UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Proiector de imagine cu laser

propusă de

Alexandru Tudor Savencu

Sesiunea: *februarie, 2020*

Coordonator științific

Lect. Dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Proiector de imagine cu laser

Alexandru Tudor Savencu

Sesiunea: *februarie, 2020*

Coordonator științific

Lect. Dr. Anca Ignat

Avizat,

Îndrumător Lucrare de Licență

Lect. Dr. Anca Ignat

Data..... Semnătura.....

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul Savencu Alexandru Tudor, domiciliul în Iași, născut la data de 27.05.1997, identificat prin CNP 1970527226712, absolvent al Universității „Alexandru Ioan Cuza" din Iași, Facultatea de Informatica, specializarea Informatica, promoția 2019, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: „Proiector de imagine cu laser", elaborată sub îndrumarea d-na Anca Ignat, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

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

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

Dată azi, …… Semnătură student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

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

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

Iași, ...

Absolvent,

Alexandru Tudor Savencu

.....

CUPRINS

INTRODUCERE

Publicitatea stradală este una dintre cele mai creative tipuri de comunicare prin intermediul căreia se transmite mesajul publicitar către consumatori. Acest tip de publicitate poate fi prezent sub forma uni afiș sau panou publicitar care include reclame pe străzi, clădiri, vehicule.

Descrierea problemei

De puțină vreme, legislația pe teritoriul țării noastre a început să fie din ce în ce mai impunătoare în ceea ce privește reglementările privind publicitatea stradală și amplasarea acestor tipuri de panouri, afișe pubicitare. Acest fapt se datorează costurilor ridicare și plângerilor tot mai numeroase ale cetățenilor.

Potrivit acestor prevederi, s-a dispus realizarea unor spații destinate special publicității stradale, drumurile căpătând un loc de cinste în clasamentul acestor spații, deoarece sunt vizibile și accesibile unei mari caterorii de consumatori. Indiferent de modul în care se face publicitatea stradală, aceasta trebuie să aibă un text scurt și clar, efecte vizuale intense și atractive și, totodată, o dimensiune considerabilă pentru a atrage consumatorii.

Principalele dezavantaje al panourilor și afișelor publicitare sunt costurile ridicate ce implica angajarea personalului specializat -alpiniști- și materialelor folosite: sistemul de iluminat, sistemul de prindere, banner-ul în sine, a chiriei și a energiei electrice. Totodată, schimbarea anunțului publicitar necesită timp datorită faptului că trebuie înlocuit tot banner-ul.

În plus, mijloacele convenționale de publicitate contravin arhitecturii și a design-ului unui anumit spațiu sau a unui oraș. De cele mai multe ori acestea sunt poziționate în locuri centrale, neadaptandu-se la peisaj.

O alta problemă a acestor sisteme de publicitate este că nu intodeauna sunt foarte bine securizate, fiind situate în spații publice aglomerate, au fost și cazuri în care acestea au devenit un adevărat pericol datorită instabilității lor.

De asemenea, un dezavantaj important îl reprezintă faptul că afisele respective nu pot fi refolosite, acestea ajungând de cele mai multe ori la gunoi, astfel făcându-se risipă de materiale, fapt ce duce, în final, la degradarea ecosistemului.

Soluția propusa

Soluția propusă este proiecția unei imagini folosind laserul prin intermediul unui sistem galvo. Conceptul de fascicol laser reflectat de oglinzi atașate pe actuatori automatizați este un concept des întâlnit în efecte de scenă la concerte și festivaluri.

Avanajul folosirii laserului este că suprafață de proiecție se poate afla la o distanță destul de mare, fară a fi făcută dintr-un material special, ci doar suficient de nereflectiv. Materialele folosite pentru a construi un astfel de dispozitiv sunt relativ simple.

Proiecția laser nu depinde de condițiile meteo, întrucât radiația laser poate fi suficient de puternică pentru a nu fi influențată de picăturile de ploaie sau fulgii de zăpadă.

Spre deosebire de panourile publicitare de pe fațadele clădirilor unde este necesar panoul, sistem de luminiat, alpiniști specializați, sistemul laser nu necesită decât o fațadă de plană și poate fi cu ușurință montat pe clădirea vecină suprafeței pe care vrem să facem proiecția laser, nemaifiind nevoie de alte măsuri suplimentare. Acest lucru duce la costuri reduse de montare și întreținere și datorită consumului redus de energie și a-l materialelor simple necesare, putem spune că este o modalitate eco-friendly de a face publicitate.

Motorul Galvo este un motor care permite rotirea axului la anumite unghuri cu o viteză mare pentru ca unda laser ce se reflectă din oglinizile amplasate la capătul lui, să poată capătă diferite forme.

Acest dispozitiv ar putea fi operat fară a fi necesara conexiunea la un calculator.

Structura capitolelor

În primul capitol, sunt prezentate aspecte teoretice asupra tuturor componentelor și a comunicațiilor folosite în acest proiect. Astfel, sunt prezentate următoarele concepte de bază: laserul, dioda laser, galvonametrul cu oglindă, driver galvo, microcontroller, digital to analgoconverter, comunicațiile SPI și UART.

În al doilea capitol, sunt descrise componentele alese pentru acest proiect impreună cu specificațiile lor și modul de funcționare. Astfel, sunt expuse următoarele componente: Microcontroller-ul, Digital to Anaog converter, Sistemul galvo.

În al treilea capitol, este prezentată schema generală a proiectului împreună cu rolul fiecarei componente in parte.

În capitolul patru se afla o scurta prezentare a pașilor pe care i-am urmat în rezolvarea problemelor apărute în dezvoltarea proiectului.

În capitolul cinci sunt detaliate tehnologiile folosite în proiect și detalii de implementare asupra codului scris atât pentru partea embedded, cât și pentru dezvoltarea high level a aplicației de procesare a imaginii dezvoltate in limbajul Python.

În sectiunea „Concluzii” se află concluziile lucrării, împreună cu posibilele îmbunătățiri ce pot fi aduse proiectului.

Ultima parte a lucrării cuprinde lista bibliografică care a fost consultată in vederea elaborării licenței.

CONTRIBUȚII

1) Documentare: Analiza problemelor, cercetarea soluțiilor adiacente și găsirea unei soluții viabile.
2) Design de sistem: Mecanică, Hardware, Software.
3) Alegerea subansamblului: Research pe diferite magazine online și găsirea celei mai bune variante din punct de vedere al prețului și calității.
4) Crearea de set-up: Așezarea și fixarea elementelor pe placa de pal. Integrarea componentelor într-un ansamblu.
5) Interfațare între partea analogică și partea digitală: DAC2Click, ClickerBoard, USB/UART converter.
6) Design Software: Construirea arhitecturii software.
7) Design Hardware: Integrare de subansamble si interfațare între MCU și restul contextului.
8) Programare Embedded: Programare in limbaj C, realizarea unui driver dac, uart, spi.
9) Programare High-Level: Programare in limbajul python pentru realizarea unei aplicații de procesare si transfer a imaginii de proiecție.
10) Testare: Testare software și hardware pentru a asigura o bună funcționare a proiectului.

1. NOȚIUNI DE BAZĂ

1.1 Laser

Laserul (Light Amplification by Stimulated Emission of Radiation) este un dispozitiv complex ce utilizează un mediu activ laser, ce poate fi solid, lichid sau gazos, și o cavitate optică rezonantă. Mediul activ, cu o compoziție și parametri determinați, primește energie din exterior prin ceea ce se numește pompare. [9]

Pomparea se poate realiza electric sau optic, folosind o sursă de lumină (flash, alt laser etc.) și duce la excitarea atomilor din mediul activ, adică aducerea unora din electronii din atomii mediului pe niveluri de energie superioare. Față de un mediu aflat în echilibru termic, acest mediu pompat ajunge să aibă mai mulți electroni pe stările de energie superioare, fenomen numit inversie de populație. Un fascicul de lumină care trece prin acest mediu activat va fi amplificat prin dezexcitarea stimulată a atomilor, proces în care un foton care interacționează cu un atom excitat determină emisia unui nou foton, de aceeași direcție, lungime de undă, fază și stare de polarizare. Astfel este posibil ca pornind de la un singur foton, generat prin emisie spontană, să se obțină un fascicul cu un număr imens de fotoni, toți având aceleași caracteristici cu fotonul inițial. Acest fapt determină caracteristica de coerență a fasciculelor laser. [9]

În funcție de tipul mediului activ și de modul în care se realizează pomparea acestuia laserul poate funcționa în undă continuă sau în impulsuri.[9]

Laserele sunt componente cheie ale multor produse pe care le folosim zilnic. Produsele de consum precum Blu-Ray și DVD playerele se bazează pe tehnologia laser pentru a citi informațiile de pe discuri. Scanerele cu coduri de bare se bazează pe lasere pentru procesarea informațiilor. Laserele sunt, de asemenea, utilizate în multe proceduri chirurgicale, cum ar fi chirurgia ochilor LASIK. Totodata, Laserele sunt utilizate pentru tăierea, gravarea, găurirea și marcarea unei game largi de materiale. [9]

1.2 Dioda laser

Figura 1: Dioda laser

O diodă laser este un dispozitiv optoelectronic, care transformă energia electrică în energie luminoasă pentru a produce o lumină coerentă de mare intensitate. Într-o diodă laser, joncțiunea pn a diodei semiconductoare acționează ca mediu laser sau mediu activ. [8]

Funcționarea diodei laser este aproape similară cu dioda cu emisie de lumină (LED). Diferența principală dintre LED și dioda laser este că LED-ul emite lumină incoerentă, în timp ce dioda laser emite lumină coerentă. [8]

O diodă de joncțiune pn este un dispozitiv semiconductor care permite fluxul de curent într-o singură direcție. Aceasta este formată din două tipuri de materiale semiconductoare și anume semiconductor de tip p și n. Semiconductorul de tip p este unit cu semiconductorul de tip n pentru a forma o joncțiune pn . Dispozitivul care rezultă din unirea unui semiconductor de tip p și n se numește diodă de joncțiune pn. [8]

Dioda laser este formată din două straturi de arsenidă de galiu dopate. Un strat de arsenidă de galiu dopat va produce un semiconductor de tip n, în timp ce un alt strat de arsenidă de galiu dopat va produce un semiconductor de tip p. În diodele cu laser, seleniu, aluminiu și siliciu sunt utilizate ca agenți de dopare. Când tensiunea continuă este aplicată pe dioda laser, electronii liberi

se deplasează în regiunea de joncțiune de la materialul de tip n la materialul de tip p. În acest proces, unii electroni vor interacționa direct cu electronii de valență și îi excită la un nivel mai ridicat de energie, în timp ce alți electroni se vor recombina cu găurile din semiconductorul de tip p și vor elibera energie sub formă de lumină. Acest proces de emisie se numește emisie spontană. [8]

Fotonii generați datorită emisiilor spontane vor călători prin regiunea de joncțiune și vor stimula electronii excitați (electroni liberi). Ca urmare, sunt eliberați mai mulți fotoni. Acest proces de emisie de lumină sau fotoni se numește emisie stimulată. Lumina generată din cauza emisiilor stimulate se va deplasa paralel cu joncțiunea. [8]

Cele două capete ale structurii diodei laser sunt reflectoare optic. Un capăt este reflectorizant complet, iar celalalt capăt este parțial reflectorizant. Capătul complet reflectorizant va reflecta complet lumina, în timp ce capătul parțial reflectorizant va reflecta cea mai mare parte a luminii, dar permite o cantitate mică de lumină. [8]

Lumina generată în joncțiunea pn va răsuna înainte și înapoi (de sute de ori) între cele două suprafețe reflectorizante. Drept urmare, se obține un câștig optic enorm. [8]

Lumina generată din cauza emisiilor stimulate este eliberata prin capătul parțial reflectorizant al diodei laser pentru a produce o lumină laser cu fascicul îngust. Toți fotonii generați datorită emisiilor stimulate vor circula în aceeași direcție. Prin urmare, această lumină va călători pe distanțe lungi, fără a se răspândi în spațiu. [8]

1.3 Galvonametru cu oglindă

Figura 2: Galvonometru cu oglindă

Un galvanometru cu oglindă este un instrument electromecanic care indică sesizarea unui curent electric prin devierea unui fascicul de lumină cu ajutorul unei oglinzi . Fasciculul de lumină proiectat pe o scară acționează ca un indicator lung fără masă. La inceputul secolului al-XIX-lea, Johann Christian Poggendorff a dezvoltat galvanometrul cu oglinzi pentru detectarea curenților electrici. Aparatul este cunoscut și sub denumirea de *galvanometru spot*, după punctul de lumină produs la unele modele. [5]

Galvanometrele cu oglindă au fost utilizate pe scară largă în instrumentele științifice înainte de a fi disponibile amplificatoare electronice stabile și fiabile. Cele mai frecvente utilizări au fost ca echipamente de înregistrare pentru seismometre și cabluri submarine utilizate pentru telegrafie.[5]

In zilele noastre, termenul de *galvanometru cu oglindă* este de obicei utilizat pentru dispozitivele care mișcă fascicule laser prin rotirea unei oglinzi intr-un sistem cu motoare galvo. Acest sistem este alcatuit din doua motoare rotative de inalta performanta special concepute pentru aplicatii optice, ce permit functionare fără erori in utilizarea pe termen lung si continuu.[5]

Dinamica lor excepțională este rezultatul anilor de dezvoltare în fabricarea sistemelor de scanare. Motoarele dispun de un senzor capacitiv de poziție foarte sensibil pentru o derivă scăzută, liniaritate îmbunătățită și unghiuri de deviere extinse. [4]

Aceste dispozitive sunt des folosite la spectacole cu lumină laser pentru a mișca fasciolul laser și pentru a produce modele geometrice colorate pe ceață din jurul publicului.

1.4 Driver Galvo

Driver-ul galvo controlează motoarele galvo și funcționează pe baza unui PID controller (Proporțional-Integrativ-Derivativ) implementat analogic. Acesta este un mecanism de autoreglare ce rulează în buclă închisă și funcționează pe baza feedback-ului primit. [18]

În cazul de față, referința este poziția dorită a oglinzii, ieșirea este comanda către motrul galvo, iar feedback-ul este semnalul analogic primit de la senzorul capacitiv din motoare. Pentru mai multe informații despre funcționarea unui PID controller se poate consulta bibliografia (v. [18])

1.5 Microcontroller Unit

Un MCU (Microcontroller Unit) este un circuit integrat compact, care este de obicei utilizat pentru o aplicație specifică și proiectat pentru a implementa anumite sarcini într-un sistem embedded. [6]

Un microcontroler tipic include un procesor, memorie și periferice de intrare / ieșire (I / O) pe un singur cip. În esență, un microcontroler colectează informații și produce anumte acțiuni bazate pe informațiile culese. MCU-urile funcționează de obicei la viteze mici, în jurul intervalului 1MHZ-200MHz și sunt proiectate pentru a consuma puțină energie. [6]

Acesta este, de obicei, încorporat într-un sistem embedded pentru a controla una sau mai multe funcții dintr-un dispozitiv, prin interpretarea datelor pe care le primește de la preferice sale I/O cu ajutorul procesorului central. Informațiile temporare pe care le primește MCU-ul sunt stocate în memoria sa de date. Procesorul accesează aceste informații folosind instrucțiunile stocate în memoria de programare pentru a prelucra și aplica informațiile ca mai apoi să efectueze acțiunea corespunzătoare folosind perifericele I/O. [6]

MCU-urile sunt utilizate într-o gama largă de sisteme și dispozitive. Dispozitivele utilizează de obicei mai multe MCU-uri care lucrează împreună pentru a rezolva mai multe sarcini.

De exemplu, o mașină poate avea mai multe microcontrolere care controlează diferite sisteme individuale din interior, precum sistemul de anti-blocare al roților, controlul tracțiunii, injecția de combustibil sau controlul suspensiei. Toate MCU-urile comunică între ele pentru a asigura o bună funcționare a sistemului. Unele MCU-uri pot comunica și cu computer central mai complex din interiorul mașinii iar altele comunică doar cu alte MCU-uri. Ele trimit și primesc date folsind prefericele I/O și prelucrează aceste date pentru a-și îndeplini sarcinile desemnate. [6]

Un MCU poate fi văzut ca un mic computer, acest lucru datorându-se componentelor esențiale din interorul acestuia. Principalele componente sunt procesorul, memoria și perifericele I/O.[11]

Procesorul (CPU) – procesează și raspunde la diverse instrucțiuni, acestea implicând efectuarea operațiilor de bază, aritmetică, logică și I/O. De asemenea, efectuează operațiuni de transfer de date si trimite comenzi altor componente din sistemul embedded. [11]

Memoria este folosită pentru a stoca datele pe care le primește procesorul și le folosește pentru a răspunde instrucțiunilor de programare. Aceasta poate fi de progam sau de date. [11]

Memoria de program (Program Memory) stochează pe termen lung rutinele pe care MCU le va executa după programare. Memoria de program nu este volatilă, ceea ce înseamnă că vă deține informații în timp fară a avea nevoie de o sursă de alimentare. [11]

Memoria de date (RAM-Random Access Memory) este necesară pentru stocarea temporară a datelor în timp ce instrucțiunile sunt executate. Spre deosebire de memoria de program, memoria de date este volatilă, ceea ce înseamnă că datele pe care le deține sunt temporare și sunt păstrate numai dacă dispozitivul este conectat la o sursă de alimtentare. [11]

Perifericele I/O – Dispozitivele de intrare și ieșire reprezintă calea prin care mcu-ul poate comunică cu aplicațiile din lumea exterioară. Porturile de intrare primesc informații și le trimit procesorului sub formă de date binare. Procesorul primește aceste date și trimite instrucțiunile necesare către dispozitivele externe care execută sarcini independent de MCU. [11]

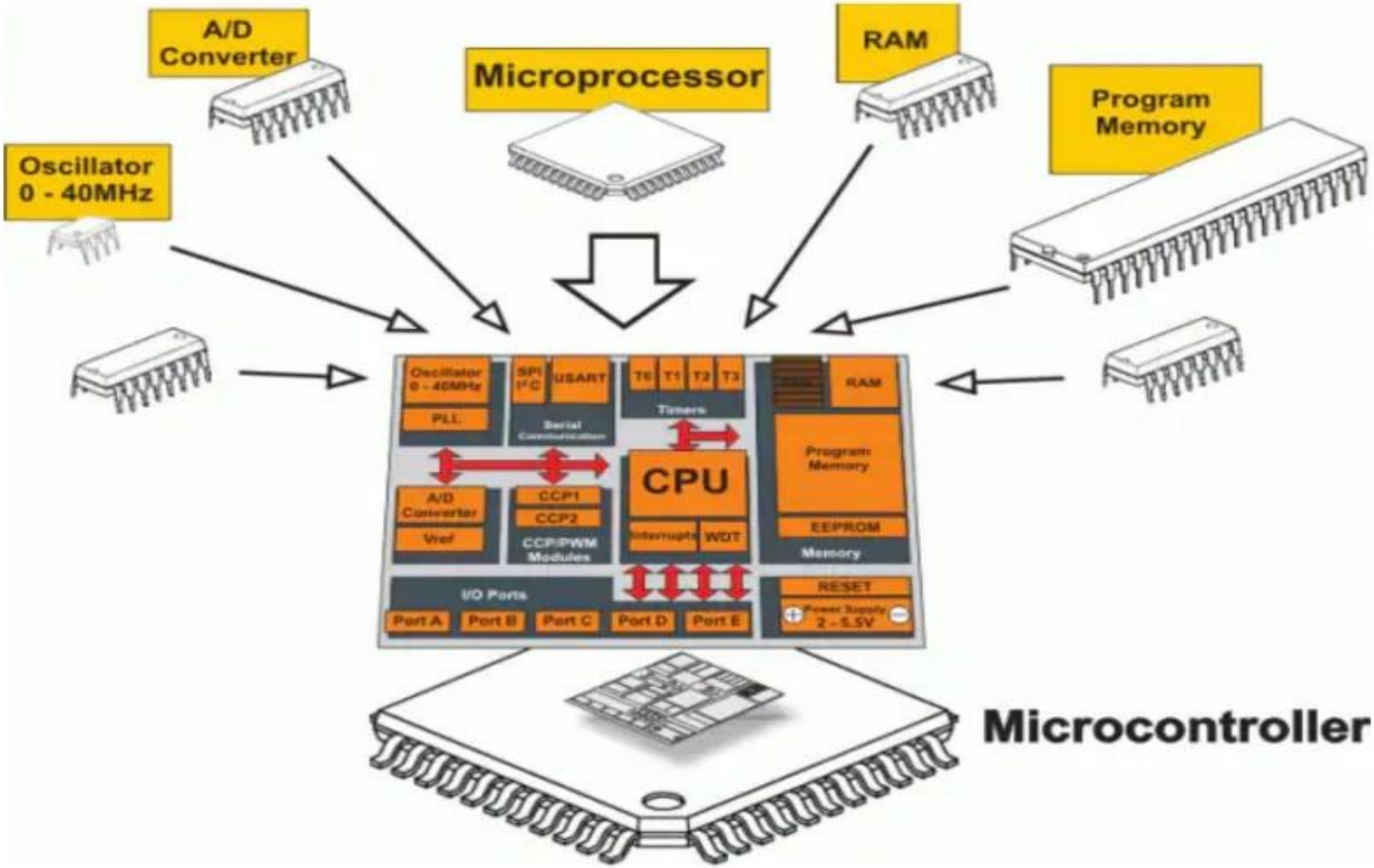


Figura 3: Structura generală a unui Microcontroller

1.6 DIGITAL TO ANALOG CONVERTER

Un DAC (Digital to Analog Converter) este un dispozitiv ce convertește semnalul digital într-un semnal analogic. Ieșirea analogică depinde foarte mult de referința atașată DAC-ului, ceea ce reprezintă factorul limită pentru aceasta. Unele DAC-uri au referință extenă pe când altele au referință internă, cele mai simple neavând niciuna din cele enumerate, referința fiind integrată în chipul de DAC. [1]

Există mai multe tipuri de DAC, ele deosebindu-se prin dimensiune, rezoluție, secvență maximă de prelevare a probelor și altele. [1]

DAC-urile sunt cel mai des folosite în industria muzicală pentru a converti semnal digital în semnale audio analogice. Mai sunt folosite și în televizoare, telefoane mobile pentru a converti un video ce este reprezentat digital, în semnale video analogice ce ajung la driver-ul de ecran pentru a putea afișa imaginea pe display. [1]

1.7 SPI

SPI (Serial Peripheral Interface) este o interfață ce permite schimbul de informații între două dispozitive. Modul de funcționare este „full duplex”, ceea ce înseamnă că datele pot fi transferate în ambele direcții în același timp, urmând o arhitectura de tip Master-Slave. SPI-ul este cel mai des folosit în sistemele embeded unde asigură comunicarea dintre CPU și dispozitivele periferice. De asemenea, este posibil să se conecteze două MCU-uri prin SPI. Inițial, termenul a fost creat de Motorola. [16]

Interfețele seriale au anumite avantaje față de interfețele paralele. Cel mai semnificativ avantaj este cablarea mai simplă, iar aceste cabluri pot fi mai lungi față de cablurile de la o intrefață paralelă. [16]

Multe tipuri de dispozitive pot fi controlate de un SPI, inclusiv cipuri de memorie, „port expanders” , drivere de afișare, convertoare de date, senzori, microprocesoare. [16]

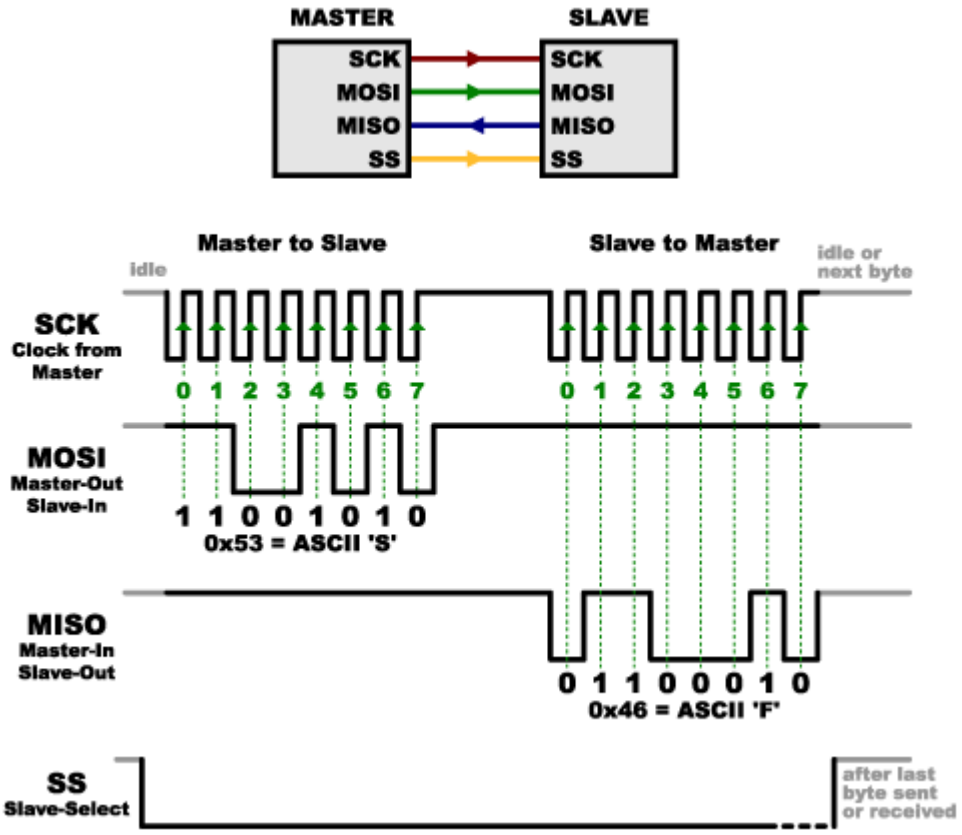


Figura 4: SPI

In *Figura 4* se poate observa cum sunt conectate si cum comunică un master si un slave.

SCLK - clock-ul este generat intodeauna de master.

MOSI - (Master Out Slave Input) – canalul prin care masterul comunică cu slave-ul

MISO - (Master Input Slave Output) – canalul prin care slave-ul comunică cu master-ul.

SS - (Slave Select) – Pinul prin care masterul poate accesa slave-ul.

Întotdeauna master-ul va fi cel care începe „conversația” generând semnale de clock pe pinul de SCLK și un LOGIC LOW pe pinul SS, indiferent dacă comunicarea se va face dintre Master către Slave sau invers.

1.7 UART

UART (Universal Asynchronous reciver-transmitter) este cel mai comun protocol utilizat pentru comunicarea în „full duplex” în care formatul și viteza de transmisie sunt configurabile. Se transmit date în mod asincron, iar datorită acestui fapt, nu există niciun semnal de „clock” pentru sincronizarea datelor. În locul acestui semnal, comunicarea pe UART se realizează prin adăugarea unui start-bit și stop-bit la pachetul de date transferat. Acești biți semnalează începutul și sfârșitul pachetului de date astfel încât dispozitivul, care primește datele prin intermediul UART, știe când să înceapă citirea. [19]

Figura 5: Exemplificarea transmisiei UART

Când dispozitivul ce primește date detectează un bit de start, acesta începe să le citească la o viteză prestabilită, numită rată de transfer (Baud Rate). Baud Rate-ul este o viteză de transfer a datelor exprimată în biți pe secundă (bps). Ambele dispozitive care comunică cu ajutorul UART-ului trebuie să funcționeze la aceeași rată de transfer.[19]

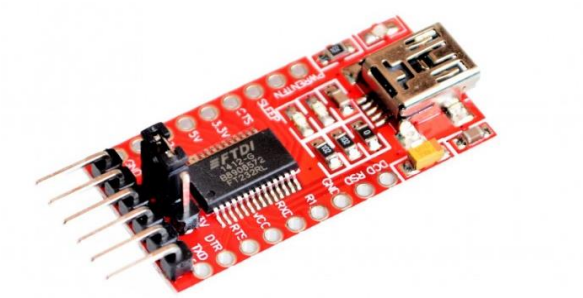


Figura 6: Convertor de serială

Pentru ca UART-ul să fie interfațat cu PC-ul există necesitatea introducerii un convertor de serială. Acesta va face posibilă comunicarea între MCU și PC.

2. ALEGEREA COMPONENTELOR

Pentru prezenta lucrare de licență, au fost alese diferite componente în urma analizei unor mai multor platforme de documentație online.

2.1 Microcontroller

Figura 7: Microcontroller PIC24FJ1024GA610

PIC24FJ1024GA610 este un Microcontroler dezvoltat de firma Microchip ce este deja integrat în placa de dezvoltare Clicker 2 realizată de MikroElektronika. Datorită componentelor esențiale funcționarii deja existente și lipite pe placa de dezvoltare, utilizarea MCU-ului este mult mai ușoară.

Specificațiile necesare pentru acest proiect regăsite în datasheet-ul MCU-ului (v.[12]) sunt: posibilitatea de a comunica pe doua canale SPI

- posibilitatea de a comunica pe un canal UART
- memorie de minim 32Kb RAM, suficient pentru memorarea unei poze de 50x50 px
- servicii de întrerupere
- analog to digital converter (ADC)

2.2 Laser

Laserul ales este dezvoltat și vândut de compania Laserlans. Are o putere de 100mW, tensiunea necesară de alimentare fiind de 12 V. Unda laser este de culoare roșie, lățimea ei fiind de 650nm.

2.2 Sistemul galvo

System galvo este dezvoltat și vândut de LH-LASER. Are o viteză de 18Kpps (Kilo points per second) cu o deviere optică a oglinzii de 20 grade. Inputul pe fiecare motor este de +-5V, fiind unul foarte sensibil.

2.3 DAC

LTC2601 este un DAC dezvoltat de Linear Tehnology și este integrat în DAC2CLICK- un dispozitiv care poate fi atașat la placa de dezvoltare Clicker 2. Conform datasheet-ului (v. [10]), acest DAC are o rezoluție de 16 biți, ceea ce înseamnă că poate reprezenta tensiunea de output din intervalul 0V-5V cu o acuratețe de 2^{16}. Deoarece proiectul are în componenta două motoare galvo, sunt necesare două astfel de DAC-uri, câte unul pentru fiecare motor. Referința acestui DAC este de 5V.

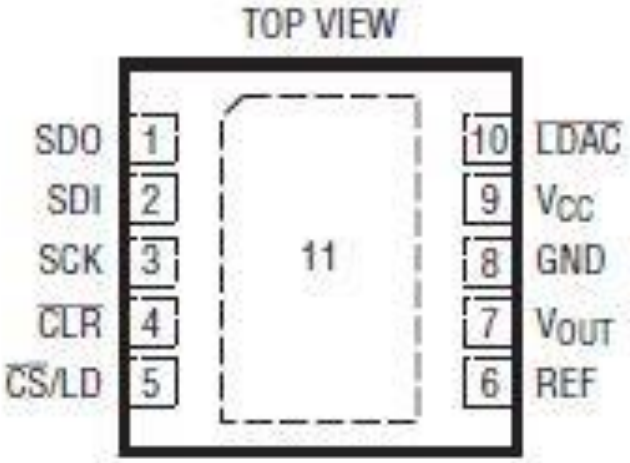


Figura 9: DAC pins configuration

Formula pentru calculul tensiunii de ieșire este: $V_{OUT(IDEAL)} = \left(\frac{k}{2^N}\right) V_{REF}$

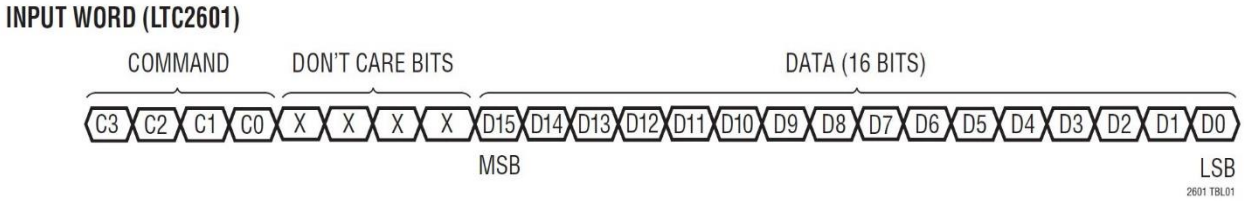
K - valoarea decimală echivalentă, primită prin SPI.

N - rezoluția DAC-ului

Vref - voltajul corespunzător referinței

Vout – tensiunea dorită pentru output.

Comunicarea dintre DAC si MCU se face prin SPI. Cuvantul care este primit de la MCU este format din 24 de biți, după cum urmează:



Cuvant-24biti

COMMAND – format din 4 biți, comenzile fiind reprezentate in tabelul cu comenzi.

DON'T CARE BITS – 4 biți ce pot fi puși pe 0 sau pe 1, nu afectează cuvântul in sine.

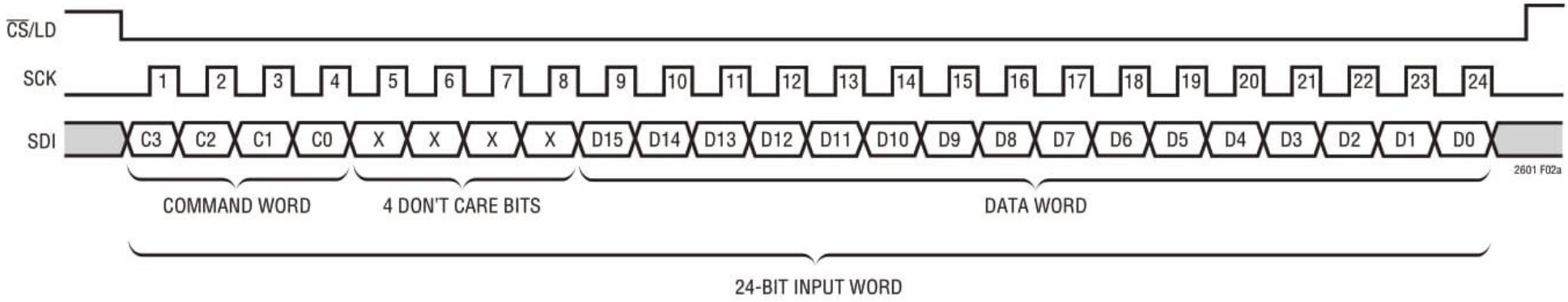
DATA – valoarea K pentru formula de calcul a tensiunii de ieșire.

Table 1.

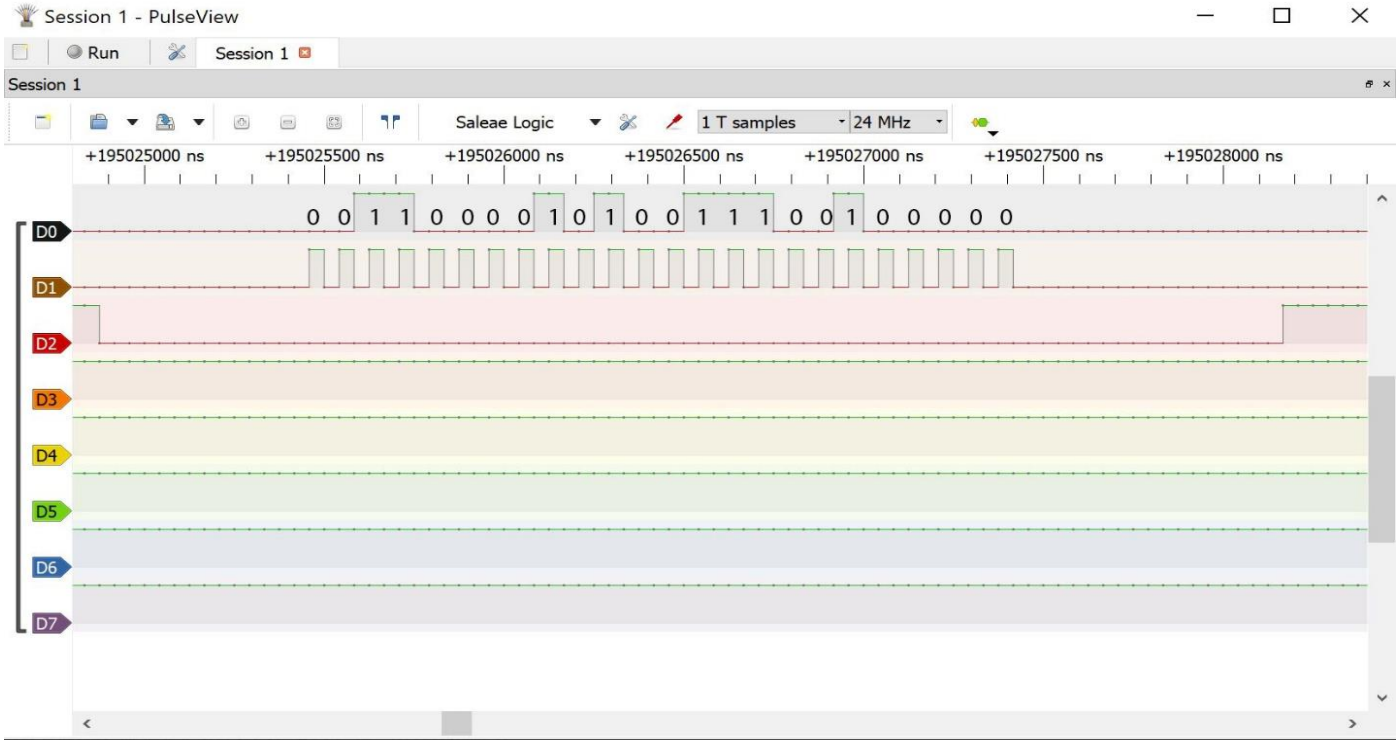
COMMAND*				
C3	C2	C1	C0	
0	0	0	0	Write to Input Register
0	0	0	1	Update (Power Up) DAC Register
0	0	1	1	Write to and Update (Power Up)
0	1	0	0	Power Down
1	1	1	1	No Operation

*Command codes not shown are reserved and should not be used.

Tabel cu comenzile DAC



Schema transmisiei unui cuvânt către DAC



Exemplificarea cu ajutorul unui Logic Analyzer a unui cuvânt transmis 0x30 0xA7 0x20

3. FUNCȚIONAREA SISTEMULUI

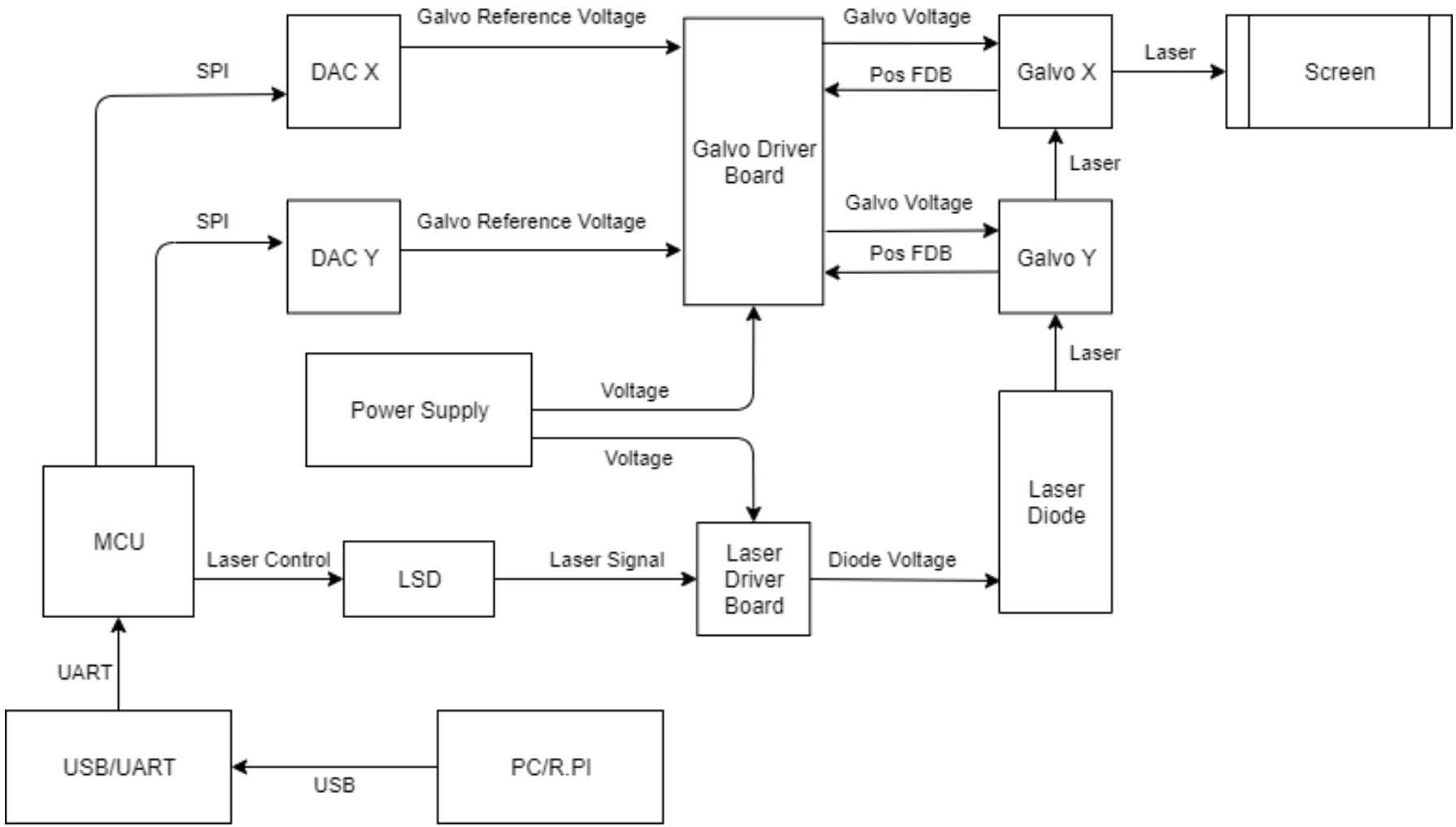


Figura 12: Schema generală a sistemului

Rolul componentelor existente in *Figura 12* este:

1) Power Supply – Transformă 220V AC in +12V -12V +5V DC, furnizând astfel tensiunea necesară pentru laser driver și galvo driver board.
2) Laser Driver Board – Controlează dioda laser care este foarte sensibilă la curent și temperatură, primind feedback direct de la diodă.
3) Galvo Driver– Functionează pe baza voltajului primit de la Power Supply. Preia semnalele analogice de la cele două DAC-uri și comunică cu cele două motoare galvo.
4) Galvo Motors (Galvo X, Galvo Y) – fiecare motor conține câte un senzor capacitiv ce transmite feedback către driverboard despre poziția oglinzii.
5) Dioda Laser – este indreptată spre oglinzile motoarelor galvo.
6) MCU – Preia datele de la PC prin UART și comunică cu cele două DAC-uri prin SPI pentru a controla motoarele galvo.
7) DAC (DAC X, DAC Y) – Preia comenzile de la MCU prin SPI și transmite către driverboard valorile analogice respective.
8) LSD – Un tranzistor ce permite închiderea sau deschiderea diodei cu ajutorul semnalului primit de la MCU.
9) USB/UART – convertor de serială ce face face posibilă comunicarea dintre MCU si PC/R.PI.
10) PC/R.PI – prelucreaza imaginea si transmite date catre MCU.
11) Screen – suprafata nereflectivă suficient de mare pentru imaginea proiectată cu laserul.

4. DEZVOLTAREA PROIECTULUI

Am ales această tema pentru a face cercetare, dezvoltare și inovație într-un domeniu de actualitate (Embedded), pentru a obține un concept de urmat în viitor. Ideea mi-a venit după ce am participat la un festival unde erau prezente numeroase jocuri de lumini, printre care și proiecții laser de diferite forme.

Analizând piața am aflat că acestea sunt proiectoare ce folosesc sistem galvo cu oglinzi ce pot proiecta laserul în diferite forme. Acestea funcționează pe baza tensiunii analogice primite de la o sursă externă.

La o simplă căutare pe internet, am ajuns la concluzia că aceste echipamente sunt scumpe, sub ordinul miilor de dolari, dar limitând puterea și performanța se pot găsi lucruri rezonabile. Astfel, am putut comanda kitul galvo (2xmotoare galvo cu oglindă, driver motoare,sursa de alimentare) , împreună cu o dioda laser roșie de pe o platformă online chinezească.

Montarea acestor echipamente nu a fost ușoară datorită faptului că laserul trebuie să fie perfect aliniat cu oglinzile pentru ca reflexia să fie una uniformă. Ca și suport am folosit o placă de pal de 25x40 cm, iar ca materiale de fixare am folosit șuruburi și piulițe rămase de la alte proiecte personale. Cablajul echipamentelor l-am realizat conform instrucțiunilor primite de la vânzător.

După ce am codat driver-ul de DAC și driver-ul de serială UART, pentru a-l testa am folosit funcția *void DAC_Send_Data* ce oferă posibilitatea de a trimite comanda de pe seriala UART, iar utilizând convertorul am putut să scriu comenzi de la tastatură in terminalul Hterm. Astfel, am obținut tensiuni de ieșire intre 0-5V, controlând dioda laser pe suprafața reflectată.

Pentru a putea da comandă motoarelor, am folosit placa de dezvoltare CLICKER2 care interfațează un microcontroller PIC24FJ1024GA610 ce are o arhitectură de 16biti și cele două „add-on”-uri DAC2CLICK care interfațează câte un DAC LTC2601 cu o rezoluție de 16biti.

Cea mai bună reprezentare a unei imagini, în cazul de față, este de a o reprezenta printr-o matrice de puncte ale căror coordonate să fie reprezentate prin valori din intervalul 0-65535 (rezoluția DAC).

După mai multe încercări am ales ca matricea mea să fie alcătuită 50x50 puncte, adică 250 de pixeli în total. Pentru reprezentarea unui pixel am folosit o structură de date Ram_Pixel_Array ce conține 3 tipuri de date: prima reprezintă diferența dintre oricare două puncte active consecutive, a doua este valoarea coordonatei de pe axa OX, iar a treia este valoarea coordonatei de pe axa OY..

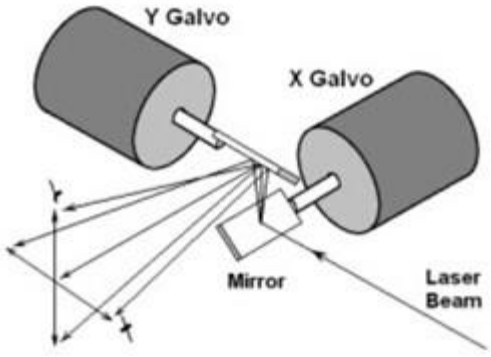


Figura 13: Galvo motors

După crearea matricii și scrierea ei într-un fișier din proiect cu ajutorul aplicației python, am parcurs-o in direcția sus-jos, stânga-dreapta trimițând cont de comenzile aferente coordonatelor către DAC. În Figura 14 se observă rezultatul parcurgerii.

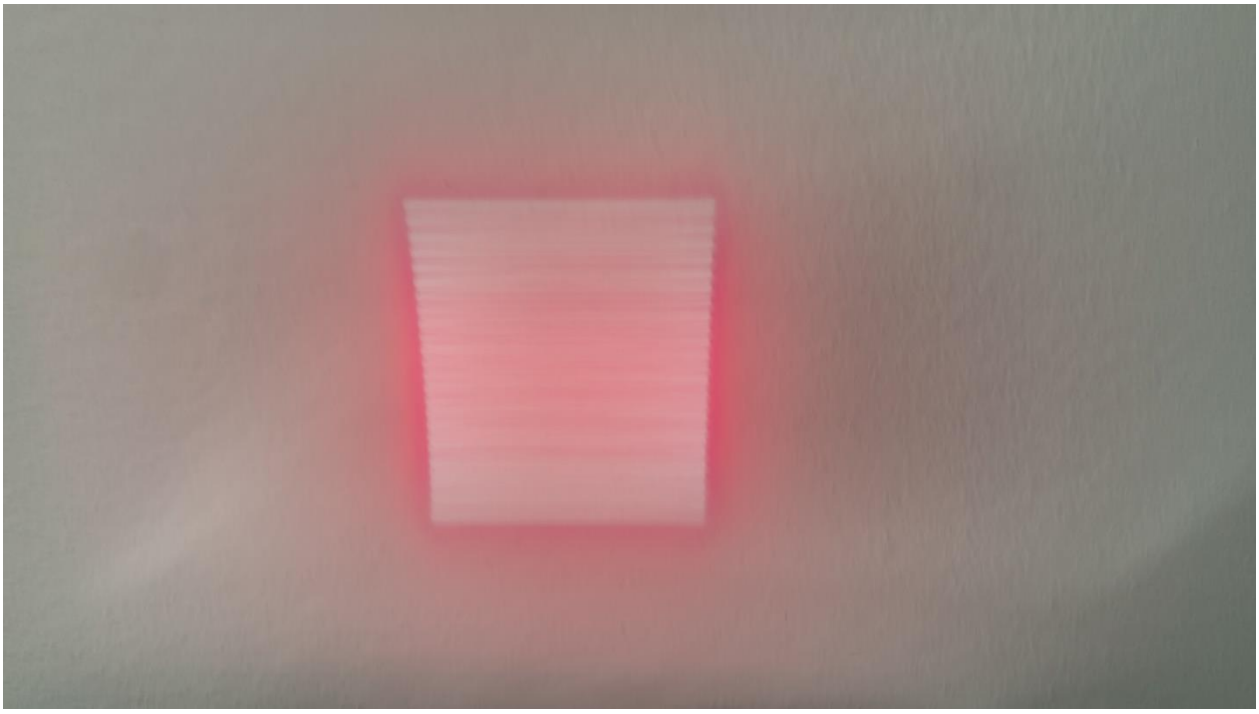


Figura 14: Parcurgerea matricei cu laserul

Pentru a eficientiza parcurgerea am ales să dezvolt aplicația scrisa in limbajul python, astfel încât, după ce am transformat imaginea într-o matrice de puncte, am parcurs în aceeași ordine ca mai devreme și am extras acele puncte care sunt „active” (pixel diferit de alb), cu scopul de a transpune coordonatele acestora într-un array de pixeli activi. Astfel laserul va „urmări” strict pixelii negrii.

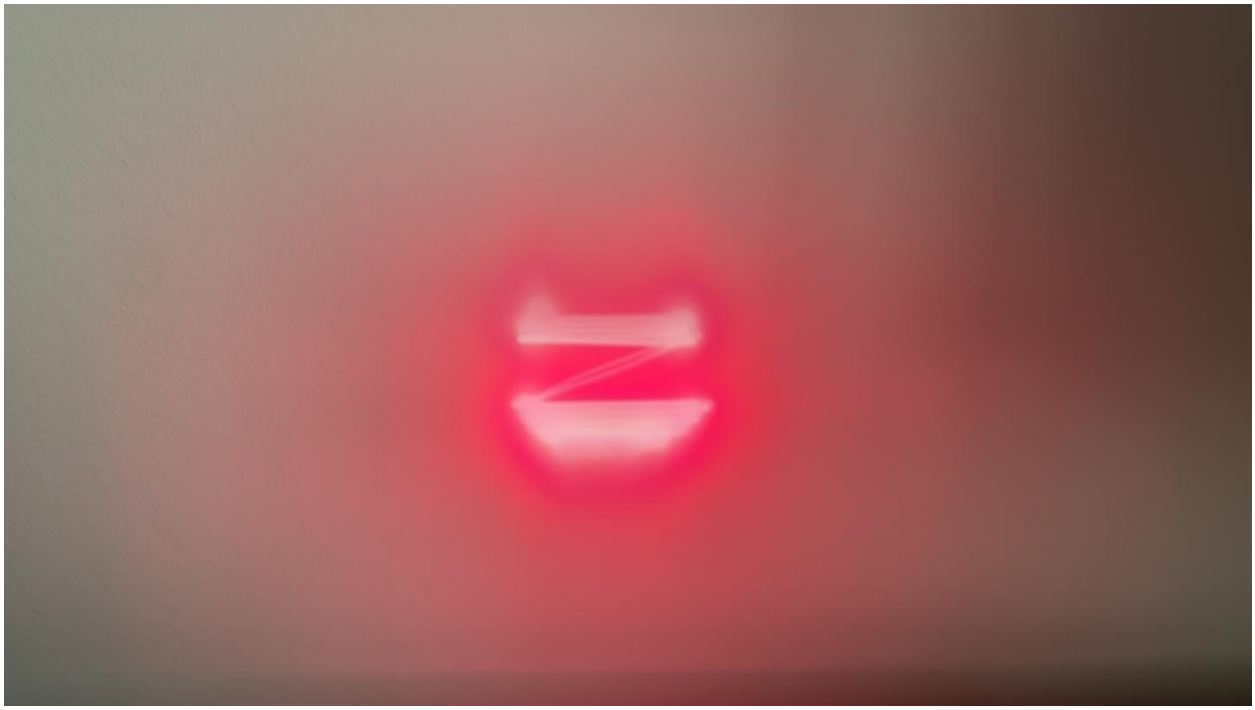


Figura 15: Parcurgerea pixelilor activi

O problemă ce intervine în aceasta modalitate este că dioda nu se stinge niciodată, iar când laserul trece de la un pixel aprins la unul stins va lumina pixeli nedoriți, lansând o urmă nedorită de lumină. Pentru a rezolva aceasta problemă am folosit propietatea de întrerupere a MCU-ului, setând să se declanșeze câte o întrerupere după fiecare coordonată trimisă pe SPI de MCU către DAC. Astfel, voi putea controla dioda laser imediat ce motoarele au primit coordonatele pentru noul pixel, modificând valoarea pinului ce duce la LSD care mai apoi controlează dioda laser.

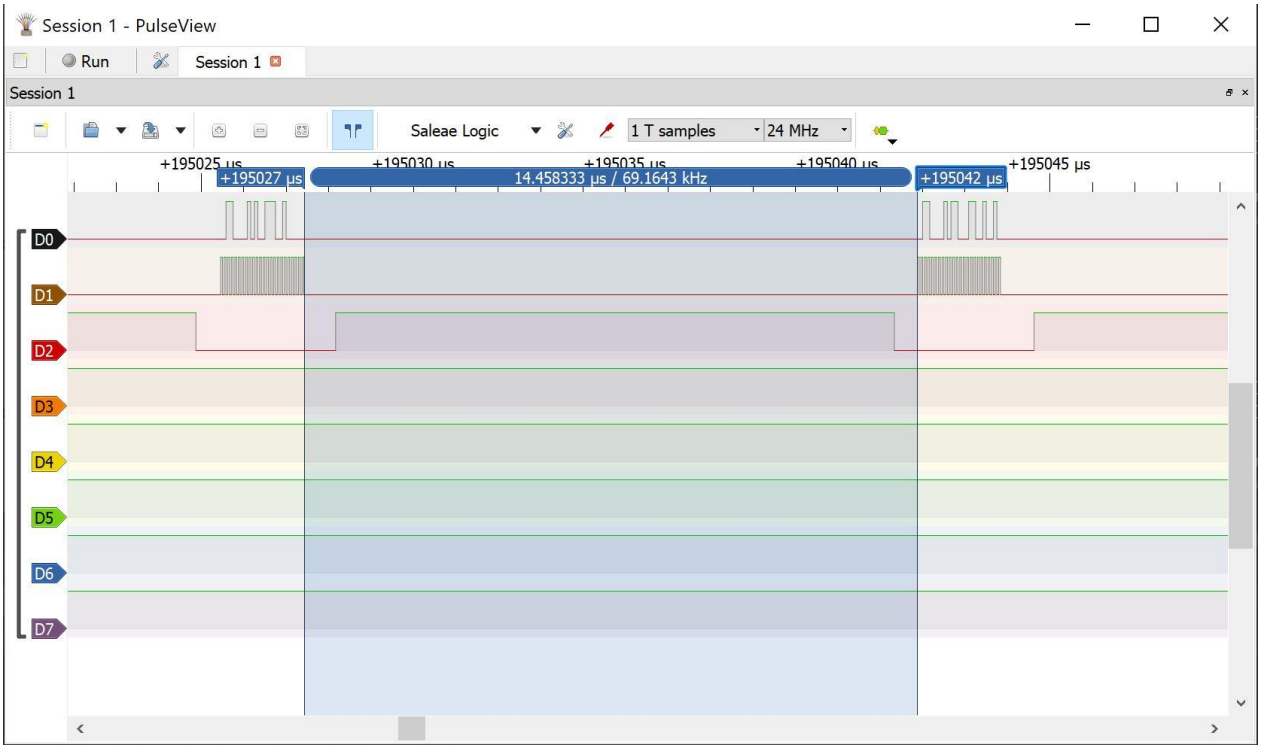


Figura 16: Distanta dintre două cuvinte consecutive trimise pe SPI

După cum se vede și în Figura 16, timpul dintre două transmisii consecutive pe SPI este de 14,5us, la care se adaugă timpul de delay dintre transmisiile pe spi, în acest timp dioda laser trebuie stisă și reaprinsă.

Pentru a putea măsura acest timp am creat modulul de Timer1 care poate fi configurat pentru intervalul de timp descris mai sus. Pentru a face acest lucru variabil, am implementat și modulul de ADC care poate interpreta tensiunea dată de potențiometru într-o valoare numerică. Această valoare se conține într-o variabilă și este modificată de utilizator cu ajutorul potențiometrului modificând timpul în care dioda stă aprinsă între două transmisii pe SPI.

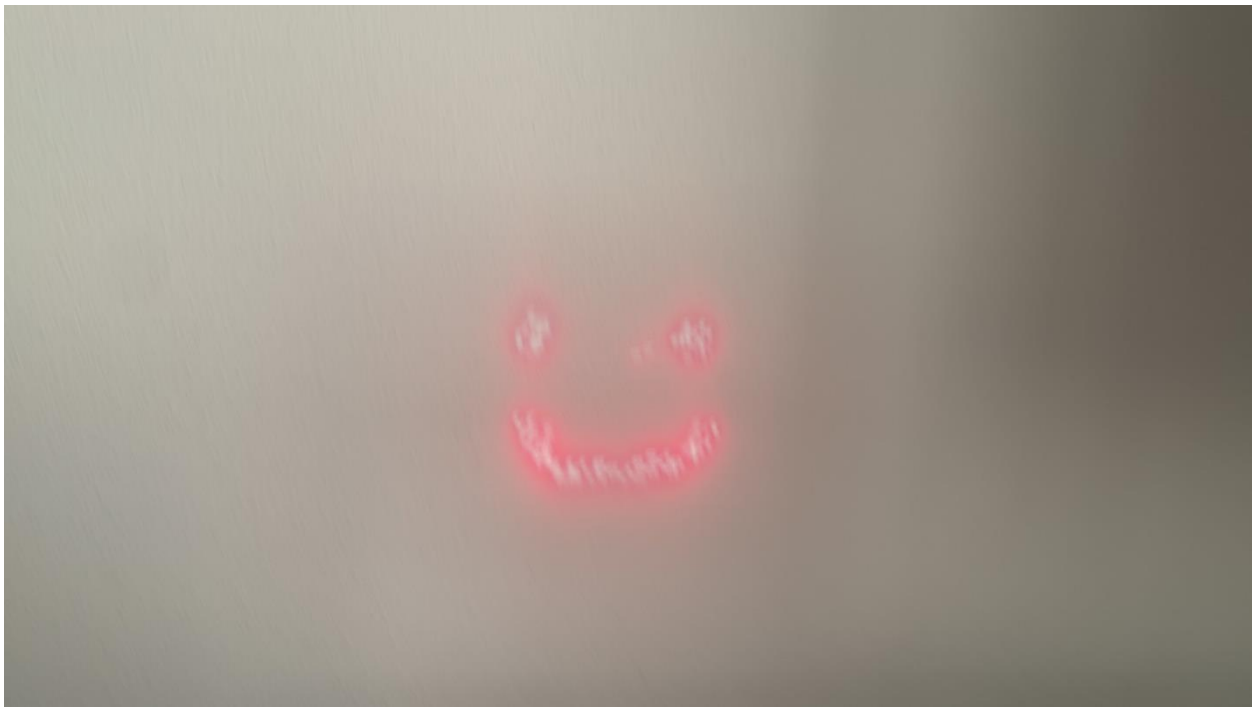


Figura 17: Parcurgerea pixelilor activi și stingerea diodei cu ajutorul întreruperilor

Reglând potențiometrul, implicit perioada de timp în care dioda stă aprinsă între două transmisii, se poate ajunge la o imagine clară și nedistorsionată.

5. IMPLEMENTARE

5.1 Tehnologii folosite

5.1.1 Embedded

SPI (Serial Peripheral Interface) asigură comunicarea dintre MCU și cele două DAC-uri. În cazul de față, SPI-ul este configurat să lucreze la o viteză de 4 MHz, adică maximul posibil pentru configurația proiectului. Pentru mai multe detalii despre modul general de funcționare se poate consulta capitlolul „Noțiuni de bază", subcapitolul SPI.

UART (Universal Asynchronous Receiver-Transmitter) asigură comunicarea dintre PC/R.Pi pentru transmiterea coordonatelor pixelilor și pentru debug. Este configurat să lucreze cu un baud-rate de 11520. Pentru mai multe detalii despre modul general de funcționare se poate consulta capitlolul „Noțiuni de bază", subcapitolul UART.

GPIO (General Purpose Input/Output) asigură configurarea direcției pinilor MCU-ului, astfel încât, să putem scrie valoarea pinilor cu 1 sau 0 logic sau să putem citi valoarea unui pin ce este comandat din exterior cu 1 sau 0 logic.

Întreruperile sunt evenimente ce sunt declanșate separat față de rutina de execuție a MCU-ului, cu ajutorul acestora se poate realiza multitasking-ul într-un sistem cu un singur procesor și un singur set de resurse hardware și resurse software. În acest proiect, întreruperile sunt folosite pentru a detecta diferite eventimente și de a executa cod după apariția acestor evenimente, cum ar fi terminarea transmisiei unui cuvânt pe SPI sau expirarea unui timer.

Limbajul de programare C este limbajul folosit in mediul de lucru MPLAB X, folosind compilatorul XC16.

5.1.2 High Level

Limbajul de programare Python este un limbaj relativ nou, ușor de adaptat cerințelor aplicației. Codul acestui limbaj este unul simplu de citit și de urmărit. Acest limbaj permite programatorului să-și exprime ideile în mai puitine linii de cod, fară a reduce din lizibilitate. De asemenea, Python este un limbaj multi-paradigmă, ce suportă paradigme de programare OOP și

management automat al memoriei. De asemenea, limbajul Python are o multitudine de librării open-source bine documentate.

OpenCV (Open source computer vision) este o librărie disponibilă pe mai multe platforme, aceasta fiind una dintre cele mai uzuale librarii pentru procesarea imaginilor. Această librărie are o structura simplă și lizibillă, fiind ușor de folosit în aplicații.

Serial este o librărie care face posibil accesul la portul serial și este folosită pentru a realiza comunicarea dintre PC și MCU, mai precis, pentru a transmite array-ul de pixeli din memoria PC-ului către memoria MCU-ului.

Struct este o librărie pentru manipularea datelor binare, aceasta fiind folosită pentru a pregăti datele pentru transmiterea prin seriala usb.

5.2 Detalii de implementare

Pentru dezvoltarea proiectului, am utilizat tehnologiile prezentate în capitolul anterior, folosind că mediu de lucru MPLAB X pentru programarea MCU-ului și PyCharm pentru programarea aplicației de procesare a imaginii.

5.2.1 MPLAB X

MPLAB X(v5.30) este un mediu de dezvoltare care asigură programarea MCU-urilor Microchip, cu ajutorul unui programator cu cablu USB, în cazul de față PICKit3. Acesta realizează funcția de programare a memoriei MCU-ului și funcția de debug folosind breakpoint-uri.

Dezvoltarea programelor se face cu ajutorul limbajului C, folosind unul din compilatoarele oferite de Microchip, în funcție de arhitectura MCU-ului 8, 16, 32 biți. În cazul de față, se va folosi XC16 datorită faptului că MCU-ul ales are o arhitectură de 16biti.

5.2.2 PyCharm

Pycharm(v2019.2.5) este un mediu de dezvolare multi-platformă care asigură programarea aplicațiilor în limbajul Python. Acesta facilitează o ușoară instalare a librăriilor open-source folosite în proiect, având totodată un meniu intuitiv și ușor de urmărit. Interpretorul de

proiect folosit este Python 3.8, fiind ultima versiune de python disponibilă la momentul dezvoltării proiectului.

5.3 Embedded Software

Partea de Embedded este împărțită pe mai multe module după cum urmeaza:

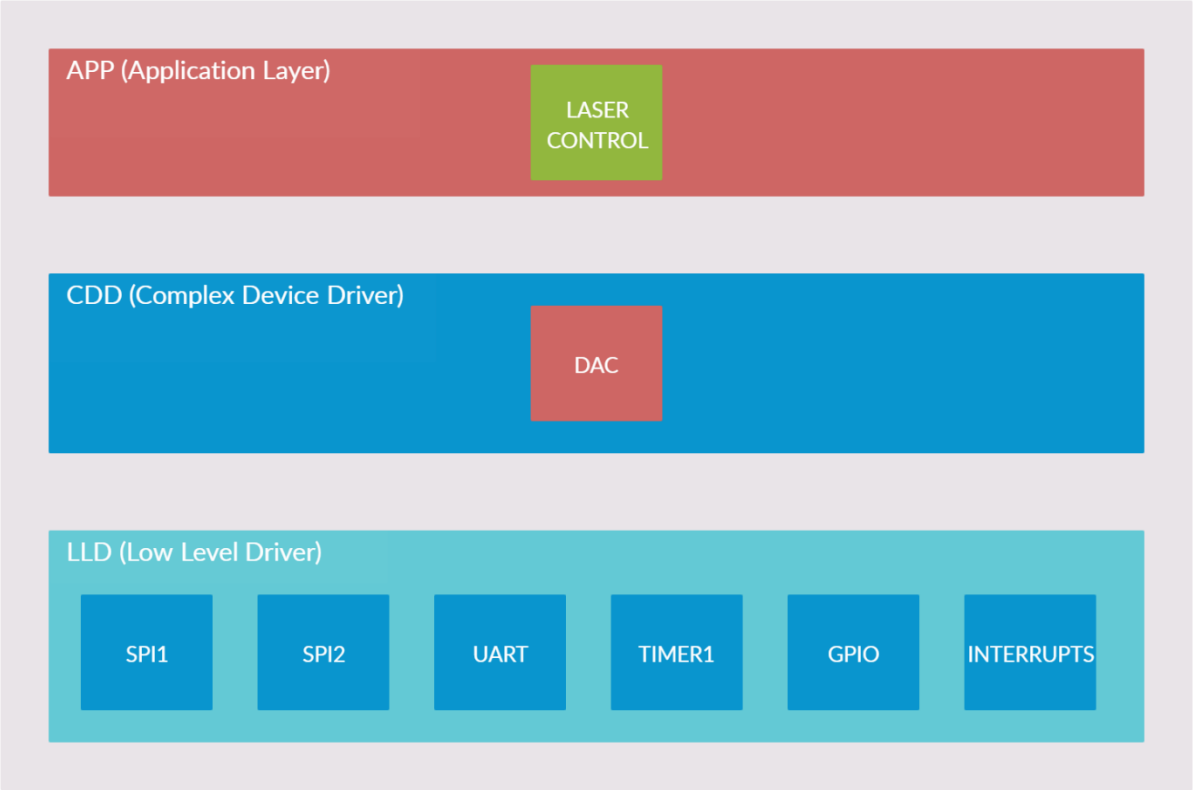


Figura 18: Diagrama arhitecturii software

5.3.1 Modulul SPI

Din datasheet-ul MCU-ului, SPI poate fi configurat și folosit cu ajutorul regiștrilor:

- SPIxCON1L, SPIxCON1H, SPIxCON2L - regiștrii de configurare a funcționarii SPI-ului.
- SPIxSTATL, SPIxSTATH - regiștrii de status, conțin o serie de flag-uri.
- SPIxBUFL SPIxBUFH - regiștrii de tip buffer pentru transmiterea datelor.
- SPIxIMSKL,SPIxIMSKH - regiștrii care configurează întreruperile din modulul SPI
- SPIxURDTL, SPIxURDTH – regiștrii care țin date pentru transmitere in cazul unei condiții de „Underrun”.

Modulul SPI (spi.c, spi.h) conține driverul de SPI, fiind alcătuit dintr-o funcție de inițializare și mai multe funcții de trimitere a informației, făcându-se astfel posibilă comunicarea dintre MCU și cele două DAC-uri.

În modul de funcționare normală, masterul SPI controlează generarea clock-ului serial. Numărul de impulsuri de clock corespund mărimii datelor de transfer: 2-32 biți.

Conform procedurii din datasheet-ul MCU-ului, configurarea SPI-ului în modul Master se va efectua folosind următorii pași:

1. Dezactivarea și resetarea întregului modul (SPI1CON1Lbits.SPIEN = 0).
2. Dezactivarea întreruperilor din registrul IECx corespunzător. (IEC0bits.SPI1TXIE=0)
3. Curățarea buffer-ului de recepție (SPI1BRGL = 0x00)
4. Curățarea buffer-ului ENHBUF din registrul SPIxCON1L, pentru a folosi bufferul standard.
5. Dacă folosim întreruperi, va trebui să facem următorii pași:
6. Curățarea flagurilor folosite din regiștrii IFSx respectivi.
7. Scrierea priorității întreruperii din regiștrii IPCx respectivi.
8. Pornirea întreruperilor prin setarea registrului IECx respectiv
9. Scrierea registrului Boud Rate SPIxBRGL
10. 6. Curățarea bit-ului SPRIROV din registrul SPIxSTATL
11. Selectare Master-mode cu ajutorului bit-ului MSTEN din registrul SPIxCONL
12. Activarea modulului SPI scriind bitul SPIEN din registrul SPI1CON1L
13. Scrierea in buffer a datelor care vrem sa fie transmise SPIxBUFL/H.

Funcțiile void *SPI1_Init_24bit_Interrupt()* și *void SPI2_Init_24bit_Interrupt()* sunt construite pentru a executa pașii 1-9.

În plus, aceste funcții configurează mărimea cuvântului trimis, cu ajutorulul registrului SPI1CON2L, setând biții WLENGTH cu valoarea 0x17, adică cuvântul va avea o lungime de 24biti. De asemenea, bitul CKE din registrul SPIxCON1L este setat pe 1 deoarece DAC-ul necesită ca începutul trasmiterii să reprezinte o tranziție a clock-ului din starea active în starea idle. Întreruperea este configurată să apară atunci când un cuvânt este transmis, motivul acestei configurații va fi explicat la modulul de timer.

Cele două buffere de date ale spi-ului SPIxBUFL și SPIxBUFH sunt alcătuite din 16 biți fiecare, dar datorită faptului că lungimea cuvântului are 24biti vom folosi toți cei 16 biți ai buffer-ului SPIxBUFL și doar 8 biți din SPIxBUFH.

Funcțiile *void SPI1_Write_24bit_(unsigned int dataH, unsigned int dataL)* și *void SPI1_Write_24bit(unsigned int dataH, unsigned int dataL)* execută pasul 10, adică primesc ca parametru dataH(8 biți) și dataL(16 biți) reprezentând cele două bucăți de cuvânt ce vor fi transmise pe SPI. După încărcarea datelor în cele două buffere (SPIxBUFL și SPIxBUFH), se interoghează flag-ul de SPIRBF (Recive Buffer Status bit) pentru a se asigura că transferul a fost efectuat înainte de a trece mai departe.

5.3.2 Modulul UART

Folosind datasheet-ul (v.[20]), se poate trage concluzia că modulul UART (uart.c, uart.h) poate fi configurat și folosit cu ajutorul regiștrilor:

- UxMODE – registru de configurare
- UxSTA – registru de status și control
- UxRXREG – registru de primire a datelor
- UxTXREG – registru de trasmitere a datelor

Pentru a configura modulul UART pentru transmiterea pachetelor de date a câte 8biti, trebuie realizați următorii pași:

1. Scrierea valorilor pentru lungimea pachetului de date, paritate și „stop bits”.
2. Scriem valoarea pentru „boud rate” în registrul UxBRG.
3. Activarea întreruperilor pentru primire și trimitere a datelor din registrul aferent (U1MODEbits.UARTXEN= 1)
4. Activarea modulului UART
5. Activarea modulului de trimitere a datelor (UTXEN)
6. Transmiterea datelor se va realiza scriind în registrul UxTXREG.
7. Primirea datelor se va realiza citind registrul UxRXREG.

Funcția *void UART_Init()* realizează pașii 1-5, configurând modulul UART să conțină 1-stop bit, transmiterea să fie realizată în 8 biți. Valoarea pentru a configura viteza de transfer se va scrie în registrul UxBRG, folosind următoarea formulă de calcul:

$$UxBRG = \frac{F_p}{16 \times Baud\ Rate} - 1$$

F_p = Instruction cycle clock frequency (FOSC/2)

$Baud\ Rate$ = frecvența cuvintelor

Funcția *void UART1_Write_Char(char ch)* primește ca parametru o valoare de 8biti care urmează să fie transmisă pe UART. Înainte de a se transmite se va interoga bitul UTXBF din registrul de status U1STA pentru a verifica dacă bufferul este gol, însemnand că este gata de a primi date pentru a putea fi transmise.

Funcția *char UART1_Get_Char()* așteaptă fixarea flagului de semnalare a primirii unui caracter pe serială (U1RXIF) și apoi returnează valoarea de 8 biți primită.

Funcția *unsigned int UART1_Get_2Bytes()* returnează o valoare acumulată din 2 caractere primite pe serială, deci o valoare de 16 biți.

5.2.3 Modulul TIMER1

Modulul Timer1 este un timer de 16 biți care poate fi folosit ca RTC (Real-Time Clock) sau poate fi operat separat ca un timer obișnuit.

Conform datasheet-ului (v.[13]), modulul de Timer1 poate fi configurat și folosit cu ajutorul registrului:

- T1CON – registru de configurare

Pentru a configura modulul de Timer1 trebuie realizați următorii pași:

1. Dezactivarea modulului prin bitul TON al registrului T1CON
2. Selectarea prescaller-ul dorit.
3. Selectarea sursei de clock.
4. Scrierea valorii dorite pentru perioada timer-ului in registrul PR1.

5. Setarea priorității întreruperii .
6. Activarea întreruperii.
7. Activarea întregului modul.

Funcția *void T1_Init()* realizează toți pașii descriși mai sus. Pentru a calcula valoarea care se va scrie în registrul PR1, ce reprezintă perioada timer-ului, se va folosi următoarea formulă:

$$PR1 = \frac{F_p}{F_i}$$

F_p = Instruction cycle clock frequency (FOSC/2)

F_i = Frecvența dorită a întreruperii

5.3.4 Modulul PPS (Peripheral Pin Select)

Funcția PPS constă în setarea funcționalității unui pin de către programator. Acest lucru duce la o mai bună organizare a proiectului, datorită faptului că acești pini nu au o funcționalitate prestabilită. Perifericele gestionate de PPS sunt toate digitale. Acestea includ SPI, UART, Timer, întreruperi externe.

Funcția *void Port_Init()* configurează funcționalitatea pinilor pentru întreg proiectul. Astfel, vom avea:

Pin	Funcționalitate
RD9	U1RX
RD8	U1TX
RD2	SDO1
RD3	SCK1
RB4	SDO2
RB5	SCK2

Tabelul pinilor configurabili (PPS)

PIN	Output Set Value
RE4	1
RA10	1
RE3	1
RA9	1
RD0	1

Tabelul pinilor GPIO

5.3.5 Modulul DAC

Conform datasheet-ului(v.[10]), pentru ca DAC-ul să funcționeze conform așteptărilor proiectului, va trebui realizată următoarea configurație a pinilor:

PIN NR	PIN	MCU PIN
1.	SDO	SDI
2.	SDI	SDO
3.	SCK	SCK
4.	CLR	LOGIC HIGH
5.	CS	CS
6.	REF	VCC
7.	Vout	-
8.	GND	GND
9.	VCC	VCC
10.	LDAC	LOGIC LOW

Configurarea pinilor pentru DAC

Datorită faptului că modulul de SPI al MCU-ul are doi regiștrii a câte 16 biți fiecare, nu putem să punem întreg cuvântul(24biti) într-un singur registru, așadar o să se separe într-un cuvânt de 16 biți și unul de 8 biți.

Funcția *unsigned int Make_Low_Uart_Command_Dac(unsigned char byte1, unsigned char byte2, unsigned char byte3)* primește ca parametru întregul cuvânt dorit de transmis pe SPI către DAC împărțit în 3 bytes (3x8biti=24biti) și returnează o valoare de 16 biți, care reprezintă partea din dreapta a cuvântului.

Funcția *unsigned int Make_High_Uart_Command_Dac(unsigned char byte1, unsigned char byte2, unsigned char byte3)* primește aceleași argumente ca în funcția de mai sus, dar de data aceasta returnează o valoare de 16 biți în care doar primii 8biti sunt completați.

Funcția *void DAC1_Send(unsigned int dataH,unsigned int dataL)* și *void DAC2_Send(unsigned int dataH, unsigned int dataL)* trimit datele primite ca argument pe serială apelând funcția descrisă de la modulul SPI (*SPI1_Write_24bit* respectiv *SPI2_Write_24bit*) acționând un LOGIC LOW si un LOGIC HIGH pe pinul de CS al DAC-urilor, înainte de a transmite si,respectiv, la terminarea transmiterii datelor.

Funcția *void DAC_Send_Data(unsigned int data)* primește ca argument întreg cuvântul care urmează a fi trimis pe SPI și trimite cuvântul celor doua DAC-uri.

5.3.5.1 Functii pentru debug

Funcția *void Uart_Spi_Dac()* preia 3 bytes de pe seriala UART și cu ajutorul lor, pregătește cele doua bucăți de cuvânt pentru a fi transmise pe serială.

Funcția *void DAC_UART_Control()* transmite cele doua bucăți de cuvânt primite mai devreme către cele două DAC-uri. (aceeași valoare pentru ambele)

Funcția *void DAC_Send_Data(unsigned int data)* primește ca argument întreg cuvântul, urmând a fi trimis pe SPI și trimite cuvântul celor două DAC-uri.

Funcția *void DAC_Data_Increment()* transmite întregul interval de valori de ieșire posibile pentru DAC, cu o incrementare de 10.

5.3.6 Laser Control

Modulul Laser Control (laser_control.c, laser_control.h) se ocupă cu stocarea imaginii în memoria RAM și cu trimiterea de comenzi pentru controlul laserului.

Reprezentarea unei imaginii se face cu ajutorul unei structuri de date *PixelData_Type*. Fiecare element din această structură este reprezentat prin 3 valori:

- *unsigned char Laser_Diode_State* – distanța dintre doi pixeli consecutivi
- *unsigned int Pixel_Coord_Motor_X* – coordonata unui pixel pe axa OX

- *unsigned int Pixel_Coord_Motor_Y* – coordoata unui pixel pe axa OY

Astfel, pentru a reprezenta o imagine în memorie, se vor primi coordonatele pixelilor prin seriala UART și se va memora în RAM.

Fuctia *void Get_Array()* așteaptă primirea numărului de pixeli activi, iar în funcție de acest număr va salva pixelii respectivi în structura de date declarată global *Ram_Pixel_Array.*

Funcția *void Draw_Ram_Array()* parcurge de la stânga la dreapta și de la dreapta la stânga array-ul de pixeli *Ram_Pixel_Array* și preia coordonatele din fiecare pixel apoi folosind funcția Start_Laser() trimite comenzile la cele două DAC-uri. La sfârșit se actualizează starea diodei laser și se execută un delay datorită faptului că rezoluția celor două motoare nu este suficient de rapidă pentru a putea urmării punctele cu viteza de transmitere a datelor pe SPI. Dacă nu ar exista acest delay, motoarele nu ar avea suficient timp să ajungă la coordonatele pixelului iar imaginea ar fi distorsionată.

Funcția *void Prepare_Command_Motor_X(unsigned int value)* primește ca parametru valoarea de 16biti pentru valoarea de output a DAC-ului care corespunde motorului OX și salvează jumătățile de cuvânt ce urmează a fi transmise pe SPI în variabilele globale de 16 biți fiecare: *high_word_full_command_x și low_word_full_command_x.*

Funcția *void Prepare_Command_Motor_Y(unsigned int value)* face același lucru ca funcția de mai sus, doar că salvează în variabilele *high_word_full_command_y low_word_full_command_y* pentru DAC-ul care corespunde motorlului OY.

Funcția *void Start_Laser()* trimite către DAC-uri comenzile scrise în variabilele descrise mai sus.

5.3.7 Modul ADC

Conform datasheet-ului (v.[14]) , modulul ADC poate fi configurat cu ajutorul regiștrilor:

- AD1CON1 – registru de configurare
- AD1CON2 – registru de configurare
- AD1CON3 – registru de configurare
- AD1CON5 – registru de congirurare
- AD1CHS – A/D SAMPLE SELECT REGISTER
- ANCFG – BAND GAP REFERENCE CONFIGURATION REGISTER
- AD1CHITH - SCAN COMPARE HIT REGISTER
- AD1CSSH - INPUT SCAN SELECT REGISTER
- AD1CTMENH CTMU ENABLE REGISTER

Datorită faptului că modulul ADC al proiectului trebuie configurat să funcționeze cu un potențiometru, mulți din regiștrii de mai sus vor rămâne în starea lor default.

Funcția *void ADC_Init()* inițializează modulul adc astfel încât pin-ul RB3 va fi luat ca input.

Funcția *unsigned int ADC_Get()* inițializează analiza, așteaptă ca flag-ul DONE ce indică faptul că rezultatul analizei este gata și returnează valoarea de 10 biți din bufferul adc-ului, aceasta reprezentând valoarea citită de la potențiometru.

5.3.8 Modul Întreruperi

Modulul de întreruperi (interrupts.c, interrupts.h) poate fi configurat și utilizat conform datasheet-ului (v.[11]) folosind următorii regiștrii

- INTCON1 – registru de control
- INTCON2 – registru de control
- INTCON4 – registru de control
- IFx – registru cu flag-uri
- IECx – registru de configurare generală
- IPCx – registru de configurare a priorității întreruperilor

Pentru ca întreruperile din proiect să funcționeze este necesar ca bitul GIE din registrul de control INTCON2 sa fie setat, astfel vom activa întreruperile globale. Acest lucru este realizat de funcția *void Enable_Global_Interrupts()*

Pentru o mai buna organizare a codului, întreruperile folosite în proiect au fost configurate direct în module.

5.3.9 Modulul MAIN

Biții de configurare (Configuration bits) permit programatorului să ajusteze anumite condiții care determină modurile de funcționare ale microcontrolerului. Starea biților de configurare determină modul în care MCU-ul funcționează atunci când este alimentat.

Modificările aduse biților de configurare față de starea lor „default”:

#pragma config POSCMD = HS //Selectăm oscilatorul extern

#pragma config FNOSC = PRIPLL //Selectăm modulul PLL

#pragma config PLLMODE = PLL6X //Configurăm modulul PLL pentru a multiplica frecvența de 6 ori.

Astfel, pentru sursa de clock vom aveam oscilatorul extern ce se află pe placa de test ce emite o frecvență de 8MHz. Aceasta frecvență este multiplicată cu ajutorul modulului de PLL la 48MHz, iar apoi va fi înjumătățită datorită faptului că arhitectura MCU-ului de 16 biți presupune că o instrucțiune să se execute pe durata a doua perioade de clock, în final ajungându-se la o frecvență interna de 24MHz.

În bulca infinită *while(1)* se execută funcția Draw_Ram_Array() împreună cu asignarea într-o variabila a valorii venite de la modulul de ADC, respectiv interpretarea potențiometrului.

Rutinele întreruperillor de SPI și Timer1 se află la finalul fișierului. Acestea controlează închiderea și deschiderea diodei laser în funcție de input-ul primit de la potențiometru prin intermediul modulului ADC.

5.4 Descrierea aplicației Python

Funcția *def img_transform_greyscale(img_name)* primește ca parametru calea către o imagine și folosind librăria opencv convertește imaginea în format greyscale. Mai departe, crează o matrice care este inițiată cu valori de 0 pe linie și 0 pe coloana, folosind ca dimensiune numărul de pixeli din imaginea inițială. Se parcurge imaginea pixel cu pixel și acolo unde se găsește un pixel ce are valoarea diferită de 0, se va marca acest lucru în matricea nou creată punând valoarea 1 valorii corespunzătoare pixelului. La final, se returnează matricea A creata.

Funcția *def active_value_matrix(A)* primește ca parametru o matrice de tipul celei returnate de funcția img_transform_greyscale(img_name) și crează o nouă matrice AVM inițializată cu tuple de (0,0,0) folosind ca dimensiune matricea primita. În variabila locala diff se calculează care va fi diferența de distantă dintre oricare doi pixeli folosind valoarea maximă de output a dac-ului (2^{16}=65535) împărțită la numărul de valori a matricei cu „pixeli" de pe linie sau coloană. Mai departe, se parcurge matricea primită ca parametru și acolo unde se găsesc valori diferite de 1 (adică pixeli care nu sunt albi) se transpun în matricea AVM sub formă de coordonate. Diferența dintre oricare doua coordonate consecutive este valoarea calculata în variabila *diff*. La sfârșit se returnează matricea AVM creată.

Funcția *def array_transform(AVM,A)* primește ca parametru o matrice de tipul celei returnate de funcția active_value_matrix și o matrice de tipul celei returnate de funcția img_transform_greyscale, parcurgând cea de a doua matrice folosind o parcurgere sus-jos stangă-dreapta, acolo unde sunt valori diferite de 1 (pixeli care nu sunt albi), se va adaugă tupla de coordonate respectivă în lista creată. La sfârșit se returnează lista cu tuplele valorilor acelor pixeli care sunt diferiți de 1.

Funcția *def add_laser_diff(L)* primește că parametru o listă de tipul celei returnate de funcția *array_transform(AVM,A),* iar parcurgând această listă, va scrie pe prima poziție a fiecărei tuple diferența de pixeli dintre pixelul actual și următorul pixel, astfel se va ști în orice moment distanța dintre doi pixeli activi (diferiți de 1) consecutivi.

Funcția *def WriteSerial(L,NR_Active_Pixels)* primește ca parametru o listă de tipul celei returnate de funcția add_laser_diff și trimite pe serială numărul de pixeli activi și valorile pixelilor din lista de tuple. Acest lucru se face folosind funcția Serial din librăria Serial, ce primește ca

parametrii numele serial-port-ului pe care urmează să fie transmisă informația, rata de transfer și „timeout”.

Funcția *def start_laser()* apelează toate funcțiile descrise mai sus, cu scopul de a trimite pe serială valorile pixelilor activi dintr-o poză.

Interfața grafică are scopul de a oferi utilizatorului o metodă simplă și eficientă de a interacționa cu aplicația pentru ca aceasta să-și atingă scopul. Aspectul aplicației este unul simplu, având doua butoane și un preview la poza selectata.

Funcția *def deschide()* apelează funcția *filedialog.askopenfilename* din librăria *tkinter,* cu scopul de a oferi utilizatorului șansă de a selecta o poză pe care o dorește să o proiecteze cu laserul, salvând calea ei în variabila file_root. Funcția nu acceptă extensii de tipul „.jpg” , iar pentru acest lucru trebuie folosită funcția ImageTk.PhotoImage din librăria PIL care va face transformarea către formatul acceptat. La final se vă pune poza în GUI, poziționând-o cu ajutorului funcției grid în dreapta butoanelor.Prin apasarea butonului „Alege o imagine” se va apela functia *deschide().*

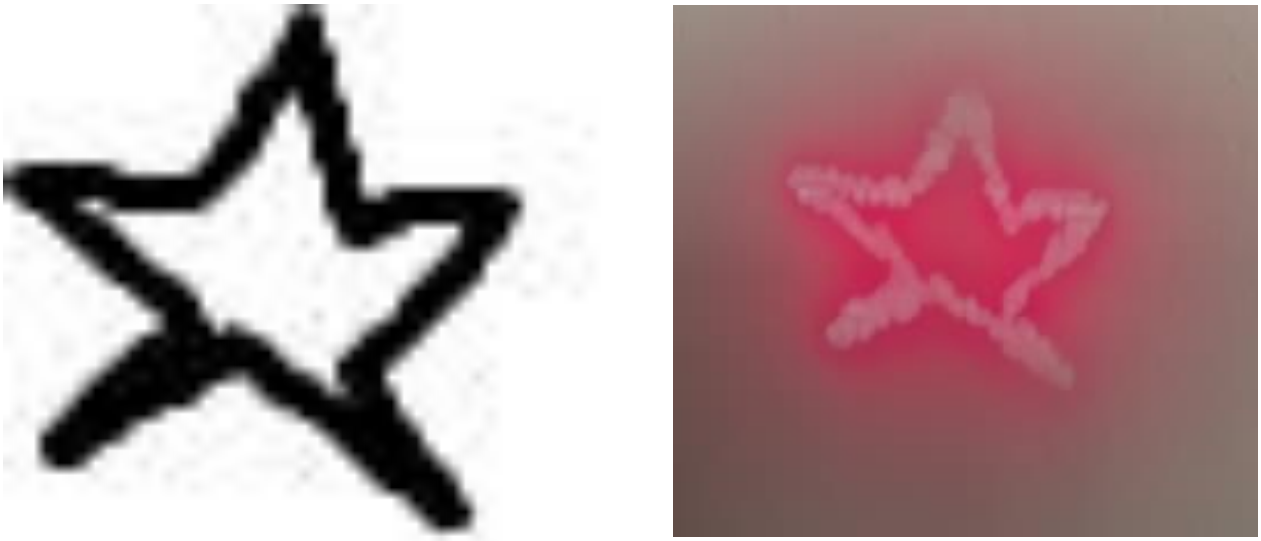
Prin apăsarea butonului „Start laser” se va apela funcția *start_laser(),* iar sistemul va începe să funcționeze.

CONCLUZII

Rezumat

Lucrarea de față poate să preia o imagine salvată local, să o proceseze cu ajutorul aplicației dezvoltate in limbajul Python, salvând pixelii activi într-un array pe care îl trimite cu ajutorul serialei către MCU. MCU-ul preia de pe serială dimensiunea array-ului împreună cu array-ul în sine și îl parcurge trimițând valori către cele doua DAC-uri prin intermediul SPI. Totodată, MCU-ul controlează dioda laser prin intermediul unui pin GPIO. Cele două DAC-uri preiau comenzile de la SPI și setează semnalul analogic pentru cele două motoare galvo cu oglindă, acestea redirecționează unda laser, proiectând o imagine pe suprafața de proiecție.

Rezultate obținute



Imaginea obținută din fișierul star50.jpg

Imaginea obținută din fișierul heart50.jpg

Posibile îmbunătățiri

Una din îmbunătățirile ce pot fi aduse proiectului ar fi eliminarea reglajului cu potențiometru, acesta făcându-se automat în funcție de distanta pixelilor activi.

Folosirea unei modalități de a prelua imaginile direct de pe internet ar fi o îmbunatățire majoră datorită faptului că proiectorul nu va mai fi dependent de o conexiune la calculator. O alta îmbunatățire ar fi citirea pozelor de pe un sd-card și realizarea unui nou design compatibil cu standardul IP61 pentru a putea fi pus în aer liber fară a fi distrus de condițiile meteo nefavorabile sau praf. Acest lucru poate fi realizat o dată cu integrarea ansamblurilor electronice pe o singură placă de circuit integrat.

Pentru a proiecta o imagine color se poate alege un sistem format din 3 lasere (RGB). De asemenea, se poate implementa conceptul de „projection mapping”, o tehnică ce este folosită în proiecția imaginilor pe spații cu forme neregulate.

Datorită faptului că ansamblul consumă relativ puțin curent și poate fi amplasat deasupra unei clădiri, se poate implementa și încărcarea cu ajutorul celulelor fotovotaice, astfel sistemul va putea funcționa pe baza energiei alternative.

BIBLIOGRAFIE

[1] „Analog-Digital Conversion”, http://www.analog.com/media/en/training-seminars/design-handbooks/Data-Conversion-Handbook/Chapter3.pdf

[2] „Basic Of Uart Communication” http://www.circuitbasics.com/basics-uart-communication/

[3] „Dual Axis Galvanometer Optical Scanners” https://www.edmundoptics.com/f/dual-axis-galvanometer-optical-scanners/13188/

[4] „Galvonameter Scanners” https://www.scanlab.de/en/products/galvanometer-scanners

[5] „Introduction to Microcontrollers” https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller

[6] „Introduction to OpenCV-Python Tutorials” https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html

[7] „Laser diode”, https://www.physics-and-radio-electronics.com/electronic-devices-and-circuits/semiconductor-diodes/laserdiode.html

[8] „Laser”, https://ro.wikipedia.org/wiki/Laser

[9] „LTC2601”, Linear Tehnology

[10] „Microcontroller” https://internetofthingsagenda.techtarget.com/definition/microcontroller

[11] „PIC24FJ1024GA610/GB610 Family Manual”, Microchip

[12] „Section 14. Timers”, Microchip

[13] „Section 51. 12-Bit A/d Converter with Threshold Detect”, Microchip

[14] „Serail Peripheral Interface (SPI)”, Microchip

[15] „Serial Peripheral Interface(SPI)” https://whatis.techtarget.com/definition/serial-peripheral-interface-SPI

[16] „The Fundamentals of Laser Tehnology”, https://www.ulsinc.com/learn.

[17] „The Working Principle of PID Controller for Beginners” https://www.elprocus.com/the-working-of-a-pid-controller/

[18] „UART Communication Protocol – How it works?” https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/

[19] „Universal Asynchronous Reciver Transmitter (UART)”, Microchip