații universale. Oriunde sunt necesare rapoarte și cupluri mari, de obicei în spații restrânse, un reductor melcat se dovedește a fi o adaptare extrem de utilă la motorul electric. Principalele caracteristici ale reductoarelor melcate snt puterile variate – situate între 0/09 și 7.5 kW, sau rapoartele de reducie ce pot ajunge până la reducerea turației la cota 1Ș100 a angrenajului electric. Forma reductoarelor melcate este de obicei una vubică, permițând o fixare universală, zgomotul produs de acestea în timpul funcționării este unul redus, ceea ce le transformă în varianta ideală de personalizare în funcție de necesitațile turației motorului electric. Angrenajele conice sunt folosite de asemenea la construcția reductoarelor, oferind la rândul lor un randament bun în utilizre. Standardele de putere nominală și de turație ale acestora nu sunt în majoritatea cazurilor conforme cu ceea ce se cere la o anumită aplicație.

Așa cum am specificat și mai sus unde am facut o amplă descriere a comenzilor pe care roborul le îndeplinește, aceste trebuie să fie capabil să meargă în ambele direcții cu ajutorul motoarelor de tracțiune. Acest lucru se realizeaza cu o punte h.



Fig. 16: Motor electric cu reductor

O punte h este un circuit electronic ce permite aplicarea unei tensiuni pe osarcină în orice sens. Aceste circuite sunt adesea folosite în robotică și alte aplicații pentru a permite motoarelor de curent continuu să ruleze înainte și înapoi. Punțile H sunt disponibile ca circuite integrate sau pot fi construite din componente discretem tranzistoare bipolare sau MOS. Puntea H are numele derivat de la modul obișnuit de desenare a circuitului. Aceasta este singura cale de tip solid state de a comanda motorul în ambele direcții.

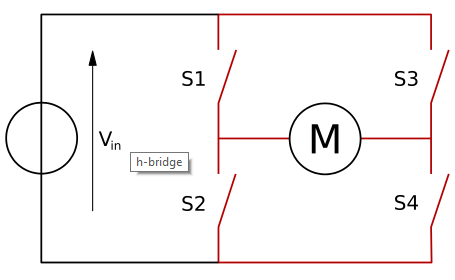


Fig. 17: Schema punții H

Atunci când întrerupătoarele S1 și S4 din figura de mai sus sunt închise și S2 și S3 sunt deschise, o tensiune pozitivă va fi aplicată la nivelul motorului. Prin deschiderea întrerupătoarelor S1 și S4 și închiderea întrerupătoarelor S2 și S3, această tensiune este inversată, astfel încât să permită funcționarea inversă a motorului. Întrerupătoarele S1 și S2 nu trebuie să fie închise în același timp, deoarece acest lucru ar provoca un scur-circuit la sursa de tensiune (Vin). Același lucru se aplică și întrerupătoarelor S3 și S4.Întrerupătoarele S1,S2,S3,S4 sunt tranzistoare bipolare MOS-FET.

O punte H este de obicei folosită pentru a schimba polaritatea sau mai bine spus direcția unui motor electric dar poate fi folosită pentru a implementa o frână pentru acel motor. În tabelul de mai jos vom vedea cum este posibil acest lucru.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S1 | S2 | S3 | S4 | Rezultat |
| 1 | 0 | 0 | 1 | Motorul se mișcă la dreapta |
| 0 | 1 | 1 | 0 | Motorul se mișcă la stânga |
| 0 | 0 | 0 | 0 | Motorul stă pe loc |
| 1 | 0 | 0 | 0 | Motorul stă pe loc |
| 0 | 1 | 0 | 0 | Motorul stă pe loc |
| 0 | 0 | 1 | 0 | Motorul stă pe loc |
| 0 | 0 | 0 | 1 | Motorul stă pe loc |
| 0 | 1 | 0 | 1 | Motorul frânează |
| 1 | 0 | 1 | 0 | Motorul frânează |
| 1 | 1 | 0 | 0 | Scurt circuit |
| 0 | 0 | 1 | 1 | Scurt circuit |
| 0 | 1 | 1 | 1 | Scurt circuit |
| 1 | 0 | 1 | 1 | Scurt circuit |
| 1 | 1 | 0 | 1 | Scurt circuit |
| 1 | 1 | 1 | 0 | Scurt circuit |
| 1 | 1 | 1 | 1 | Scurt circuit |

Fig. 18: Combinațiile de input posibile pentru o punte H și rezultatul acestora

O alternativă ar fi să avem un releu DPDT(Double Pole Double Throw) pentru a seta direcția curentului și un tranzistor pentru a activa “scurgerea” curentului. Această abordare poate extinde viața releului, releul fiind închis cât timp tranzistorul este închis și nu trece nici un curent prin el. De asemenea putem folosi semnalul PWM controlând fluxul curentului.

**2.3 Comunicația**

În acest proiect au fost folosite mai multe tipuri de comunicație pentru o sincronizare cât mai bună atât între componentele hardware ale proiectului cât și între PC și robot.

Primul tip de comunicație este comunicația serială între PC și placa Arduino care se ocupă cu trimiterea datelor spre robot. Pentru a ne asigura ca această conexiune este activă, trebuie mai întâi sa ne conectăm cablul USB între PC și placa Arduino apoi să setam din aplicație portul la care ne conectăm. Această comunicație este bidirecțională există un flux de date trimis din ambele părți. Aplicația desktop trimite domenzi către placa Arduino dar în același timp se poate trimite o comandă care necesită un raspuns din partea placii (informațiile despre vreme).

De asemenea, comunicația serială este întâlnită și între cele 2 microcontrollere care se ocupă de funcționarea robotului. Primul microcontroller preia datele de la Laptop printr-un dispozitiv de tip receiver. Dupa ce a captat datele, acestea sunt filtrate și trimise mai departe la cel de-al doilea microcontroller care se ocupă de controlul și sincronizarea perifericelor. Comunicația se realizează cu ajutorul pinilor RX/TX adică pinii 0 și 1. Pinul Rx de pe primul cip este conectat la pinul Tx de pe al doilea cip, iar pinul Tx de pe primul cip este conectat la pinul Rx de pe al doilea cip. Între cele doua cipuri este o comunicare unidirecțională, comunicația fiind una de tip master/slave, masterul fiind primul cip care dă ordinele iar slave-ul fiind al doilea cip care execută instrucțiunile primite.

Comunicația dintre placa Arduino conectată la PC și robot se realizează prin niște module de tip Transmitter/Receiver.

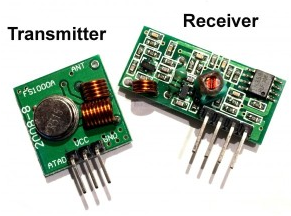


Fig. 19: Transmitter/Receiver

Dacă vrem să realizăm o comunicație între două placi Arduino, aceasta este una din cele mai bune soluții existente deoarece aceste module sunt foarte ieftine iar raza lor de acțiune este foarte mare ajungând la peste 100m(daca alimentăm Transmitter-ul la 12V). Datele pot fi transferate la o viteză de 4800 bps. Modul de conectare la o placă Arduino este foarte simplu, nu trebuie decât să conectăm pinii VCC, Gnd și un pin I/O de pe placa Arduino. Un dezavantaj la aceste module este acela că datele sunt transmise doar într-un singur sens, de la Transmitter la Receiver, nu și invers. Există diferite metode de a creste raza de acțiune a acestor module cum ar fi o antenă mai lungă sau alimentarea receiver-ului la tensiunea maximă suportată de 12V.

**2.4 Concepte și echipamente**

Pentru realizarea acestui proiect au fost folosite numeroase echipamente hardware. Principalele echipamente făradecare acest proiect nu putea fi realizat sunt senzorii, platforma cu cele patru motoare și evident placile Arduino. Interacțiunea cu mediul a fost realizată prin cei patru senzori, trei în față dispuși la un unghi de 45 de grade și unul în spate. Platforma robotului a venit însoțită de 4 motoare cu redauctor ce asigura o tracțiune integrală cu o aderență și flexibilitate destul de bună datorită reductoarelor prezente la nivelul fiecarui motor. Echipamentele adiționale de care nu am vorbit până acum sunt LED-urile, rezistențele, fotorezistorul și regulatorul de tensiune L7805 care ne ajută să scadem voltajul la un nivel acceptabil (5V) pentru o bună funcționare a plăcilor Arduino.

Un LED (Light Emitting Diode – diodă emițătoare de lumină) este o diodă semiconductoare ce emite lumină la polarizarea direct a joncțiunii p-n. Acest effect este o formă de electroluminescență. Este o sursă de lumină mica, de cele mai multe ori însoțită de un circuit electric ce permite modularea formei radiației luminoase. De cele mai multe ori acestea sunt utilizate ca indicatori în cadrul dispozitivelor electronicem dar din ce mai mult au început să fie utilizate ca și sursă de iluminare. Culoarea luminii depinde de compoziția și de starea materialului semiconductor folosit.

Electroluminescența a fost descoperită în anul 1907 de către H. J. Round, folosind un cristal de carbură de siliciu și un detector primitive dintr-un metal semiconductor. Rusul Oleg Vladimirovich a fost primul care a creat primul LED in anul 1920. Cercetarea sa a făcut înconjurul lumii, însă nu s-a găsit nici o întrebuințare a acesteia pentru câteva decenii.

Primul LED cu emisie în spectrul vizibil (roșu) a fost realizat în anul 1962 de către Nick Holonyak, când lucre la General Electric Company. Un fost student al acestuia a inventat primul LED de culoare galbenă și a îmbunătățit factorul de iluminare al LED-urilor roșu și roșu-portocaliu de circa zece ori în anul 1972.

Dea lungul anilor, LED-urile și-au facut simțită prezeța tot mai mult în viața noastră de zi cu zi, fiind prezente în multe dispozitive pe care le întâlnim. Led-urile au suferit de-a lungul anilor numeroase schimbari. In anul 2010 firma Nexxus Lighting a prezentat cea mai putenică lampă LED de uz casnic disponibilă pe piață, cu o eficiență de 50 Lumen/Watt. Recent, în anul 2013, firma Cree a prezentat Lampa Led de uz casnic cu o eficiență de 84 Lumen/Watt, consum 9-9.5 Watt, având prețul sub 10$ în variant de 40W.

În acest priect LED-urile au fost folosite pentru a semnala că un anumit dispozitiv este conectat la o sursă de curent sau dacă un anumit modul emite anumite semnale. Atunci când folosim un led și avem o sursă de curent de un voltaj mai mare trebuie sa folosim și o rezistență (rezistor) conectată la alimentarea LED-ului.

Rezistorul este o piesă component din circuitele electrice și electronice a cărei principală proprietate este rezistența electrică. Rezistorul obișnuit are doua terminale conform legii lui ohm, curentul electric care curge prin rezistor este proportional cu tensiunea aplicată pe terminalele rezistorului (I=U/R). Cel mai important parametru al unui rezistor este rezistența electrică exprimată în ohmi. Rezistoarele sunt complet caracterizate prin relația între tensiunea la borne și intensitatea curentului electric prin element atunci când dependența U=f(I) este liniară. Rezistoarele se pot clasifica după mai multe criterii. Dupa materialul folosit, se realizeazăȘ rezistoare din metale sau aliaje metalice(fire sau benzi), rezistoare peliculare cu carbon(pelicule depuse pe un support izolat)ș rezistoare cu peliculă de metal-oxid, rezistoare cu lichid bazate pe rezistența unui strat de lichid între doua plăci metalice cufundate în lichid. Un rezistor variabil este un rezistor a carui rezistență electrică poate fi ajustată prin deplasarea mecanică a unui contact(cursor) electric intermediar, cel mai adesea rezistoarele de acest tip au trei terminale: capetele rezistorului(între care rezistența este maximă și constantă) și conexiunea la contactul mobil(cursor). Dacă contactul mobil nu face punct comun cu unul din capete, atunci uzial se vorbește despre “potențiometru”, care este un divisor variabil de tensiune.

Pentru alimentarea componentelor electronice am folost 8 acumulatori a câte 1.2V fiecare, în total având 9.6V, o tensiune mult prea mare pentru unele componente cum ar fi cipul Atmega328p care funcționează la o tensiune de maxim 5V. Pentru rezolvarea acestui inconvenient am folosit un stabilizator de tensiune (L7805).

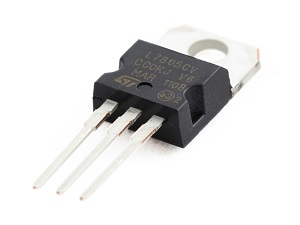


Fig. 20: Stabilizatorul de tensiune L7805

Un stabilizator de tensiune continuă este un cuadripol, care menține tensiunea de ieșire în limite foarte strânse (constante), indiferent de variația tensiunii de intrare. A curentului prin sarcina, sau a temperaturii mediului ambiant, în domenii specificate prin standarde sau norme tehnice. Stabilizatoarele de tensiune continuă fac parte din structura surselor de alimentare alături de transformator, de blocul redresor și de blocul de filtraj.

Până acum am vorbit de funcționarea anumitor dispozitive dar am omis detalii și principii care stau la baza acelor componente electrice.

Unul din cele mai importante concepte este curentul electric. Acesta reprezintă deplasarea dirijată a sarcinilor electrice. Există doua marimi fizice care caracterizează un curent electric. Prima este intensitatea curentului electric, numită adesea simplu tot curent electric, care caracterizează global curentul, referindu-se la cantitatea de sarcină electrică ce străbate secțiunea considerate în unitate de timp. Se măsoară în ampere. A doua, densitatea de curent este o mărime vectorială asociată fiecărui punct, intensitatea curentului regăsindu-se ca integral pe întreaga secțiune a conductorului din densitatea de curent. Se măsoară în amperi pe metru patrat.

Proiectul nostru folosește un accumulator deci este un curent de tip continuu. Curentul continuu prescurtat în engleză DC, este o mișcare a sarcinii electrice într-un singur sens printr-un mediu oarecare. El este produs de baterii electrice (galvanice). Termocuple, cellule(baterii) electrice solare, generatoare electrice cu collector-comutator (dinam).

Curentul continuu poate parcurge diferite medii cum ar fi conductorii electrici metalici (sârma) materiale semiconductoare, vacuum, unde curentul este reprezentat ca flux/fascicul de ioni sau electroni, medii isolate(izolatori), unde are valori infima care în practică cunt neglijabile.

Intensitatea curentului electric continuu este cantitatea sarcinilor electrice care trec printr-o sectiune traversală de mediu conductor într-o secundă. I=Q/t unde I este intensitatea curentului electric în ampere, Q este cantitatea de sarcini electrice în coulombi și t este timpul în secunde.

Tensiunea electrică între doua puncta ale unui circuit electric este diferența de potential între cele doua puncte și este proporțională cu energia necesară deplasării de la un punct la celălalt a unei sarcini electrice. O definitie mult mai succintă ar fi accea că tensiunea electrică este mărimea fizică scalar egală cu raportul dintre lucrul total fectuat de câmpul electruc pentru a transporta sarcina electrică pe întregul circuit și mărimea sarcinii electrice. U=L/Q unde U este tensiunea electromotoare, L este lucrul forței electrice și Q este sarcina electrică.

Legea lui Ohm sau legea conducției electrice stabilește legăturile între intensitatea curentului electric dintr-un circuit electric și tensiunea electrică(U) aplicată și rezistența electrică (R) din circuit.Legea lui Ohm se poate aplica și unei porțiuni de circuit. Legea lui Ohm se aplică pentru conductor electrici la capetele cărora se aplică tensiuni electrice. Legea lui Ohm spune că într-un circuit intensitatea (I) curentului electric este direct proporțională cu rezistența(R) din circuit. Formula matematică a legii lui Ohm este: I=U/R unde I este intensitatea curentului masurată în ampere, U este tensiunea aplicată, măsurată în volți și R este rezistența circuitului măsurată în ohmi.

**Capitolul 3**

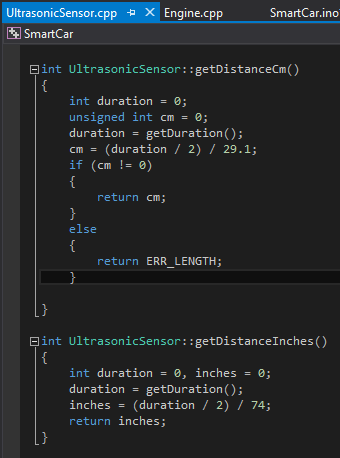
**3.1 Implementare**

Implementarea software a proiectului a fost făcută într-un mod iterativ, fiind realizată în paralel cu dezvoltarea parții hardware a proiectului. În cele ce urmează voi reproduce pas cu pas într-un mod cât mai descriptiv cum a fost realizată partea software a proiectului.

Primul pas făcut atunci când am interacționat prima dată cu platforma Arduino a fost banalul program “Blink” care nu face altceva decât să facă un led să se deschidă și să se închidă la un anumit interval de timp pe care îl prestabilim cu ajutorul funcției delay.

Dupa realizarea acestui program si vizualizarea interacțiunii dintre hardware și software am decis să mă aventurez mai mult în lumea Arduino.

Implementarea proiectului “Smart Car ” a început cu scrierea librariilor pentru fiecare componentă hardware folosită în proiect. Testarea a fost un pas foarte important deoarece uneori era nevoie de o calibrare software pentru ca acele componente să ofere rezultatul dorit.

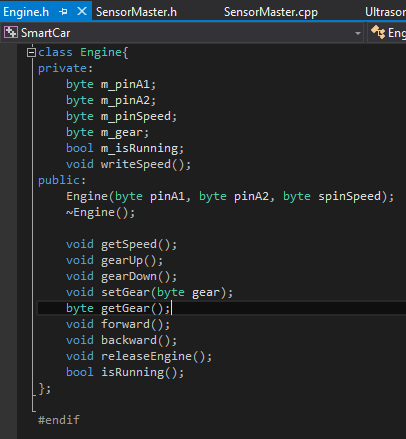
 Prima clasă scrisă pentru acest proiect a fost clasa “UltrasonicSensor” care realiza controlul unui senzor ultrasonic. Clasa are la bază trei funcții care realizează trimiterea unui semnal și mai apoi receptarea acelui semnal ultrasonic care va fi convertit mai apoi în ceea ce avem nevoie, mai exact distanța de la modul la obstacol. Cum spuneam și mai sus, în această clasă au fost implementate trei funcții, una din ele fiind folosită în celelalte doua deci a fost declarată ca fiind private iar celelalte doua funcții sunt publice. Funcția privată “getDuration()” trimite un semnal de cinci microsecunde prin pinul trig care va genera 8 impulsuri ultrasonice. După ce acel semnal a fost trimis, el este captat de către senzorul conectat la pinul echo. Această funcție are ca și output durata necesară parcurgerii undei de la modul la obstacol. Cele două funcții publice “getDistanceCm()” și ”getDistanceInches()”

Apelează funcția private de mai sus și folosesc datele produse de această funcție pentru a converti distanța în centimetri sau în inci.

Fig. 21: Funcții din clasa UltrasonicSensor

Având în vedere ca există patru senzori cu ultrasunete și fiecare trebuie să fie controlat de către un asemenea obiect m-am gandit să impachetez cele 4 obiecte de tip “UltrasonicSensor” într-o clasă “SensorMaster”. Clasa “SenzorMaster” ajută la o mai bună organizare a datelor venite de la cei patru senzori. Clasa aceasta are ca și membru un array de obiecte “UltrasonicSensor” iar cînd vrem să colectăm toate datele nu trebuie să facem decât o iterație prin acel array iar fiecare obiect va oferi datele necesare care mai apoi vor fi stocate într-o structura.

După parerea mea este mult mai ușor să împachetăm toate datele într-o singură clasă si mai apoi să facem apel la un singur obiect pentru a obține datele necesare decât sa facem de fiecare data când avem nevoie de datele de la senzori un apel către fiecare obiect în parte, fiecare semnificând un senzor.

 După ce ne-am asigurat că avem toate datele de la senzori putem să trecem mai departe la implementarea celorlalte clase. Urmatoarea clasă pe care am avut-o în vedere dupa ce am facut rost de toate datele de la senzori a fost clasa ce se ocupă de controlul celor patru motoare existente.

Clasa “Engine” este clasa responsabilă de controlul motoarelor electrice. Deoarece puntea H nu suportă decât controlul a doua motoare, a trebuit sa grupez cele 4 motoare în grupe de câte două pentru a beneficia de avantajele pe care le oferă puntea H. Deci un obiect de tip Engine va controla defapt doua motoare, fie motoarele de pe partea dreaptă fie cele de pe partea stângă. De asemenea am implementat și un sistem de șase viteze cu ajutorul semnalelor PWM. Vitezele sunt

Fig. 22: Funcțiile pentru controlul motoarelor

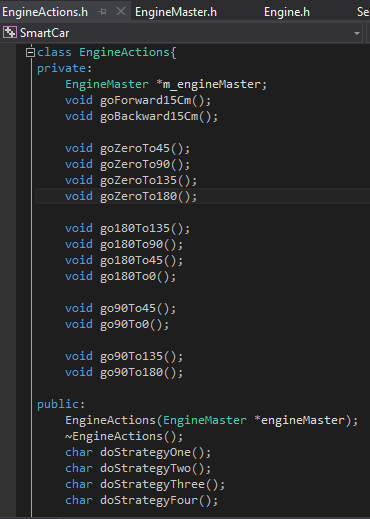
setate cu ajutorul funcțiilor gearUp() și gearDown() care nu fac altceva decât sa incrementeze sau să decrementeze o variabilă care reprezintă viteza actuală a mașinii.

De asemenea pentru a accesa viteza actuală putem apela funcția getGear() care ne întoarce viteza la care ne aflăm la acel moment. Funcția forward() dirijează cele două motoare spre direcția înainte cu viteza pe care am satat-o înainte să apelăm funcția forward(). Daca viteza de deplasare este 0 atunci va aparea mesajul de erare “Nu mergi nicaieri, ești la relanti“. Funcția backward() funcționează după aceleași principii ca și funcția forward() doar ca direcția de deplasare este opusă. Funcția releaseEngine() eliberează motoarele de orice tensiune, pinii ce sunt responsabili de motoare fiind setați pe LOW.

La fel ca și la senzori am ales să controlez cele doua grupuri de motoare cu ajutorul unei alte clase care are ca si membri cele doua grupuri de motoare. Această funcție nu face altceva decât sa folosească cele doua grupuri de motoare pentru a realiza cele patru acțiuni principale de care are nevoie robotul nostru și anume goForward(), goBackward(), goLeft() și goRight(). De data aceasta nu mai controlăm doar un grup de motoare cum făceam la funcția forward() din clasa Engine ci controlăm ambele grupuri pentru a realiza mișcările propuse. Funcția goForward() dirijează de data aceasta toate cele patru motoare: spre direcția înainte, toate având aceeași viteză stabilită înainte sa plecăm de pe loc. Funcția goBackward() funcționează după aceleași principii doar că motoarele se deplasează în sens invers. Funcțiile goLeft() și goRight() ofera robotului posibilitatea de a vira la stânga sau la dreapta, fiind folosit un singur grup de motoare pentru aceste acțiuni. De asemenea toate aceste funcții nu ar fi funcționale dacă nu am avea stabilita o viteză generală pentru cele doua grupuri de motoare. Această problemă este rezolvată de către funcția setGear(byte gear), parametrul acesteia fiind viteza cu care vrem să ne deplasăm.

După cum am spus și în celelalte capitole, robotul are 2 stări/moduri în care se poate afla la un moment dat, acestea fiind modul automat și manual.

Pentru implementarea modului automat a fost nevoie de o clasă care să facă posibilă îndeplinirea unor mișcări precise de către robot cum ar fi virajul la dreapta de la 90 de grade la 135 de grade.

 Clasa EngineActions ne oferă un control precis asupra mișcărilor pe care robotul trebuie să le facă atunci când de exemplu a întâlnit un obstacol.

După cum vedem și în imaginea din dreapta prima și prima dată avem nevoie de un obiect de tip EngineMaster care să ne ne ofere cele 4 funcționalitați de bază. Funcțiile private nu sunt altceva decât apelări ale celor patru acțiuni lăsate active doar o scurtă perioadă de timp. De exemplu funcția goForward15Cm() apelează funcția goForward din clasa EngineMaster și o ține activă o scurtă perioadă de timp cât este necesar pentru parcurgerea celor 15 cm. Dupa ce acel delay a expirat este apelată funcția releaseEngines(). Un alt exemplu potrivit ar fi funcția go90To45() care ajută roborul să faca o mișcare de 45 de grade la dreapta. Prima dată este apelată funcția goRight(), după apelare este pus un delay pentru o perioadă foarte scurta de timp iar după ce acel delay a expirat este apelată funcția releaseEngines().

Fig. 23: Acțiunile definite pentru controlul

motoarelor atunci când suntem în modul automat

Funcțiile publice din această clasă reprezintă cele 16 strategii pe care robotul le aplică în funcție de datele primite de la senzori.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Left Sensor | Front Sensor | Right Sensor | Rear Sensor | Result0 |
| 0 | 0 | 0 | 0 | goForward15Cm() |
| 0 | 0 | 0 | 1 | goForward15Cm |
| 0 | 0 | 1 | 0 | Go90To45() |
| 0 | 0 | 1 | 1 | Go90To45() |
| 0 | 1 | 0 | 0 | Go90To0() |
| 0 | 1 | 0 | 1 | Go90To0() |
| 0 | 1 | 1 | 0 | Go90To0() |
| 0 | 1 | 1 | 1 | Go90To0() |
| 1 | 0 | 0 | 0 | Go90To135() |
| 1 | 0 | 0 | 1 | Go90To180() |
| 1 | 0 | 1 | 0 | goForward15Cm() |
| 1 | 0 | 1 | 1 | goForward15cm() |
| 1 | 1 | 0 | 0 | Go90To180() |
| 1 | 1 | 0 | 1 | Go90To135() |
| 1 | 1 | 1 | 0 | Go90To180()->go90To180() |
| 1 | 1 | 1 | 1 | Error |

Fig. 24: Tabelul de strategii folosit pentru implementarea algoritmului de evitare a obstacolelor

După cum am văzut și în tabelul de mai sus acele funcții publice din clasa EngineActions sunt apelate în funcție de valorile pe care le primim de la senzori.

Pentru a afla care strategie trebuie aleasă a fost facută o altă clasă numită AvoidObstacleStrategy. Această clasă conține o funcție care are ca si parametru o structură cu cele 4 valori ale senzorilor la un anumit moment. Atunci când este apelată funcția getStrategy din clasa AvoidObstacleStrategy, valorile de la senzori sunt trecute printr-un filtru în care valorile sunt verificate pe rând până este setată o strategie pe care sâ o întoarcă funcția. Rezultatul de la această funcție este la rândul său folosit în clasa ThinkAction.

Clasa ThinkAction este responsabilă cu modul automat. Această clasă este responsabilă cu acțiunile pe care le face robotul atunci când acesta acționează în mod independent. Funcția are ca și membrii referințe la alte clase cum ar fi clasa SensorMaster pentru a face rost de datele de la senzori, clasa EngineActions pentru a controla cele miscarile robotului precum și o referință la clasa AvoidObstacleStrategy pentru a prelucra datele recepționate de la senzori.

În clasa ThinkAction funcția care face procesarea datelor și alegerea unei strategii este funcția think(). În interiorul acestei funcții datele de la senzori sunt procesate cu ajutorul funcției getStrategy din clasa AvoidObstacleStrategy. Dupa ce această funcție ne-a spus ce strategie să aplicam, putem folosi un switch pentru a procesa acea stratgie și să apelăm funcția responsabilă de acțiunile care trebuie sa fie realizate. Modul automat este setat de către utilizator prin intermediul aplicației desktop. Atunci cînd este setat acesta realizează un anumit număr de iterații, la fiecare iterație fiind realizată una din cele 16 strategii. Atunci când o strategie este aplicată, acea strategie generează un cod, care va ajuta la cartografierea făcută pe aplicația desktop.

Modul manual al robotului este implementat în clasa ManualMode care la rândul ei se folosește de clasa ManualModeActions. În clasa ManualModeActions avem definită funcția driveCarManually care pe baza parametrului care este de fapt o comanda, această funcție va realiza o acțiune. Acțiunile pe care această funcție le poate stabili sunt schimbarea celor 6 viteze, precum și cele 4 miscari de bază forward()/ backward()/ left()/right().

Aplicația oferă utilizatorului șansa de a schimba cele doua moduri de funcționare ale robotului. Fiecare din cele doua moduri de funcționare este reprezentat de către un agoritm. Pentru a face interschimbarea acestora atunci când aplicația rulează am folosit șablonul de proiectare Strategy. Acest șablon presupune încapsularea separată a fiecărui algoritm dintr-o familie, căcând astfel ca algoritmii respectivi să fie interschimbabili.

Șablonul arhitectural Strategy este un șablon de tip comportamental ce a fost conceput pentru a defini o familie de algoritmui ce sunt încapsulați de către o interfață. Scopul acestui șablon este de a selecta algoritmul optim pentru necesitatea fiecărui client, în funcție de specificațiile acestuia. De exemplu, avem de sortat un vector de numere. Pentru aceasta putem folosi algoritmi ca Bubble Sort, Merge Sort sau Quick Sort etc. Șablonul Strategy oferă utilizatorului posibilitatea de a alege singur ce algoritm poate folosi pentru a sorta vectorul său. Acest șablon este folosit atunci cnd putem folosi mai mulți algoritmi pentru o operație sau atunci când într-o clasă sunt definte mai multe comportamente, dar mai ales atunci când dorim să evităm duplicarea codului.

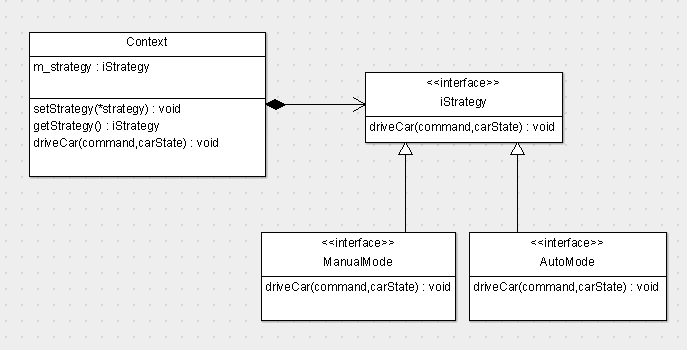


Fig. 25: Șablonul Strategy

Așa cum am specificat și mai devreme, șablonul Strategy a fost folosit pentru a face interschimbarea dintre modul manual al robotului și modul automat. Am ales să folosesc acest tip de implementare deoarece este o soluție simplă de schimbare a algoritmului la runtime și ne ajută să evităm duplicarea codului.

Clasa Context are ca și membru o interfată de tip iStrategy. Această clasă se ocupă de gestionarea algoritmilor cu ajutorul acelei interfețe. Funcțiile din clasa context ne oferă posibilitatea să setăm o strategie cu ajutorul funcției setStrategy, de asemenea putem prelua acea strategie pentru a o folosi în alte scopuri, iar pe lângă aceste două funcții mai avem nevoie de o altă funcție care va ajuta la rularea algoritmului/strategiei alese, în cazul nostru fiind funcția driveCar. Fiecare strategie este reprezentată de o clasă care va trebui să mostenească interfața iStrategy și implicit, clasa aceasta va trebui să implementeze funcția virtuală definită în interfața iStrategy.

După cum spuneam, atunci când algoritmul modului automat este activ, acesta nu numai că aplică diferite strategii pentru evitarea obstacolelor dar și stochează anumite date care sunt necesare cartografierii terenului prin care a trecut robotul. Aceste date sunt stocate într-un fișier CSV pe un card micro SD cu ajutorul unui modul SD compatibil cu platforma arduino. După ce datele au fost stocate în fisierul CVS nu trebuie decât să încărcăm fișierul în alicația de pe PC și vom vizualiza traseul făcut de robot.

**3.3 Aplicația desktop**

Rolul principal al acestei aplicații este de a trimite comenzi robotului aflat pe teren. Aplicația a fost realizață în totalitate folosind limbajul de programare Java și setul de componente din librăria Swing. Pentru realizarea interfeței au fost folosite componente ca Jbutton, Jpanel, JtextField, JComboBox precum și Jlabel, toate fiind primitive existente în această librărie. Prima dată când deschidem aplicația trebuie să ne asigurăm ca există o conexiune de la PC la placa Arduino care are atașat modulul radio de tip Transmitter care se ocupă doar cu trimiterea datelor.Cum spuneam și mai sus, conexiunea se realizează printr-un cablu USB, fiind prezentă o comunicație bidirecțională. După ce ne-am asigurat că există o conexiune trebuie să ne conectăm la placă prin alegerea portului dintr-o listă de tip JComboBox. În acea listă gasim listat fiecare port USB la care este conectat o placă de tip Arduino. După alegerea portului nu trebuie decât să apăsăm pe butonul connect aflat în stânga listei de porturi. Conectarea va fi confirmată de către un led aflat pe placa Arduino, mai exact ledul Rx.

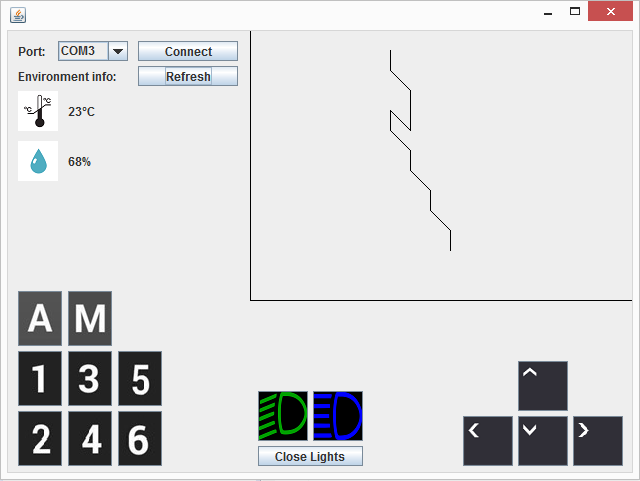


Fig. 26: Aplicația desktop

Butoanele existente în această inteftață sunt controlate de la tastatură, excepție făcând butonul connect folosit pentru conectarea laplaca Arduino și butonul refresh folosit pentru a trimite o cerere către placa Arduino, cerere care ar trebui să întoarcă datele despre vreme, respectiv temperatura și umiditatea. Butoanele din partea, respectiv săgețile ajută la controlul robotului atunci când acesta se află în modul manual. Butoanele din stânga jos sunt pentru selecția vitezei de mers și pentru a face interschimbarea între cele doua strategii.

Detalidfile despre mediu, respectiv temperatura și umiditatea sunt captate cu ajutorul senzorului DHT11 V2 conectat la placa Arduino de la PC. Atunci când avem nevoie de informațiile culese de senzor trebuie să apăsăm pe butonul refresh care va face o cerere către placa arduino. După ce semnalul trimis prin mufa serială a fost captat de placă, acesta trimite înapoi o structură având în interiorul ei datele necesare, respectiv temperatura și umiditatea.

Cartografierea despre care am vorbit și mai sus este reprezentată in partea dreaptă, delimitat de o linie, linia aceea având rolul de granițâ pentru desenul realizat pe baza datelor din fișierul trimis de către robot. Desenarea acelor linii care reprezintă ruta traversată de robot este realizată cu ajutorul Clasei “Line2D” care se află în pachetul de clase AWT. Clasa Line2D ne ajută să desenăm un segment de linie cu ajutorul unor coordinate spațiale (x,y). Liniile desenate pe interfață au fost realizate pornind de la doua coordonete de start reprezentate de variabilele “startX” și “startY” după care trebuie să tratăm toate cele 16 posibile strategii pe care robotul le folosește pentru a evita obstacolele. Alegerea tipului de linie pe care trebuie să îl desenăm se face pe baza datelor din fișier, fiecare valoare fiind trecută printr-o instrucțiune switch care mai apoi aplica una din cele 16 cazuri, fiecare caz desenând câte un tip diferit de linie.

**3.4 Proiectare hardware**

Hardware-ul este partea fizică a unui sistem, spre deosebire de software care este partea logică, cea care comandă hardware-ul prin intermediu unor programe care pot fi aplicații, sisteme de operare sau drivere. De asemenea, el reprezintă ansamblul elementelor fizice și tehnice cu ajutorul cărora datele se pot culege, verifica, prelucra, transmite, afișa și stoca.

La nivel hardware proiectul dispune de numeroase echipamente care au ajutat la dezvoltarea acestui proiect. Cea mai importantă parte atunci când vorbim de partea hardware sunt bineânțeles micorcontrollerele ATmega328p. Pe lângă aceste microcontrollere au fost folosite diverse periferice care au ajutat fie la interacțiunea cu mediul fie la mișcarea robotului sau la stocarea unor date.

Robotul beneficiază de douămicrocontrollere Atmega328p, unul din ele fiind montat pe o plăcuță arduino iar celălalt este montat pe un soclu realizat de către mine. Am abordat această idee, de a folosi doua microcontrollere deoarece multitudinea de periferice existente au ocupat toate porturile de pe microcontrollerul inițial montat pe robot.

În schema electrică de mai jos sunt reprezentate cele două microcontrollere folosite pentru controlul robotului. Primul microcontroller se ocupă de recepționarea datelor, având conectat un modul de tip Receiver. După ce aceesta a recepționat datele, ele sunt trecute printr-un filtru iar mai apoi trimise la celălalt microcontroller cu ajutorul pinilor de comunicare serială RX/TX. Pentru conectarea celor doua cipuri, pinul Rx de pe primul cip va fi atașat la pinul Tx de pe al doilea cip iar pinul Tx la pinul Rx. Cum spuneam, acest microcontroller nu se ocupă decât de recepționarea datelor deci ca și periferice atașate nu are decât receiverul, un led care se aprinde atunci când receiverul primește date, precum și componentele aferente funcționării micoronotroller-ului (oscilatorul și cei doi condensatori).

Al doilea microcontroller se ocupă de recepționarea datelor de la primul microcontroller și controlul perifericelor. Ca și periferice conectate la acest microcontroller avem cei patru senzori cu ultrasunete care ocupă în total 8 pini I/O și sunt alimentați direct de la placa Arduino la 5V. Cele patru motoare nu sunt conectate în mod direct la microcontroller ci sunt conectate mai întâi la o punte H care ne ajută să schimbăm sensul de mers al motoarelor și de asemenea să controlăm viteza cu care se deplasează. Pentru conectarea punții H la microcontroller avem nevoie de 6 pini printre care doi trebuie să fie pini PWM. Acești doi pini sunt folosiți pentru a implementa un sistem de viteze asupra motoarelor. După conectarea tuturor perifericelor se observă, că am ramas fără pini liberi pe acest microcontroller, de aici și nevoia celui de-al doilea microcontroller.

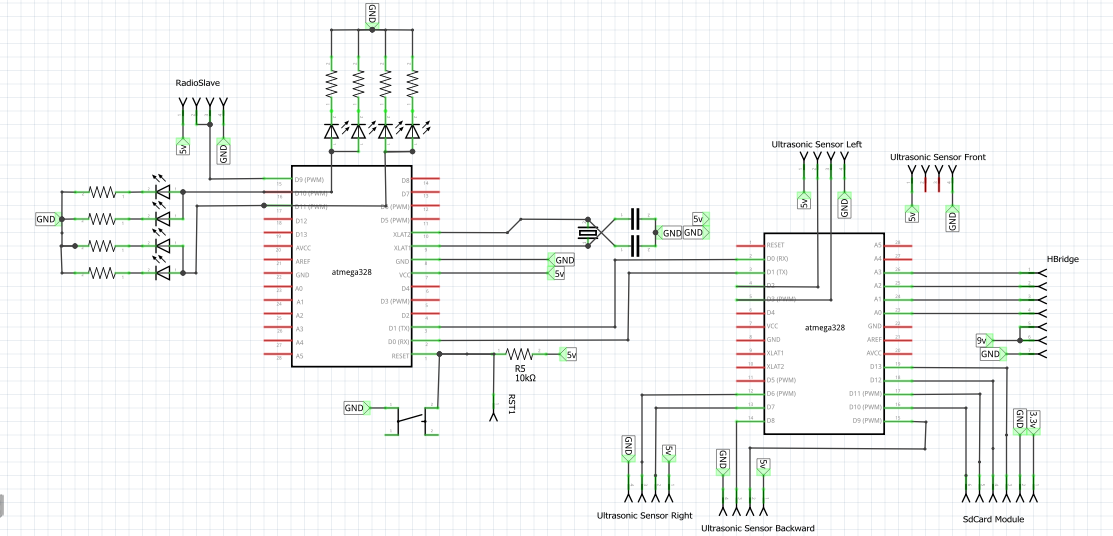


Fig. 27: Schema electrică

După cum spuneam și mai sus, la primul microcontroller se ocupă doar de recepționarea datelor. Observăm în stânga sus Receiverul cu numele “Radio Slave” conectat la pinul digital 2. După ce am recepționat Semnalul cu ajutorul Receiverului, acesta este mai departe transmis prin pinul 2 la microcontroller care face o filtrare a acestuia înainte ca semnalul sau mai bine zis comanda să fie transmisă spre cel de-al doilea microcontroller. Ca o paranteză, putem observa că între cele doua microcontrollere este o relație de tip master-slave unde masterul este reprezentat de primul cip iar slave-ul este al doilea cip care primește comenzile. După ce al doilea cip a primit o comandă, slave-ul trece această comandă la rândul lui printr-un alt filtru pentru a interpreta comanda. În funcție de starea în care se afla robotul (manuală/automată), acea comandă va fi interpretată în mod diferit sau uneori chiar va fi ignorată daca robotul se află în starea automată si execută comenzile într-o buclă care nu poate fi oprită manual.

Microcontroller-ul master nu este montat pe o placă Arduino așa cum este montat slave-ul. Acesta se află pe un soclu care la rândul lui se află conectat pe o placă de test. Pe prima suprafață a plăcii găsim soclul pentru microcontroller precum și pinii mamă de inserție a componentelor legate la cip. Comunicația/conexiunea între componente se realizează pe cealaltă față a plăcii cu ajutorul unor fire.

**3.4 Librarii folosite**

În realizarea acestui proiect nu a lipsit folosirea unor librarii deja existente care au fost integrate în proiect.

Un exemplu de lirarie folosită pentru realizarea comuncației dintre PC și Arduino este libraria RXTX. Această librarie oferă un protocol de comunicare precum si un set de funcții care ne ajută să trimitem date de la PC placa Arduino, comunicația fizică fiind făcută cu ajutorul unui cablu USB.

Librăria VirtualWire a fost folosită pentru modulele radio. Această librarie oferă un set de funcții ce ajută la manipularea modulelor radio.

Cea mai importantă librarie din acest proiect este, evident libraria Arduino.h. Aceasta oferă toate funcționalitățile de care avem nevoie pentru proiectul nostru arduino.

De asemena, fiecare componentă fizică a fost însoțită de o librărie scrisa ori de producător ori de un utilizator ce a făcut câteva experimente cu acea componentă și a realizat mai apoi o librarie pentru a usura lucrul cu acea componentă.

**Concluzile lucrării**

După părerea mea, platforma Arduino ne oferă o varietate nelimitată de posibilitați atunci când vine vorba de realizarea unui proiect. Simplitatea, flexibilitatea și costurile reduse o face accesibilă oricui vrea să se aventureze în lumea roboticii. Am început acest proiect dintr-o simplă curiozitate, și anume, voiam să văd cum anume funcționeaza dispozitivele electrice prezente oriunde mergem. Prin simplitatea sa, Arduino ne face să intelegem cum funcționează anumite lucruri pe care le întâlnim în viața de zi cu zi și să ne dăm seama cum am putea să construim și noi la rândul nostru un asemenea dispozitiv. Un exemplu banal la care mă gandeam zilele trecute că s-ar putea realiza foarte simplu cu o placuță Arduino este controlul unei mașini de spalat sau al oricarui dispozitiv pe care îl găsim într-o bucătărie sau de ce nu într-o casă. Cum spuneam, această platformă oferă posibilități nelimitate de dezvoltare atât pe partea software unde avem o multitudine de librarii la dispoziție, unele librarii fiind realizate chiar de către utilizatori Arduino dar și pe partea hardware unde anumite companii oferă componente special realizate pentru această platformă.

Costul realizarii unui proiect având la bază această platformă este foarte mic, proiectul inițial fiind realizat pentru studenți, ceea ce ne arată faptul că Arduino este în primul rând un proiect ce ne ajută să oferim unor studenți de exemplu anumite concepte teoretice și practice într-un mod mult mai distractiv și simplu decât banalele cursuri unde ne sunt prezentate doar niște concepte teoretice greu de digerat.

Datorita acestor avantaje, Arduino a crescut într-un ritm foarte rapid. În anul 2010, Arduino a fost lansat urmând ca în anul 2011 să fie produse aproape 300.000 de plăcuțe de acest tip. În anul 2013 700.000 de plăcuțe Arduino erau deja în posesia utilizatorilor.

Un proiect de acest gen ne ajută să avem o viziune mai amplă asupra lumii dar mai ales a dispozitivelor care ne înconjoara. Având în vedere că peste tot suntem înconjurați de tehnologie, nu este rău să știm și cum sunt realizate aceste dispozitive. Arduino nu este numai o platformă de dezvoltarea ci și un bun mentor atunci când vrem să învățăm anumite concepte, iar cu ajutorul platformei Arduino putem să și testăm acele concepte.

Bibliografie:

[1] J.M. Hughes: Arduino A Technical Reference

[2] John Nussey: Arduino for Dummies

[3] Brock Craft: Arduino projects for Dummies

[4] John Boxall: Arduino Workshop

[5] Jeremy Blum: Exploring Arduino

[6] https://ro.wikipedia.org/wiki/Arduino

[7] https://www.arduino.cc/

[8] http://www.imdb.com/title/tt1869268/

[9] https://www.arduino.cc/en/Guide/Libraries

[10] http://forum.arduino.cc/index.php?/topic,132130.0.html

[11] https://www.agerpres.ro/sci-tech

[12] https://www.baldengineer.com/5-arduino-myths.html

[13] https://readwrite.com/2014/03/29/10-arduino-projects-microcontroller-electrical-engineering/

[14] http://www.instructables.com/id/20-Unbelievable-Arduino-Projects/

[15] https://www.baldengineer.com/5-arduino-myths.html

[16] https://www.arduino.cc/en/Guide/Introduction

[17] https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/

[18] https://ro.wikipedia.org/wiki/SMD

[19] https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/

[20] https://arduino.stackexchange.com/questions/16348/how-do-you-use-spi-on-an-arduino

[21] http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[22] https://ro.wikipedia.org/wiki/Motor\_electric#Motorul\_pas\_cu\_pas

[23] https://ro.wikipedia.org/wiki/Motor\_electric\_de\_curent\_continuu

[24] https://thefishjunkies.com/ce-este-un-reductor-tipuri-si-detalii-tehnice/

[25] http://www.electronicstefan.ro/2012/01/ce-este-puntea-h/

[26] https://en.wikipedia.org/wiki/H\_bridge

[27] https://wiki.eprolabs.com/index.php?title=Humidity\_Sensor\_DHT11

[28] https://ro.wikipedia.org/wiki/LED

[29] https://ro.wikipedia.org/wiki/Rezisten%C8%9B%C4%83\_electric%C4%83

[30] http://www.rasfoiesc.com/inginerie/electronica/STABILIZATOARE-DE-TENSIUNE86.php

[31] https://ro.wikipedia.org/wiki/Curent\_electric

[32] https://ro.wikipedia.org/wiki/Curent\_continuu

[33] https://ro.wikipedia.org/wiki/Legea\_lui\_Ohm

[34] http://www.preferatele.com/docs/informatica/noi/java-sing-5241142411.php

[35] https://ro.wikipedia.org/wiki/SMD