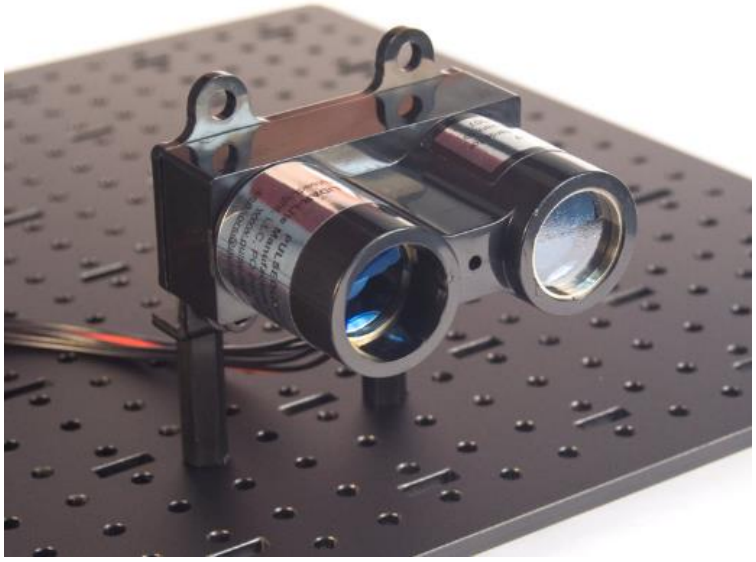
Facultatea de Automatică și Calculatoare
Programul de Licență: Calculatoare

DRIVER PENTRU MODUL LIDAR

Proiect de diplomă



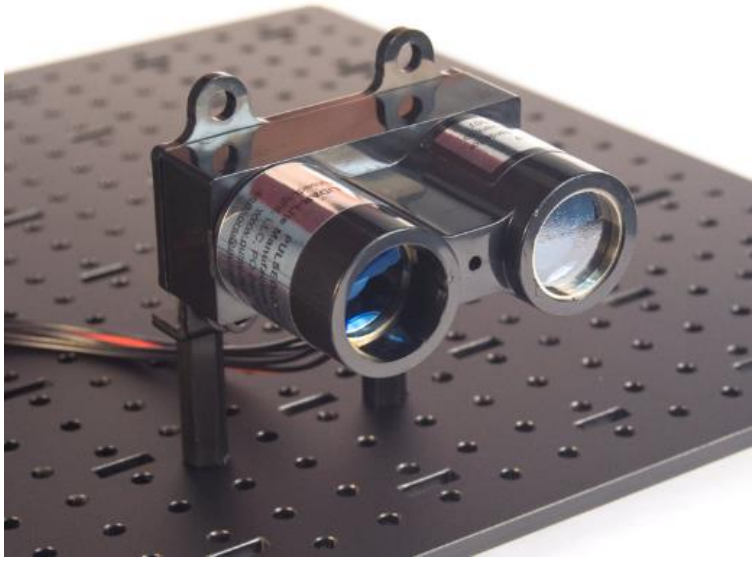
Candidat:
Octavian Teofil DIACONESCU

Conducător ştiinţific:
Ș.l dr.ing. Valentin STÂNGACIU

Timișoara
2019

DRIVER PENTRU MODUL LIDAR

Proiect de diplomă



Candidat:
Octavian Teofil DIACONESCU

Conducător ştiinţific:
Ș.l dr.ing. Valentin STÂNGACIU

Timișoara
2019

DECLARAŢIE DE AUTENTICITATE A LUCRĂRII DE FINALIZARE A STUDIILOR

Subsemnatul _____,
legitimat cu _____seria _____nr. _____,
CNP _____autorul lucrării_____
_____
_____
elaborată în vederea susţinerii examenului de finalizare a studiilor de licență
_____
organizat de către Facultatea de Automatică și Calculatoare din cadrul Universităţii Politehnica Timişoara, sesiunea _____ a anului universitar_____, luând în considerare conţinutul art. 39 din RODPI – UPT, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Timişoara,
Data Semnătura

_____ _____

Declaraţia se completează „*de mână*" şi se inserează în lucrarea de finalizare a studiilor, la sfârşitul acesteia, ca parte integrantă.

Cuprins

Cuprins **3**
Listă acronime **5**
Listă figuri **6**
Listă tabele **7**
Listă secvențe cod **8**
1 Introducere **9**
1.1 Domeniul abordat ... 9
1.2 Contextul lucrării ... 9
1.3 Specificații generale ... 10
1.4 Structura lucrării ... 11
2 Specificații teoretice **13**
2.1 Senzori ... 13
2.2 Tipuri de senzori ... 15
2.2.1 Senzor infraroșu ... 15
2.2.2 Senzor temperatură ... 15
2.2.3 Senzor ultrasonic ... 16
2.2.4 Senzor Hall ... 17
2.3 Măsurarea mărimilor fizice cu ajutorul senzorilor ... 17
2.4 Principii de funcționare ale senzorului LIDAR ... 19
2.4.1 Măsurarea distanței cu ajutorul senzorul LIDAR-Lite v3 ... 19
2.4.2 Utilizarea senzorulului LIDAR-Lite v3 ... 20
3 Sisteme Încorporate **21**
4 Fundamente de fiabilitate, redundanţă şi mentenabilitate **23**
4.1 Fiabilitatea ... 23
4.2 Mentenabilitatea ... 23
4.3 Redundanţa ... 24
4.4 Fault Tree Analysis ... 24
5 Funcționarea programului **26**
5.1 Schema bloc hardware ... 26
5.2 Schema bloc software ... 27
6 Implementarea hardware a modulului de măsurare **28**

6.1 Sursele de tensiune folosite 28
6.2 Conectarea senzorului LIDAR prin I2C 29
6.3 Conectarea senzorului LIDAR prin PWM 30
7 Implementarea software a modulului de măsurare 31
7.1 Drivere 31
7.2 Microcontroller-ul NXP LPC2294 31
7.3 Inițializarea modulelor 32
7.4 Interfața serială UART 34
7.5 Implementarea protocolului de comunicație I2C 36
7.5.1 Transmisia datelor de la master la slave 37
7.5.2 Citirea datelor primite de la slave 39
7.5.3 Mecanismul de tratare al acknowledge-urilor 41
7.5.4 Măsurarea distanţei cu ajutorul senzorului LIDAR 43
7.6 Configurarea senzorului LIDAR 45
8 Funționarea sistemului de măsurare 49
9 Rezultate experimentale 51
9.1 Distanțe măsurate și performanțele senzorului LIDAR 51
9.2 Verificarea implementării protocolului de comunicație I2C 52
10 Concluzii și perspective de dezvoltare 53
10.1 Concluzii 53
10.2 Sinteza contribuțiilor 53
10.3 Perspective de dezvoltare 53
Referințe bibliografice 55

Listă acronime

DTM - **D**igital **T**errain **M**odels

DSM - **D**igital **S**urface **M**odels

DEM - **D**igital **E**levation **M**odels

LIDAR - **Li**ght **D**etection **A**nd **R**anging

DSPLabs - **D**igital **S**ignal **P**rocessing **Lab**oratorie**s**

UART - **U**niversal **A**synchronous **R**eceiver-**T**ransmitter

I2C - **I**nter-**I**ntegrated **C**ircuit

PWM - **P**ulse **W**idth **M**odulation

LED - **L**ight **E**mitting **D**iode

LCD - **L**yquid **C**rystal **D**isplay

GPIO - **G**eneral-**P**urpose **I**nput/**O**utput

IoT - **I**nternet **o**f **T**hings

FTA - **F**ault **T**ree **A**nalysis

UART - **U**niversal **A**synchronous **R**eceiver-**T**ransmitter

PLL - **P**hase **L**ocked **L**oop

CPU - **C**entral **P**rocessing **U**nit

I/O - **I**nput/**O**utput

LCR - **L**ine **C**ontrol **R**egister

THR - **T**ransmit **H**olding **R**egister

RDR - **R**eceive **D**ata **R**egister

SCL - **S**erial **C**lock **L**ine

SDA - **S**erial **D**ata **L**ine

Listă figuri

Figura 1-1. Tehnologia DEM
Figura 1-2. Senzorul LIDAR-Lite V3
Figura 1-3. Senzor HC-SR04
Figura 2-1. Sistem de senzori
Figura 2-2. Senzor activ și senzor pasiv
Figura 2-3. Senzor infraroșu
Figura 2-4. Senzor temperatură TMP36
Figura 2-5. Senzor ultrasonic
Figura 2-6. Senzor Hall
Figura 2-7. Principiul de funcționare al mașinilor autonome
Figura 2-8. Principiul de funcționare al senzorilor ultrasonici
Figura 2-9. Măurarea distanței cu ajutorul senzorului LIDAR-Lite v3
Figura 3-1. Structura generală a unui sistem embedded
Figura 4-1. Fault Tree Analysis
Figura 5-1. Schema block hardware
Figura 5-2. Schema block software
Figura 6-1. Sursele de tensiune folosite
Figura 6-2. Pinout senzor LIDAR-Lite V3
Figura 6-3. Conectarea senzorului LIDAR prin I2C
Figura 6-4. Conectarea senzorului LIDAR prin PWM
Figura 7-1. Instruction pipiline ARM7
Figura 7-2. Load-and-store arhitecture
Figura 7-3. UART data format
Figura 7-4. Arhitectura UART LPC2294
Figura 7-5. Conectarea dispozitivelor la magistrala I2C
Figura 7-6. Inițierea secvenței de scriere de date
Figura 7-7. Inițierea secvenței de citire de date
Figura 7-8. Mecanism tratare acknowledge-uri pentru scriere
Figura 7-9. Mecanism tratare acknowledge-uri pentru citire
Figura 8-1. Control Flow Graph
Figura 9-1. Captură analizor logic

Listă tabele

Tabel 7-1. Tabel comparaţie configuraţii

Tabel 9-1. Tabel performanțe senzor LIDAR

Tabel 9-2. Tabel rezultate măsurători

Listă secvențe cod

Cod 7-1. Secvența de cod pentru inițializarea senzorului LIDAR
Cod 7-2. Secvența de cod pentru inițializarea interfeței seriale UART
Cod 7-3. Secvența de cod pentru transmiterea unui caracter pe interfața UART
Cod 7-4. Secvența de cod pentru recepția unui caracter pe interfața UART
Cod 7-5. Secvența de cod pentru inițierea condiției de start
Cod 7-6. Secvența de cod pentru scrierea de date pe magistrala I2C
Cod 7-7. Secvența de cod pentru inițierea condiției de stop
Cod 7-8. Secvența de cod pentru citirea datelor
Cod 7-9. Secvența de cod pentru executarea unei măsurători
Cod 7 10. Secvența de cod pentru configurarea default a senzorului
Cod 7-11. Secvența de cod pentru modul 1 de configurare a senzorului
Cod 7-12. Secvența de cod pentru modul 2 de configurare a senzorului
Cod 7-13. Secvența de cod pentru modul 3 de configurare a senzorului
Cod 7-14. Secvența de cod pentru modul 4 de configurare a senzorului
Cod 7-15. Secvența de cod pentru modul 5 de configurare a senzorului

1 Introducere

1.1 Domeniul abordat

Proiectul ales de către mine abordează domeniul din industria calculatorelor ce se ocupă de controlul unui sistem încorporat (embedded system) folosit în tehnologia de măsurare și topografiere a zonelor foarte mari. Această tehnologie a devenit cea mai utilizată în momentul de fața pentru generarea digitală, la o rezoluție ridicată, a zonelor de teren foarte întinse și greu accesibile. Cele mai cunoscute tehnologii din acest domeniu poartă numele de DTM (**D**igital **T**errain **M**odels), DSM (**D**igital **S**urface **M**odels) și DEM (**D**igital **E**levation **M**odels) [1]. Aceste tehnologii sunt folosite în general pentru maparea spațiului și pentru a prezice anumite fenomene naturale, cum ar fi alunecările de teren sau inundațiile.

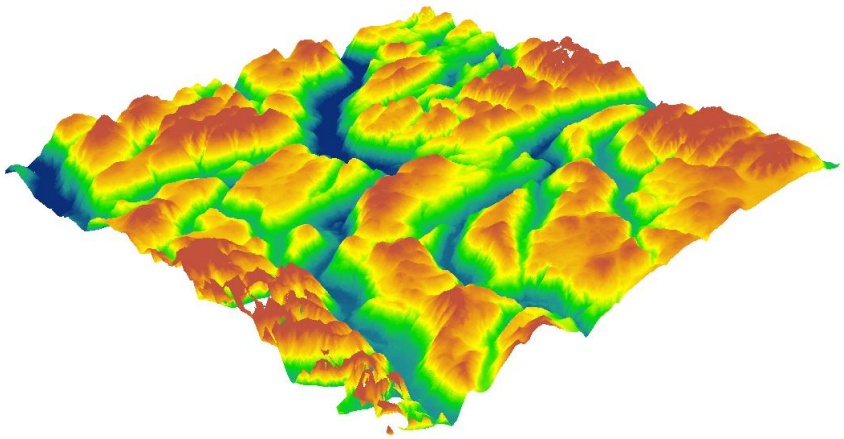


Figura 1-1. Tehnologia DEM [1]

1.2 Contextul lucrării

Evoluția rapidă a sistemelor încorporate din ultima perioadă a dus tot mai mult la dezvoltarea activitaților din diverse domenii, în care, nu cu mult timp în urmă, executarea activităților nu se realiza cu ajutorul sistemelor încorporate, însa, în momentul în care sisteme au devenit suficient de performante, aceste domenii au putut progresa foarte mult. În acest context se încadrează și domeniul din care face parte lucrarea abordată de către mine, cel al mapării spațiului înconjurator. Dacă în urmă cu ceva timp acest lucru era realizat cu ajutorul aparatelor de topografiat, în prezent există sisteme bazate pe tehnologia LIDAR (**Li**ght **D**etection **A**nd **R**anging) capabile să mapeze suprafețe foarte mari de teren cu o precizie mult mai bună decât dispozitivele utilizate în trecut.

Motivația pentru a implementa un astfel de sistem a venit din dorința de a învăța mai multe despre sistemele încorporate, despre protocoalele de comunicație dintre un microcontroller și dispozitivele cu care acesta comunică, dar și dezvoltarea cunoștiințelor de electronică și hardware.

Lucrarea a fost realizată in cadrul DSPLabs (**D**igital **S**ignal **P**rocessing **Lab**oratorie**s**) din cadrul Facultății de Automatică și Calculatoare, Departamentul Calculatoare, fiind coordonată de domnul s.l dr. ing. Valentin Stângaciu.

Ulterior, această lucrare va fi folosită in cadrul DSPLabs.

1.3 Specificații generale

Preocuparea generală a acestei lucrări a fost implementarea unor librării care conțin funcțiile de bază necesare pentru a putea comunica și pentru a realiza măsurători cu ajutorul senzorului LIDAR- Lite V3. Aceste librării sunt implementate independent de platforma pe care v-or fi utilizate, asigurând astfel portabilitatea sistemului.

La început, axare s-a făcut pe implementarea unor funcții generice cum ar fii: comunicația UART (**U**niversal **A**synchronous **R**eceiver-**T**ransmitter), implementarea unor timere software și a modulului de întreruperi.

Cea mai interesantă și complicată parte a fost implementarea protocolului de comunicație I2C (**I**nter-**I**ntegrated **C**ircuit) dintre senzorul LIDAR-Lite V3 și placa de dezvoltare Olimex.

Proiectul a presupus totodată și implementarea unui prototip hardware cu ajutorul căruia a fost posibilă testarea funcțiilor implementate.

Figura 1-2. Senzorul LIDAR-Lite V3

Ultima parte a proiectului a presupus implementarea funcțiilor necesare pentru a putea configura și pentru a putea realiza măsurători cu ajutorul senzorului LIDAR și compararea rezultatelor obținute în urma măsurătorilor cu acest senzor cu rezultatele obținute în urma măsurătorilor folosind un senzor HC-SR04.

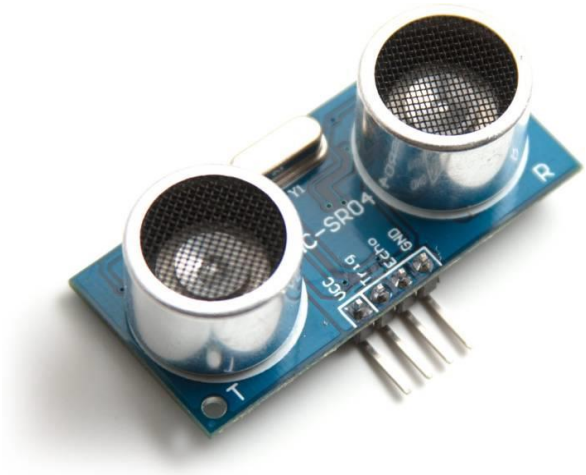


Figura 1-3. Senzor HC-SR04

Din punct de vedere al complexității, proiectul a fost de dificultate medie, având atât parți care au fost relativ ușor de implementat cât și părți care au necesitat o atenție sporită la implementare.

1.4 Structura lucrării

În cele ce urmează se v-or prezenta atât aspecte teoretice legate de senzori, cât și implementarea hardware și software a unui sistem de măsurare cu ajutorul unui senzor LIDAR-Lite v3.

În capitolul 1 se prezintă o scurtă introducere a lucrării și este prezentată motivația alegerii acestei teme.

În capitolul 2 sunt prezentate aspectele teoretice legate de senzori și executarea măsurătorilor utilizând sistme de senzori. Este prezentată în detaliu execuția măsurătorilor utilizând senzorul LIDAR-Lite v3.

În capitolul 3 sunt prezentate aspecte principale legate de sistemele embedded cât și domeniile de aplicație în care se folosesc aceste sisteme. Totodată sunt prezentate și direcțiile de dezvoltare ale acestor sisteme.

Capitolul 4 descrie mecanismele de fiabilitate și redundanță atât la nivel teoretic, cât și la nivel de contribuție proprie. Contribuția proprie constă în analizarea unor metode de fiabilitate pentru sistemul de măsurare și întocmirea unei scheme bazate pe metoda Fault Three Analysis.

În capitolul 5 prezintă o contribuție proprie ce conține descrierea de ansamblu a sistemului. Se prezintă schemele bloc hardware și software ale sistemului de măsurare și se descrie pe scurt fiecare componentă folosită.

Capitolul 6 prezintă implementarea hardware a senzorului LIDAR-Lite v3, cât și interconectarea acestuia cu placa de dezvoltare Olimex.

Capitolul 7 prezintă implementarea software care stă la baza sistemului.

În capitolul 8 se prezintă funcționare sistemului de măsurare, cât și flow-ul programului parcurs din momentul inițializării sistemului până în momentul logării datelor preluate de la senzor.

În capitolul 9 sunt prezentate rezultatele experimentale rezultate după implementarea proiectului.

În ultima parte a acestei lucrări sunt prezentate concluziile rezultate în urma implementării acesteia, contribuțiile proprii la această lucrare, cât și sursele de inspirație care au făcut posibilă implementarea acestei lucrări.

2 Specificații teoretice

2.1 Senzori

Progresul tehnicilor de procesare și dezvoltarea rapidă a sistemelor încorporate și a tehnologiei calculatoarelor necesită și cercetări avansate în implementarea senzorilor. Microprocesoarele sunt tot mai des încorporate în sisteme de măsurare și control foarte avansate. Având în vedere faptul că dezvoltarea acestor sisteme crește, rolul senzorilor, folosiți ca și unitați principale de captare a datelor, crește considerabil, crescând totodată și necesitatea de dezvoltare continuă a acestora. Senzorii au devenit factori importanți în automatică și în industria roboților, totodată căpătând și o foarte ridicată importanță într-un sistem.

În teorie, senzorii sunt unitați primare, parți componente ale lanțului de dispozitive de măsurare dintr-un sistem, care convertesc o mărime fizică într-un semnal electric ce poate fi ulterior procesat [6].

Schema generală a unui sistem de senzori este prezentată în figura 2-1. Semnalul ce intră în senzor este de amplitudine mică și interferează cu alte semnale și cu zgomot. Pentru ca semnalul de ieșire să poată fi conturat cu caracteristici optime și procesat ulterior, semnalul de intrare este trecut prin mai multe amplificatoare, filtre digitale și alte circuite analogice. Semnalul este mai apoi convertit în semnal digital și transmis unității de procesare.

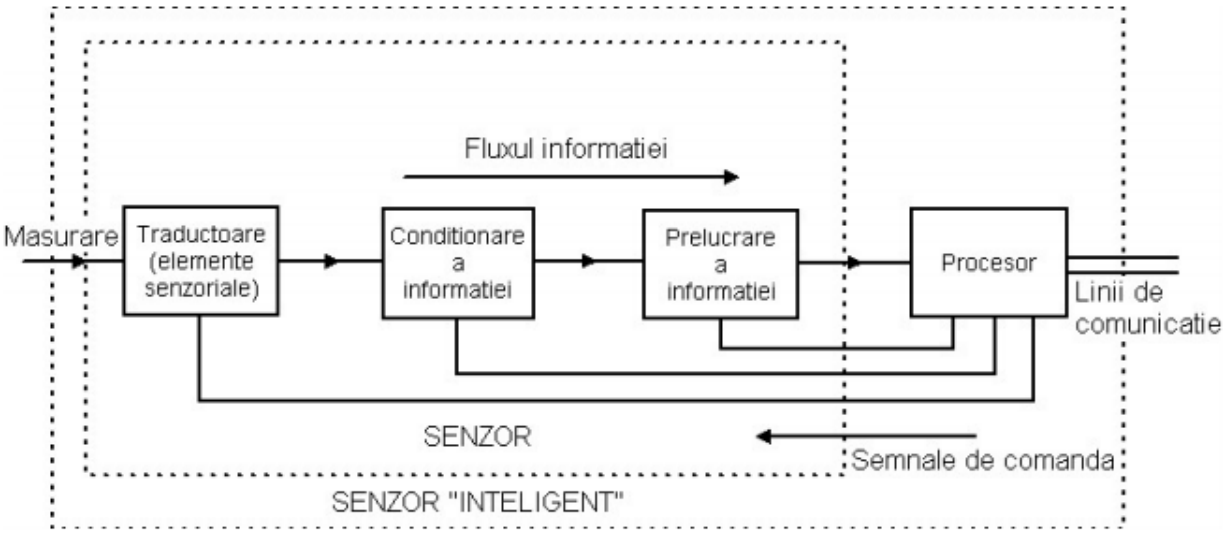


Figura 2-1. Sistem de senzori [4]

Extrapolând considerațiile despre sistemele senzoriale ale lumii vii la sisteme de senzori, senzorii trebuie să fie capabili să identifice, în anumite condiții și limite, parametrii mediului ambiant și să reacționeze la modificări ale acestora. Pentru a face posibilă identificarea parametrilor, senzorii, în funcție de nivelul de integrare, pot avea funcții mai simple sau mai complexe. Senzorul cuprinde traductorul, folosit pentru transformarea mărimii de intrare într-un semnal electric util, dar și circuite pentru

adaptarea și conversia semnalelor și, eventual, pentru prelucrarea și evaluarea informațiilor.

Caracteristicile sistemelor de senzori sunt puternic determinate de către senzorii din componența acestor sisteme. Acești senzori pot fi împărțiți în două categorii: activi și pasivi. Senzorii activi convertesc un tip de energie direct în alt tip de energie, fără a avea nevoie de surse externe de energie. Senzorii pasivi nu pot converti direct energiile captate, dar le pot controla.

Ieșirea senzorilor este reprezentată de conversia unei mărimi fizice (de exemplu: mecanică, termică, electromagnetică, etc.) într-o valoare electrică.

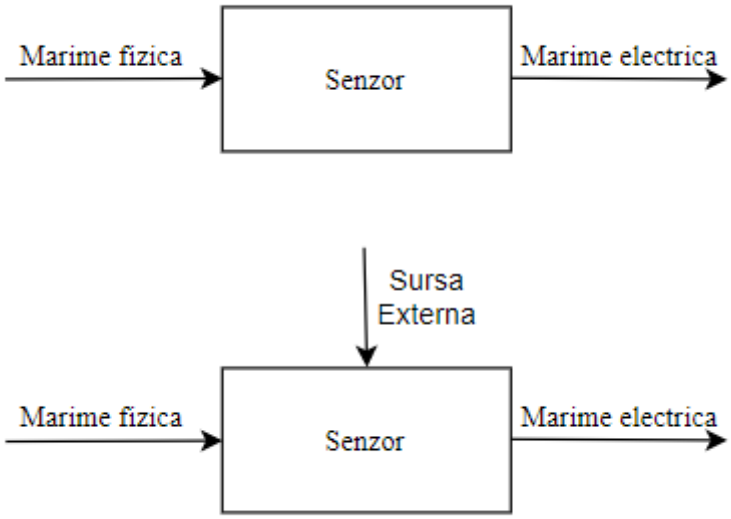


Figura 2-2. Senzor activ și senzor pasiv

În funcționarea senzorilor intervin diferite efecte fizice și principii de funcționare. Senzorii pot capta direct mărimea măsurată sau pot face acest lucru fără a intra în contact cu aceasta. De exemplu, senzorii de umiditate intră în contact direct cu mediul din care aceștia preiau informații, dar există însă şi senzori, cum ar fii senzorii Hall, care pot capta informații fără a intra în contact cu mediul.

Pentru ca un senzor să fie eficient, anumite cerințe clare trebuiesc îndeplinite. Aceste cerințe pot fi de bază (specifice oricărui tip de senzor) sau specifice (în funcție de tipul de senzor folosit). Cerințele de bază ale unui senzor sunt: senzitivitate ridicată, liniaritate, acuratețe, reproductibilitate, viteză, stabilitate la perturbații și zgomote, perioadă îndelungată de funcționare, dimensiuni reduse, rezistentă la factorii mediului înconjurător (căldură, umiditate, apă, praf, etc.), siguranță și cost redus. Cerințele specifice ale senzorilor sunt în funcție de tipul acestora.

2.2 Tipuri de senzori

Având în vedere domeniul vast în care se folosesc senzorii, este de așteptat să existe un număr foarte mare de tipuri de senzori. Cele mai uzuale tipuri de senzori sunt: senzori cu infraroșu, senzori de temperatură si umiditate, senzori de proximitate, senzori ultrasonici, senzori Hall, senzori de presiune și senzori laser.

2.2.1 Senzor infraroșu

Un senzor cu infraroşu este un dispozitiv electronic care măsoară în infraroşu, lumina care radiază de la obiectele aflate în câmpul său vizual. Mişcarea este detectată atunci când un corp cu o anumită temperatură (cum ar fi un om) trece prin fața sursei infraroșu cu o altă temperatură (cum ar fi un zid). Senzorul detectează căldura de la trecerea unui obiect prin câmpul de acțiune al său și acel obiect rupe câmpul pe care senzorul l-a determinat ca fiind „normal”. Orice obiect, chiar unul de aceeaşi temperatură ca şi obiectele din jur, va activa senzorul dacă corpul se deplasează în câmpul vizual al senzorului. Toate obiectele emit energie sub formă de radiaţii. De obicei, radiaţiile infraroşii sunt invizibile pentru ochiul uman, dar pot fi detectate de dispozitive electronice concepute cu acest scop.

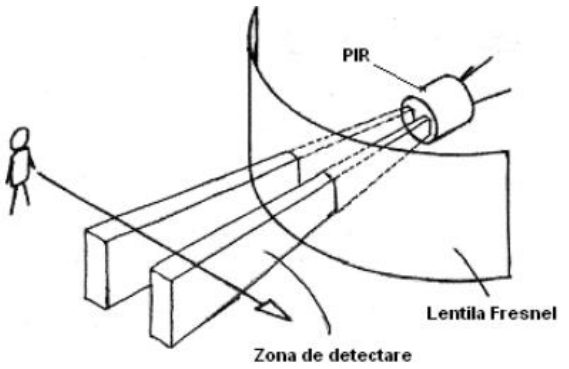


Figura 2-3. Senzor infraroșu [4]

2.2.2 Senzor temperatură

Măsurarea temperaturii constituie una dintre cele mai uzuale procese de măsurare. Probabil, cel mai simplu şi mai des folosit principiu în măsurarea temperaturii este dilatarea termică. Acesta este principiul ce stă la baza termometrelor din sticlă cu lichid. Pentru a transforma energia termică în semnal electric, se folosesc detectori rezistivi, termoelectrici, optici şi piezoelectrici. Când un senzor este introdus într-un obiect sau plasat pe suprafaţa obiectului, va exista un transfer de căldură între sondă şi obiect: senzorul se va răci sau se va încălzi, în funcţie de temperatura obiectului [5].

Există două metode de procesare a semnalului în măsurătorile de temperatură: metoda echilibrării şi metoda predictivă. În primul caz, temperatura se va măsura doar în momentul în care nu mai există gradient de temperatură între senzor şi obiect (au

aceeaşi temperatură), iar în cazul metodei predictive, punctul de echilibru nu este atins niciodată, ci este determinat din viteza de schimbare a temperaturii senzorului.

Figura 2-4. Senzor temperatură TMP36

2.2.3 Senzor ultrasonic

Senzorii ultrasonici sunt relativ simplii și ușor de interfațat cu diferite dispozitive din diferite arii de aplicabilitate. Senzorul emite un impuls sonic și așteaptă ecoul apărut la întâlnirea undei cu un obiect. Impulsul este emis de un traductor care face conversia între energia electrică, mecanică și acustică. Timpul dintre emiterea pulsului și recepționarea ecoului său, este folosit în determinarea distanței. Senzorii ultrasonici nu sunt perfecți având și unele dezavantaje. Pulsul emis are o formă de con iar orice corp care intră în interacțiune cu unda emisă va declanșa un ecou. Este imposibil să se facă diferența între obiecte mici sau mari având în vedere că toate emit un ecou. Această problemă are ca soluție folosirea mai multor senzori sau folosind senzori rotativi.

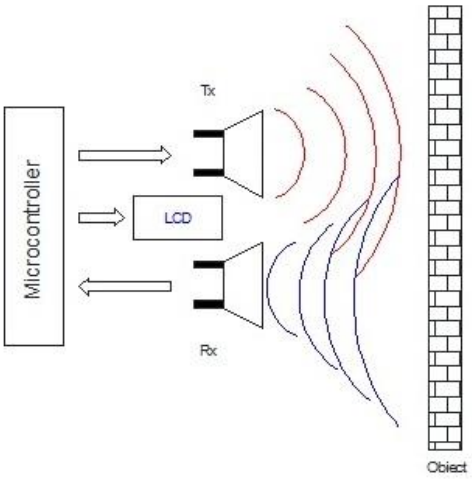


Figura 2-5. Senzor ultrasonic [7]

2.2.4 Senzor Hall

Senzorul funcționează bazat pe principiul efectului magnetic (numit și efectul Hall). Mărimea de intrare a senzorului este reprezentată de câmpul magnetic, iar mărimea de ieșire este un semnal electric. Atunci când, într-un câmp magnetic, se plasează un conductor purtător de curent, o tensiune este generată perpendicular pe câmpul magnetic și pe curentul electric [].

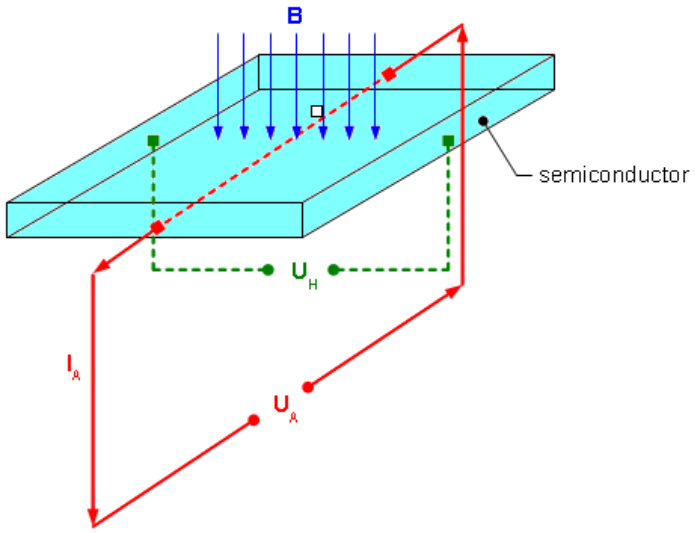


Figura 2-6. Senzor Hall [8]

2.3 Măsurarea mărimilor fizice cu ajutorul senzorilor

Evoluția continuă a tehnologiei și necesitățile din ce în ce mai ridicate au fost unii din factorii principali care au dus la înlocuirea treptată a instrumentelor clasice de măsurare a distanței cu instrumente electronice de măsurare. Principalele instrumente de măsurare a distanței, în momentul de fața, sunt senzorii. Printre cei mai utilizați senzori de măsurare a distanței sunt senzorii ultrasonici, care fac măsurători cu ajutorul undelor sonore, senzorii cu infraroșu, care folosesc razele infraroșii pentru a face măsurători și senzorii cu laser. Principalele motive pentru care acești senzori au înlocuit instrumentele clasice sunt: precizia de măsurare și rezoluția acestor dispozitive.

Spre exemplu, un impact major a înlocuirii dispozitivelor clasice de măsurare cu senzori de măsurare, se poate observa în domeniul automotive. Având în vedere că mașinile inteligente, capabile să se conducă singure, nu mai sunt o noutate, a fost necesară dezvoltarea senzorilor de măsurare a distanței în spațiu. Acești senzori trebuie să fie foarte preciși și, totodată, trebuie să ofere tot timpul informații corecte. Dacă măsurătorile făcute de către acești senzori, nu ar fi exacte, mașina nu ar putea să determine poziția sa în spațiul în care se află, iar impactul acestui lucru ar putea fi devastator.

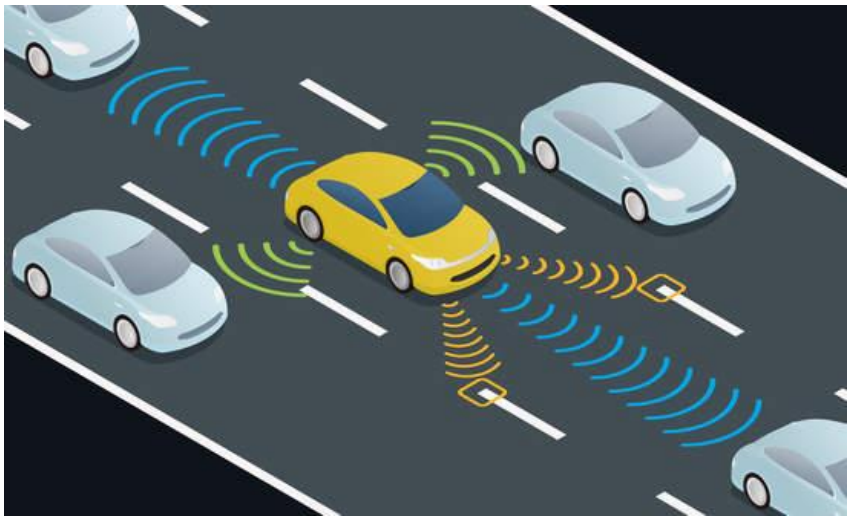


Figura 2-7. Principiul de funcționare al mașinilor autonome

Senzorii ultrasonici sunt des folosiți în automotive pentru a măsura distanțe, schimbări de poziție și măsurări de nivele. Acești senzori funcționează bazați pe principiul măsurării propagării undelor ultrasonice în timp. Acest principiu asigură posibilitatea de măsurare a obiectelor, indiferent de forma pe care o au, materialul din care sunt construite sau design-ul acestora, prin urmare fiind posibilă detectarea obiectelor solide, lichide, care au suprafețe lucioase, transparente, etc. Alt factor determinant pentru folosirea acestui tip de senzori este rezistența la diferite medii (De exemplu: praf, apă, temperaturi scăzute, temperaturi ridicate, etc.), capabilitatea de adaptare a acestor senzori fiind foarte ridicată.

Undele ultrasonice au o caracteristică de propagare similară cu undele audio, principiul care stă la baza acestei propagări fiind vibrarea particulelor din mediul în care acestea se propagă. Această propagare poate fi facută prin medii gazoase, lichide și solide. Undele ultrasonice sunt în general asemănate cu un sunet de o frecvența mai mare de 20 de kHz. Viteza de propagare a acestor unde depinde de mediul de propagare și de temperatura mediului.

Senzorii ultrasonici funcționează bazaţi pe principiul măsurării timpului dintre momentul în care se trimite o pulsație acustică și momentul în care reflecția sunetului transmis este recepționată. Aceste pulsații se propagă cu viteza lumii, iar, în momentul în care întâlnesc un obiect, o parte din ele sunt reflectate.

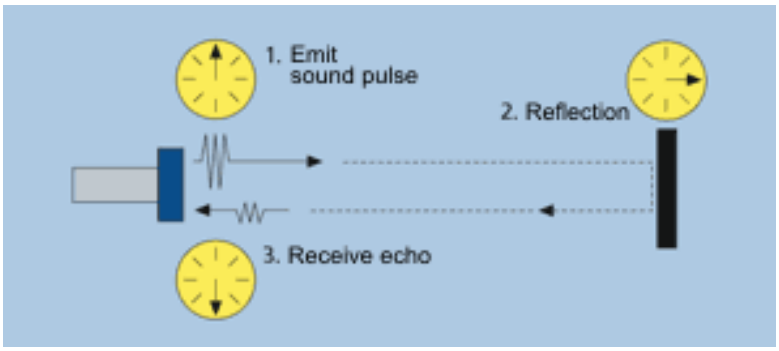


Figura 2-8. Principiul de funcționare al senzorilor ultrasonici [7]

2.4 Principii de funcționare ale senzorului LIDAR

2.4.1 Măsurarea distanței cu ajutorul senzorul LIDAR-Lite v3

Tehnologia de măsurare a distanței folosind senzori bazați pe principiul emiterii și captării undelor laser, este relativ nouă, însa, introducerea acestei tehnologii a ajutat mult la dezvoltarea industriei.

Dispozitivul folosit de către mine, pentru realizarea acestei lucrări, este senzorul LIDAR-Lite v3. Pentru a măsura distanța, senzorul calculează diferența de timp dintre momentul emiterii unei unde laser și momentul în care această unda este reflectată și captată de către dispozitiv. Translatarea timpului în distanță are ca referință viteza de propagare a luminii. Acest senzor emite o undă laser marcată și asteapă până când aceași undă se reflectă, asigurând astfel o eficiența ridicată și, totodată, o protecție suplimentară a persoanei care utilizează acest senzor [9].

Diferite tehnici de procesare a semnalelor sunt folosite pentru a putea obține rezultate diferite, în funcție de necesități. Acest senzor este capabil să execute măsurători foarte sensibile și precise, la un cost de energie scăzut.

Pentru a realiza o măsurătoare, dispozitvul execută prima dată o rutină cu ajutorul căreia se realizează o corecție de tip bias, care ajută la măsurători foarte sensibile, în cazul în care mediul ambiental se schimbă în timp ce sunt executate aceste măsurători. După aplicarea acestei corecții, dispozitivul transmite un semnal de referință de la transmitător la receptor și inițializează timer-ul cu ajutorul căruia se calculează timpul din momentul în care o undă laser este emisă până când aceasta este recepționata. După ce timer-ul este inițializat, senzorul inițiază o măsurătoare, făcând mai multe achiziții. Dacă semnătura undei emise coincide cu semnătura undei așteptată la recepție, rezultatul măsurătorii este salvat în memorie. Dispozitivul realizează achiziții până în momentul în care valoarea de vârf a semnalului receptat ajunge la o valoare maximă. Dacă unda reflectată nu e sufficient de puternică, dispozitivul se oprește. Puterea semnalului este calculată de la magnitudinea semnalului celui mai puternic până la un prag setat anterior, în funcție de zgomotul din mediul în care se execută măsurători. Dacă semnalul măsurat este peste acest prag, măsurătoarea este considerată validă, în caz contrar, dispozitivul consideră valoarea invalidă.

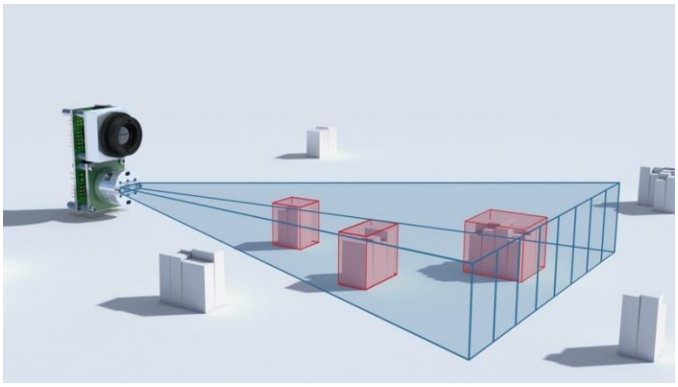


Figura 2-9. Măsurarea distanței cu ajutorul senzorului LIDAR-Lite v3 [9]

2.4.2 Utilizarea senzorului LIDAR-Lite v3

În momentul în care senzorul este alimentat sau în momentul în care acesta se resetează, acesta executa o secvență de test și inițializează toți regiștrii pe valorile inițiale. Această perioadă de inițializare durează aproximativ 20 milisecunde, iar după aceasta, se pot executa măsurători.

Senzorul poate comunica cu un sistem încorporat prin doua metode: utilizând interfata I2C sau prin semnal PWM (**P**ulse **W**idth **M**odulation).

Pentru comunicația cu ajutorul interfeței I2C, dispozitivul are un modul I2C de tip 2-wire (modul cu două fire, SDA și SCL) cu ajutorul căruia, dispozitivul se poate contecta la o magistrală de tip I2C ca și un dispozitiv de tip slave, sub controlul unui dispozitiv de tip master.

Dispozitivul poate fi configurat în diverse moduri, aceste moduri fiind prezentate ulterior în această lucrare.

3 Sisteme Încorporate

Progresul neîncetat al industriei calculatoarelor a dus la necesitatea unor noi dispositive capabile să execute anumite cerințe specifice, într-un timp cât mai scurt și cu o precizie cât mai bună. Astfel au apărut sistemele încorporate, care au rolul de a interconecta componenta hardware și componenta software a unui sistem, cu un scop predefinit, pentru a putea executa cerințe și pentru a procesa informații într-un timp foarte scurt, având o precizie foarte bună.

Aceste sisteme se bazează pe circuite logice programabile, având căi de dezvoltare diferite. Circuitele logice pot fi: microprocesorul (circuit logic programabil de utilizator pentru aplicații de uz general), microcontroller-ul (un circuit logic programabil de utilizator pentru aplicații în timp real) și DSP-ul (un circuit logic programabil de utiliator pentru procesarea digitală a semnalelor analogice) [10].

Diferențele dintre un sistem embedded și un calculator de uz general constau în modul de operare al acestora. Dacă în cazul calculatoarelor, dispozitivele de afișare a informațiilor sunt o parte foarte importantă din sistem, în cazul sistemelor embedded lipsesc, interfața cu utilizatorul realizându-se cu ajutorul LED-urilor (**L**ight **E**mitting **D**iode), LCD-urilor (**L**yquid **C**rystal **D**isplay), a comutatoarelor și a minitastaturilor. Dacă în cazul calculatoarelor, dispozitivele periferice sunt indispensabile, în cazul sistemelor embedded, acestea de asemenea lipsesc, fiind înlocuite de diferite porturi, denumite GPIO (**G**eneral-**P**urpose **I**nput/**O**utput). Cu toate acestea, diferența cea mai mare dintre calculatoarele obișnuite și sistemele embedded se găsește la nivelul software-ului. Dacă în cazul calculatoarelor, software-ul poate fi foarte diversificat, fiind capabil să execute mai multe aplicații, în cazul sistemelor embedded, software-ul reprezintă o funcționalitate fixă și specifică aplicației.

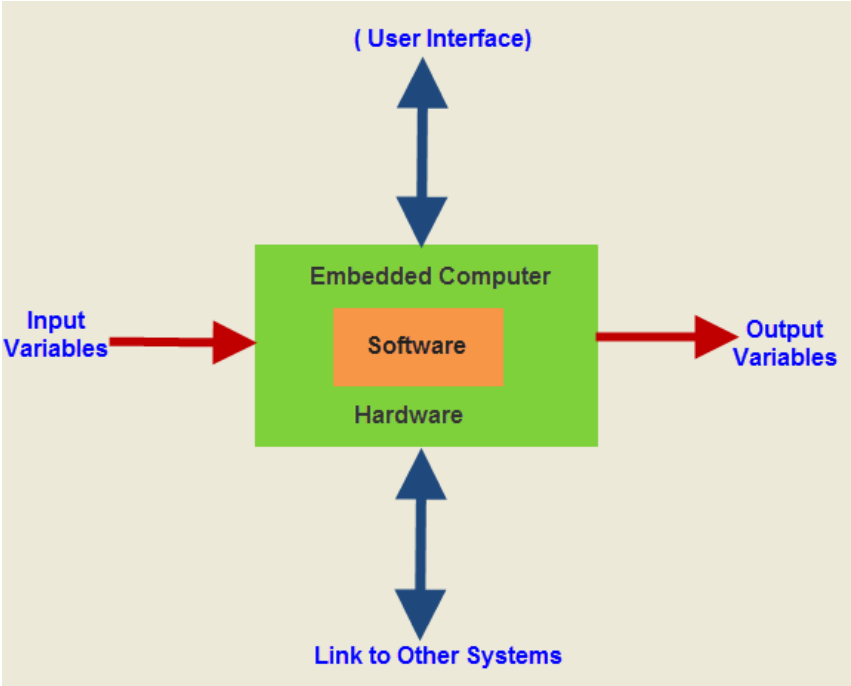


Figura 3-1. Structura generală a unui sistem embedded

Domeniile de aplicabilitate ale acestor sisteme sunt foarte vaste. Cel mai răspândit domeniu este industria automotive. În prezent, orice dispozitiv din acest domeniu, de la cel mai nesemnificativ la cel mai critic (cum ar fi unitatea de control a airbag-ului unei mașini), este format dintr-un sistem embedded. Acest domeniu nu e singurul în care aceste sisteme s-au impus. Dacă în industria roboților și a telecomunicațiilor nu este o noutate utilizarea sistemelor embedded, în domeniul medicinii aceste dispozitive devin tot mai folosite. Datorită preciziei mult superioare fața de precizia umană, nu e de mirare că viitorul medicinei constă intr-o colaborare strânsă între domeniul medicinii și cel al ingineriei.

Cu toate că aceste dispozitive au evoluat foarte mult, ele sunt înca într-o continuă îmbunatatire. Având în vedere performanțele de care dau dovadă aceste dispozitive, a apărut conceptul IoT(**I**nternet **o**f **T**hings). Principalul scop al acestui concept este combinarea senzorilor și a sistemelor embedded pentru a îmbunătăți viața de zi cu zi a omului.

4 Fundamente de fiabilitate, redundanţă şi mentenabilitate

Necesitatea de funcţionare continuă, în condiţii optime, a sistemelor embedded a dus la dezvoltarea unor principii pe baza cărora sunt implementate aceste sisteme. Pentru ca un sistem să satisfacă nevoile pentru care a fost implementat, nu este suficient ca acesta doar să îşi îndeplinească funcţionalitatea. El trebuie să fie fiabil, trebuie să aibă un echilibru între cost şi eficienţă şi trebuie să poată fi întreţinut uşor.

4.1 Fiabilitatea

Fiabilitatea (notată cu R(t)) reprezintă probabilitatea condiţionată ca un sistem să funcţioneze corect într-un interval de timp $[t_0,t]$, având în vedere că, la momentul t_0, sistemul funcţiona corespunzător, iar t este perioada de viaţa a produsului [13].

În cazul dispozitivele complexe, pentru a respecta aceast principiu, poziţionarea componentelor într-un sistem este văzută în două moduri:

- Fiabilistic: are ca scop punerea în evidenţa a faptului că fiabilitatea unui element din cadul sistemului poate influenţa fiabilitatea altor elemente din acel sistem;
- Structural: presupune analiza fiabilităţii sistemului în funcţie de poziţionarea componentelor în sistem.

Există patru moduri de dispunere a componentelor:

- serie: dacă una din componente se defectează, întreg sistemul se defectează;
- paralel: dacă toate componentele s-au defectat, atunci şi întreg sistemul se defectează (funcţionarea rămâne până când ultima componentă se defectează);
- serie/paralel: dacă toate componentele s-au defectat, atunci şi întreg sistemul se defectează;
- paralel/serie: dacă toate componentele s-au defectat, atunci şi întreg sistemul se defectează;

Conceptul de fiabilitate nu se poate separa de cel de mentenabilitate, având capacitatea ca un sistem să se repare după defectare (auto-reparare).

4.2 Mentenabilitatea

Proprietatea unui sistem de a fi mentenabil se poate definii în două moduri: calitativ şi cantitativ. Din punct de vedere al calităţii, capacitatea de funcţionare a sistemului într-o anumită perioadă de timp, fără a produce defecţiuni. Din punct de vedere cantitativ, mentenabilitatea reprezintă capacitatea sistemului de a îndeplinii funcţiile pentru care a fost implementat.

Proprietatea de mentenabilitate este foarte importantă pentru sisteme deoarece este principalul factor care trebuie luat în calcul în perioada de maturitate a produsului, după ce acesta a fost introdus pe piaţa şi pus în funcţiune.

Având în vedere uzura ce poate interveni după o perioadă îndelungată de folosire, sistemului dezvoltat de către mine a fost implementat ţinând cont de proprietatea de mentenabilitate a sistemului, deoarece, acesta presupune foarte multe măsurători într-un timp foarte scurt. Pentru a mă asigura că dispozitivul nu işi pierde din performanţă în timp, au fost implementate diferite moduri de funcţionare, astfel, atâta timp cât dispozitivul nu execută măsurători, acesta se afla într-o stare "low power" unde consumul acestuia este foarte mic. În momentul în care se primeşte un trigger de măsurare, microcontroller-ul şi senzorul LIDAR se iniţializează, după care se pot executa măsurători. Acest lucru duce la scăderea uzurii în timp a senzorului.

4.3 Redundanţa

Redundanţa reprezintă o optimizare a echilibrului dintre eficienţă şi cost operaţional. Această proprietate are rol de păstrare a funcţionalităţii în cazul de defecţiune a unuia sau a mai multor dispozitive/echipamente. Acest principiu constă în instalarea unor componente suplimentare şi preluarea funcţionarii celorlalte elemente care s-au defectat [11].

Putem avea 4 modalități de implementare a redundanței:

- Hardware;
- Software;
- Informațional;
- Temporal;

În implementarea lucrării nu s-a ţinut prea mult cont de acest aspect, însa, impelmentarea unui mecanisc care să înlocuiască funcţionarea sistemului sau funcţionarea unei componente ar putea fi o perspectiva ulterioară de dezvoltare.

4.4 Fault Tree Analysis

FTA (**F**ault **T**ree **A**nalysis) este o metodă utilizată în perioada de mentenanță, în fiabilitatea sistemului și în analizarea impactului asupra sistemului în cazul în care o componentă cedează. Aceste evenimente neplăcute pot apărea din cauze software, hardware sau a erorilor umane [12].

Metoda deductivă pornește de la o concluzie generală, iar apoi se încearcă determinarea cauzelor specifice până la ultimul element ce poate avea o cauză la acea concluzie. Prin acest mecanism se construiește un arbore al defecțiunii survenite din cauze multiple sau simple.

Principalul scop al analizei acestui arbore este de a ajuta la identificarea potențialelor cauze de defectare a sistemului înainte ca eșecurile să apară propriu-zis.

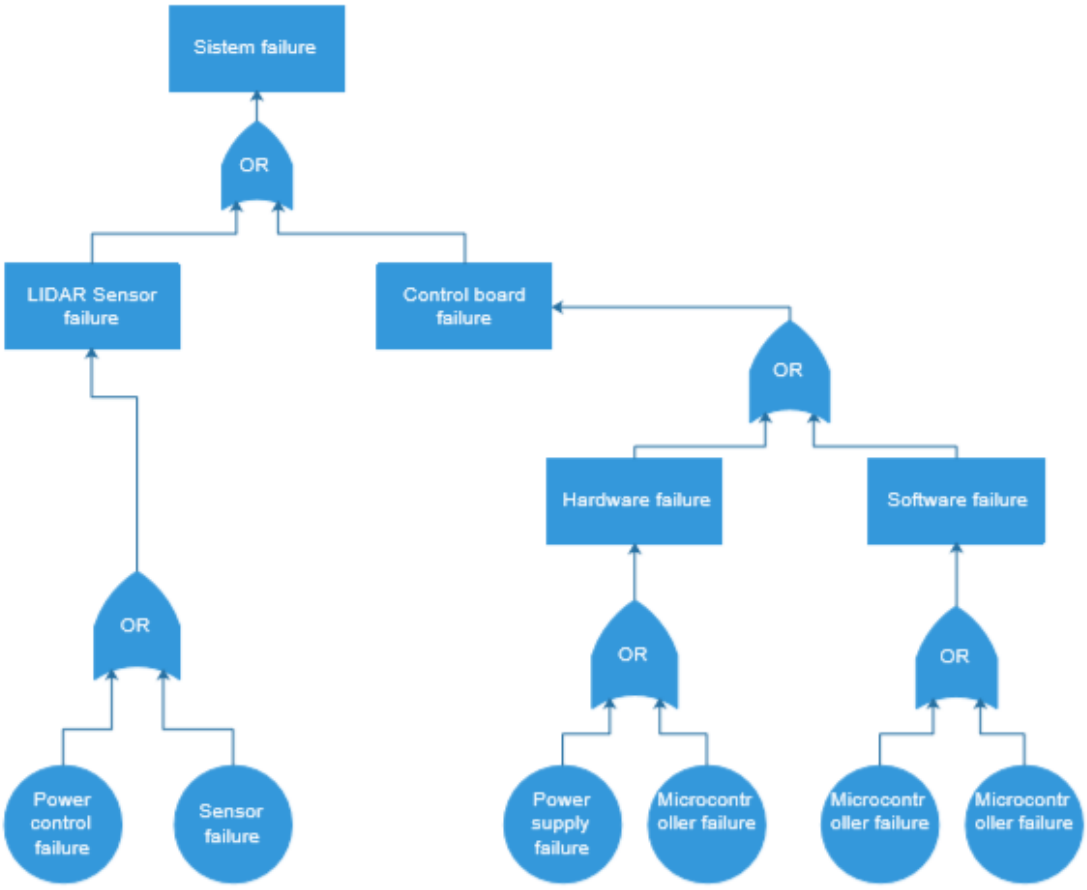


Figura 4-1. Fault Tree Analysis

Pentru realizarea acestui grafic au fost definite evenimentele nedorite la nivel de componente și deducearea efectelor imediate ale acestor evenimente. Odată definite evenimentele si efectele acestora, se continuă cu analiza la un nivel superior a sistemului, până când se ajunge la nivelul cel mai de sus al acestuia.

5 Funcționarea programului

Sistemul implementat este compus din două componente mari, componenta hardware și componenta software. În acest capitol este prezentată doar funcționarea în ansamblu a sistemului, urmând ca în următoarele capitole să se prezinte în detalui implementarea hardware și software.

5.1 Schema bloc hardware

Arhitectura hardware generală a sistemului este prezentată în Figura 5-1. Scopul principal al acestei arhitecturi este de a controla partea electronică a senzorului LIDAR-Lite v3 și de a interconecta acest senzor cu placa de dezvoltare Olimex lpc-h2294.

Rolul senzorului este de a măsura distanța de la sistem până la diferite obiecte. Această măsurare trebuie să fie suficient de precisă și trebuie ca tot timpul să rezulte aceleași rezultate, dacă se execută în aceleași condiții.

Comunicația dintre senzor și placa de dezvoltare se realizează prin intermediul protocolului I2C, acesta fiind explicat în detaliu în cele ce urmează.

Comunicația dintre sistemul implementat de către mine și alte siteme se realizează prin comunicația serială asincronă (UART). Rolul acestei comunicații este de afișarea datelor primite de la senzor pe seriala unui calculator, utilizând programul Docklight Scripting.

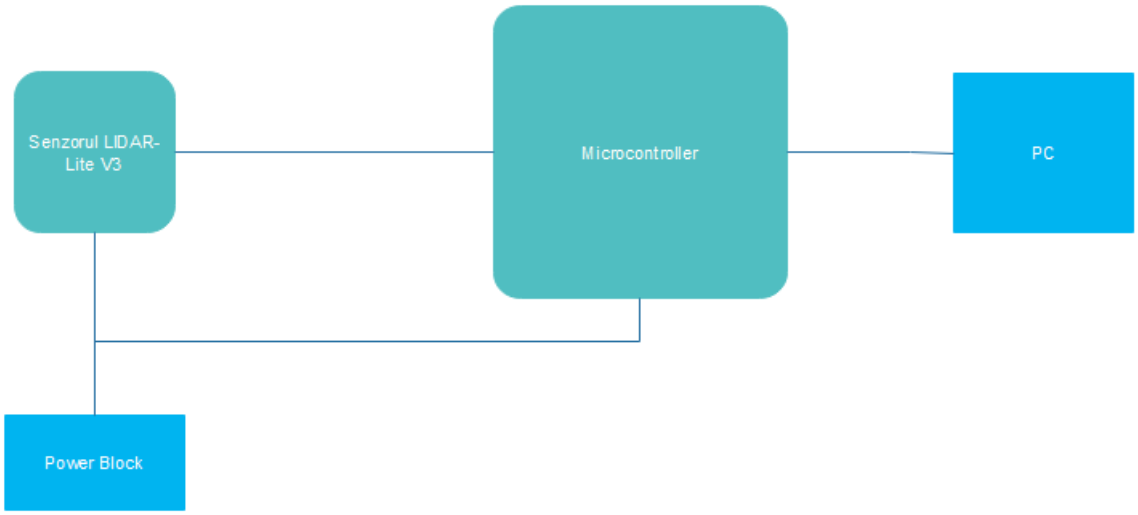


Figura 5-1. Schema block hardware

5.2 Schema bloc software

Arhitectura software generală a sistemului este prezentată în Figura 5-2. Scopul principal al acestei scheme bloc este de a prezenta flow-ul programului în momentul în care se execută o măsurătoare. "Creierul" sistemului este microcontroller-ul LPC2200 bazat pe nucleul ARM7 TDMI.

Pentru a executa o măsurătoare, flow-ul programului este următorul: se inițializează microcontroller-ul și senzorul LIDAR, se trimite o comandă de execuție de măsurătoare, senzorul execută măsurătoare, se trimit datele de la senzor la microcontroller, urmând ca datele să fie transmise pe serială unui calculator.

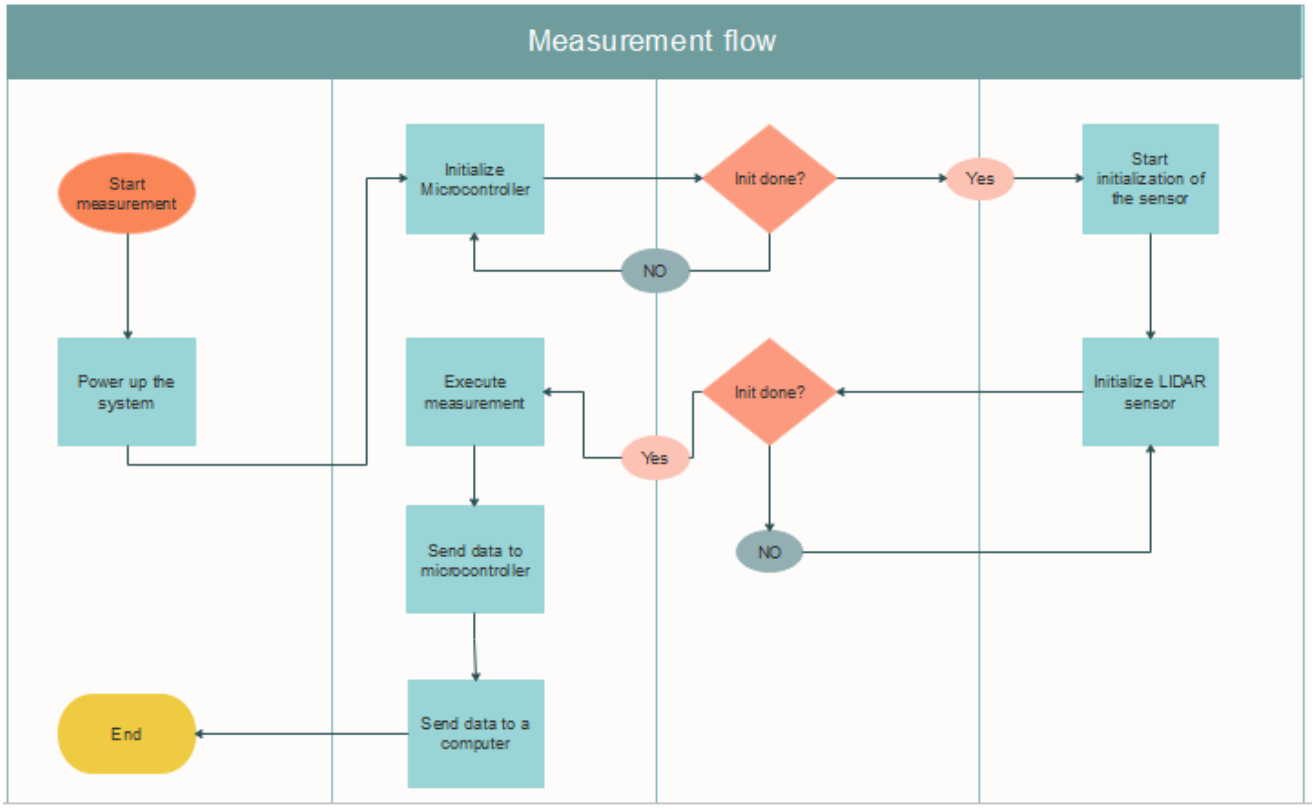


Figura 5-2. Schema block software

6 Implementarea hardware a modulului de măsurare

Implementarea hardware a lucrării a presupus construirea unui prototip hardware în care a fost integrat senzorul LIDAR.

Conexiunea dintre senzorul LIDAR şi un sistem embedded se poate realiza în două moduri, în funcţie de cum se doreşte realizarea comunicării dintre cele două dispozitive.

În primul mod de conectare se foloseşte magistrala I2C pentru a realiza comunicarea, iar în al doilea caz se foloseşte o metodă bidirectională de transfer de semnal care declanşează achiziţii şi returnează distanţa măsurată cu ajutorul pinului mode-control.

6.1 Sursele de tensiune folosite

Pentru alimentarea senzorului LIDAR şi a plăcii de dezvoltare Olimex s-a folosit un acumulator de 9 volţi.

Senzorul LIDAR necesită 5V tensiune stabilizată pentru a putea funcţiona corect. Pentru a avea această tensiune constantă de 5V, am conectat un stabilizator de tensiune de 5V pe linia de alimentare. Stabilizatorul de tensiune folosit este L78S05.

Placa de dezvoltare Olimex a trebuit alimentată de la 3.3V deoarece structura internă nu permite alimentarea cu 5V ca şi tensiune de intrare. Pentru a reduce tensiunea de alimentare de la 5V la 3.3V s-a folosit un stabilizator de tensiune de 3.3V TC1264.

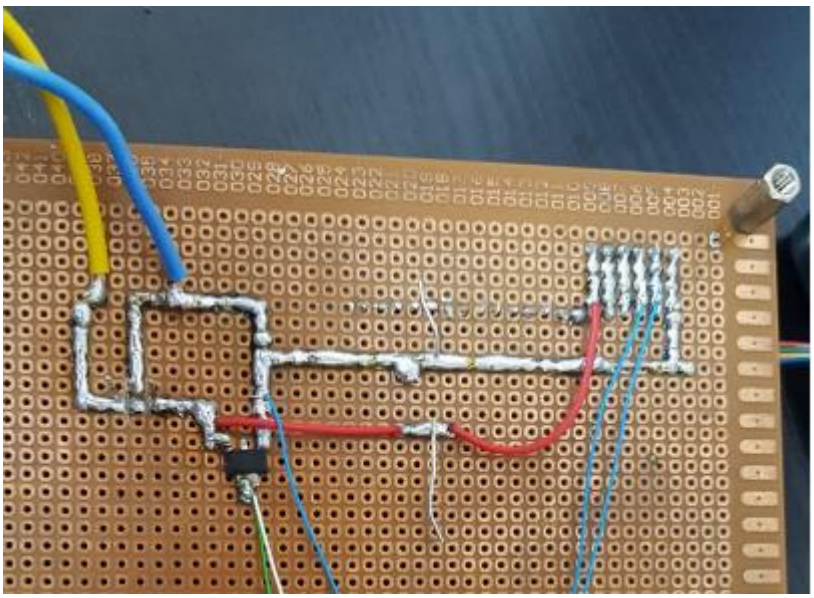


Figura 6-1. Sursele de tensiune folosite

6.2 Conectarea senzorului LIDAR prin I2C

Senzorul LIDAR-Lite V3 are în total 6 pini. Prin aceşti pini se realizează atât alimentarea senzorului cât şi comunicarea senzorului cu un dispozitiv de tip master.

Primul pin este pinul de alimentare. Alimentarea senzorului trebuie facută de la 5V fiind necesară o tensiune de alimentare constantă.

Al doilea pin este pinul de “power-enable” şi nu este folosit în implementarea facută de către mine. Pinul foloseşte un rezistor intern de pull-up cu ajutorul căruia se poate dezactiva senzorul.

Al treilea pin este pinul “mode control” şi este folosit pentru a realiza comunicarea dintre senzor şi un dispozitiv master, în cazul în care este folosită metoda bazată pe semnal PWM.

Următorii doi pini sunt pinii SDA şi SCL şi sunt folosiţi pentru a conecta senzorul la magistrala I2C a unui dispozitiv de tip master.

Ultimul pin este pinul de GND.

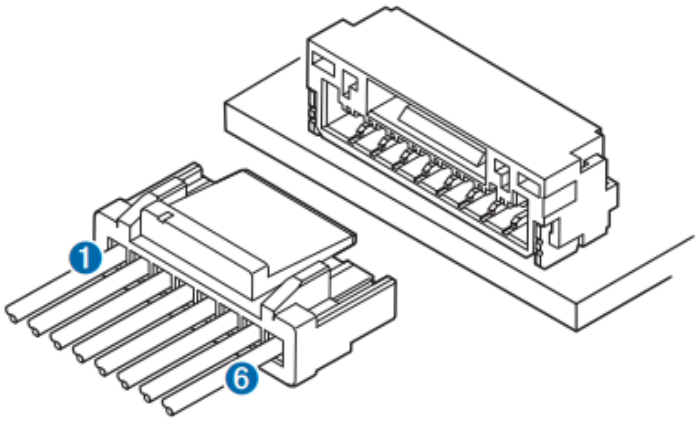


Figura 6-2. Pinout senzor LIDAR-Lite V3 [9]

Între pinul de GND şi pinul de alimentare, se montează un condensator de 680µF care este folosit ca şi un condensator de filtrare cu rolul de a reduce pulsaţiile de pe linia de alimentare a stabilizatorului de tensiune. Valoarea condensatorului trebuie să fie cel puţin egală cu 680µF, această valoare fiind calculată bazat pe consumul de curent a senzorului.

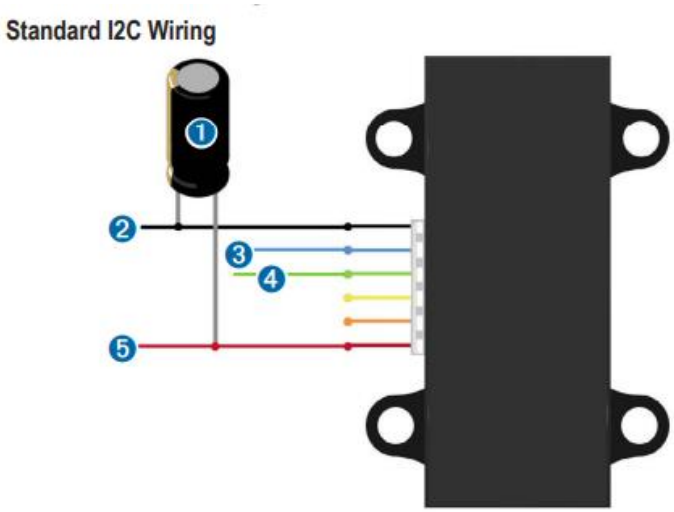


Figura 6-3. Conectarea senzorului LIDAR prin I2C [9]

6.3 Conectarea senzorului LIDAR prin PWM

Pentru a realiza comunicarea senzorului LIDAR cu un dispozitiv master, prin utilizarea semnalului de PWM, trebuie folosit pinul "mode-control". Acest pin oferă un mijloc de declanşare a achiziţiilor şi returnează valoarea măsurată printr-un semnal de tip PWM, fără a folosi interfaţa I2C.

În modul inactiv al pinului mode control, acesta are o impedanţă ridicată. Trăgând pinul în GND, se declanşează o măsurătoare şi dispozitivul ridică linia semnalului de PWM, prin care se returnează valoarea măsurată. Pentru a converti semnalul în centimetrii, se foloseşte formula 10µs = 1cm. Este necesară folosirea unei rezistor de 1kΩ pentru a preveni conflictele pe magistrală.

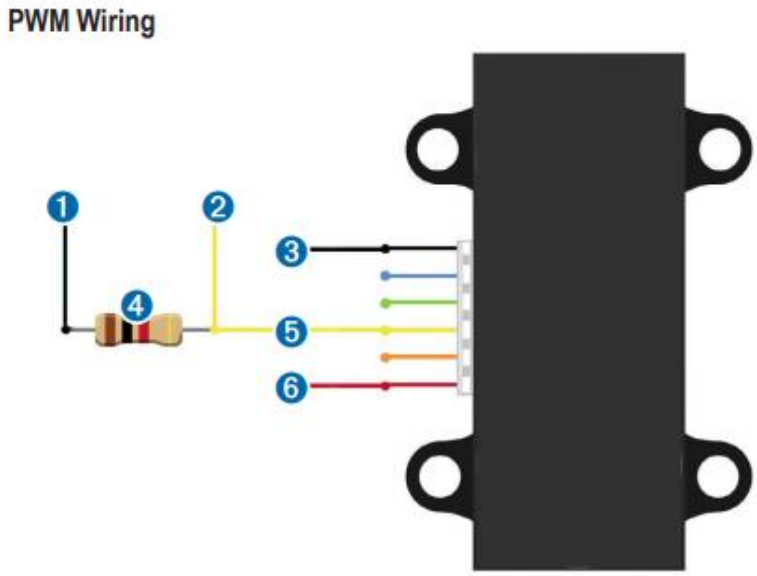


Figura 6-4. Conectarea senzorului LIDAR prin PWM [9]

7 Implementarea software a modulului de măsurare

În cele ce urmează se va prezenta în detaliu implementarea software a sistemului de măsurătoare. Implementarea s-a realizat utilizând microcontroller-ul NXP LPC2294 pentru controlul și cofigurarea senzorului LIDAR. Nucleul ARM7TDMI pe care îl are în componență, rulează la o frecvență de tact de 58.9824 MHz, iar din punct de vedere al periferiei, aceasta este configurată la o frecvență de tact de 14.7456 MHz.

Acest fapt se datorează PLL-ului (**P**hase **L**ocked **L**oop) care multiplică frecvența de tact generată de oscilatorul cu quartz de valoare 14.7456 MHz în vederea obținerii frecvenței de tact a nucleului de 58.9824 MHz.

Codul sursă rulat de microcontroller este realizat în limbajul de programare „C”, iar mediul de dezvoltare folosit este „Keil uVision 5”. Acesta este împărțit în 4 module: modulul folosit pentru senzorul LIDAR, modulul în care s-a implementat protocolul de comunicație I2C, modulul folosit pentru a implementa timerele software și întreruperile și modulul de comunicație serial [14].

7.1 Drivere

Un driver este un program software a cărui principal scop este de a acționa ca și o interfață între o componentă hardware și o componentă software. Acest driver controlează dispozitivele conectate la canalele I/O(**I**nput/**O**utput) ale unui sistem de operare sau la o unitate centrală de procesare [15].

7.2 Microcontroller-ul NXP LPC2294

Partea cea mai importantă din microcontroller-ul LPC2294 este unitatea centrală de procesare (CPU). Aceast procesor face parte din familia ARM7. Setul de instrucțiuni este unul foarte mic, performanța acestui procesor este ridicată și, împreună cu un consum scăzut, fac ca acest CPU să fie foarte folosit.

În interiorul acestuia se găsește pipeline-ul folosit pentru instrucțiuni. Tipul de pipeline folosit pe ARM7 este în trei etape.

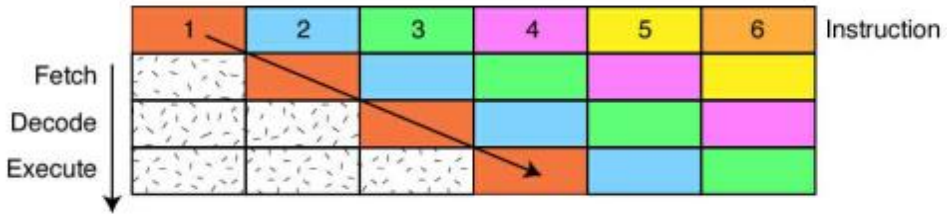


Figura 7-1. Instruction pipiline ARM7 [17]

Acest tip de pipeline reprezintă cea mai simplă formă de pipeline, neavând probleme de hazard, cum ar fi “read-before-write”. Pipeline-ul are nivele hardware independente care execută o instrucțiune în timp ce decodează o a doua instrucțiune și aduce o a treia intrstrucțiune. Pipeline-ul mărește viteza throughput-ului procesorului,

astfel majoritatea instrucțiunilor putând fi executate într-un singur ciclu. Acest procesor este foarte eficient în cazul codurilor sursă liniare, fiind puțin mai încet în cazul în care, în codul sursă, se găsesc mai multe instrucțiuni de salt.

Procesorul este bazat pe o arhitectură de tip load-and-store. Acest tip de arhitectură presupune aducerea datelor din memorie în setul de regiștrii pentru a putea procesa datele. Instrucțiunile de procesare a datelor sunt executate, iar apoi datele sunt stocate din nou în memorie.

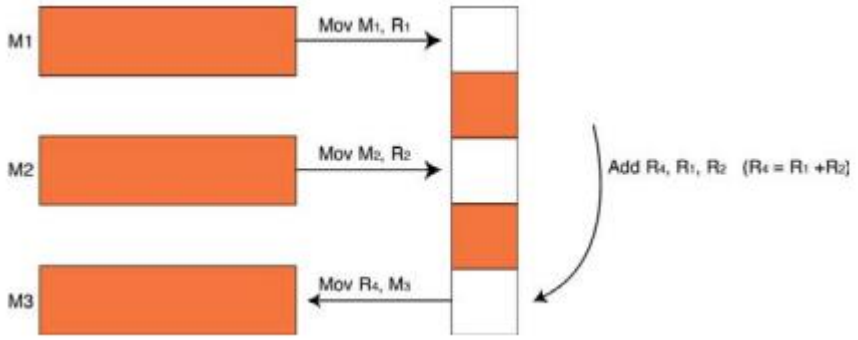


Figura 7-2. Load-and-store arhitecture [17]

Setul central de regiștrii este alcătuit din 16 regiștrii (R0-R15). Fiecare registru este pe 32 de biți, regiștrii R0-R12 neavând o funcție anume, fiind posibilă folosirea acestora în funcție de necesitate. Registrul R13 este folosit ca și stack pointer, registrul R14 este folosit ca și link register, iar registrul R15 este folosit ca și program counter.

7.3 Inițializarea modulelor

În momentul în care procesorul este alimentat sau în momentul în care acesta se resetează, fie că este vorba de un reset controlat, fie că mecanismul Watchdog forțează resetarea procesorului, acesta execută o secvența de inițializare a tuturor modulelor, resetând totodată și valorile din regiștrii care alcătuiesc setul central de regiștrii. Din acest motiv a trebuit să implementez anumite funcții de inițializare a sistemului, care se execută o singură data, la alimentarea dispozitivului sau după un reset.

Funcțiile responsabile pentru inițializarea senzorului LIDAR, a interfeței seriale și a timerelor software sunt prezentate în cele ce urmează.

Inițializarea senzorului presupune stabilirea comunicației dintre senzor și procesor, după standardele protocolului I2C. Această funcție configurează pinul P0.2 ca fiind SCL și pinu P0.3 ca fiind SDA. Această configurație v-a fi prezentată în detaliu în capitolul în care se prezintă protocolul de comunicație I2C. Regiștrii I2CONCLR și I2CONSET sunt folosiți pentru a seta și pentru a șterge biții din registrul de control. Regiștrii I2SCLH și I2SCLL sunt folosiți pentru a configura rata de transmisie a biților pe magistrala I2C. Formula de calcul pentru rata biților depinde și de frecvența oscilatorului de cuarț al procesorului. Pentru a calcula valoarea exactă a ratei, se aplică formula: Bit Rate = Pclk/(I2SCLH+I2CSLL).

Cod 7-1. Secvența de cod pentru inițializarea senzorului LIDAR

```
void I2C_init()
{
PINSEL0 |= 1<<4; /* Set P0.2 pin as SCL */
PINSEL0 &= ~(1 << 5);
PINSEL0 |= 1<<6; /* Set P0.3 pin as SDA */
PINSEL0 &= ~(1 << 7);
I2CONCLR=0xFF;
I2SCLH=0x1F;
I2SCLL=0xF;
I2CONSET |=1<<6;
}
```

Pentru a inițializa interfața UART trebuiesc configurați parametrii comunicației cât și pinii microcontroller-ului responsabili cu comunicația serială. Configurația este setată pe 8 biți, fără paritate, cu un bit de stop și cu 9600 baud rate. Pentru a realiza această configurație, se activează pinii de Rx și Tx ai microcontroller-ului și se scriu în regiștrii valorile corespunzătoare configurației alese. Implementarea în detaliu a comunicație UART va fi prezentată în următorul subcapitol.

Regiștrii pe care a trebuit să îi configurez pentru a putea realiza comunicația serială sunt:

- U0LCR (**L**ine **C**ontrol **R**egister) - acest registru este responsabil de controlul liniei de comunicație serială.
- U0DLL și U0DLM - acești regiștrii reprezintă byte-ul cel mai semnificativ și byte-ul cel mai puțin semnificativ ai baud rate-ului. În funcție de valoare setată în acești regiștrii, se modifică rata de transfer a biților pe magistrala serială.

Cod 7-2. Secvența de cod pentru inițializarea interfeței seriale UART

```
void initUART0(void)
{
    PINSEL0 = 0x5;
    U0LCR = 3 | (1<<7) ;
    U0DLL = 96;
    U0DLM = 0;
    U0LCR &= 0x0F;
}
```

7.4 Interfața serială UART

Interfața serială UART este un dispozitiv hardware folosit pentru comunicația serială asincronă, în care formatul de transmisie a datelor și timpul de transmisie a datelor poate fi configurat. De obicei, dispozitivul UART este un dispozitiv integrat, folosit pentru comunicația serială dintre un calculator și un dispozitiv periferic sau dintre un calculator și un sistem încorporat.

Pentru a transmite datele, interfața UART de transmisie preia bytes de date și îi transmite bit cu bit. La recepție, o altă interfată UART preia biții transmiși și îi reasamblează în bytes compleți.

Formatul de transmisie a datelor este următorul:

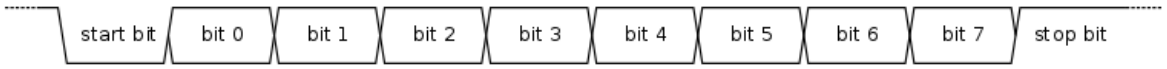


Figura 7-3. UART data format

Atât timp cât nu este transmisie pe linie, aceasta e ținută în state-ul de high voltage. În momentul în care se primește bit-ul de start, receiver-ul este anunțat că urmează să se transmită date. Se trimit între 5 și 9 biți, în funcție de cum e configurată interfața UART. După ce acești biți au fost transmiși, se primesc unul sau doi biți de stop. Aceștia au rolul de a anunța receptorul că toți biții au fost transmiși cu succes.

Pentru a putea realiza comunicația dintre două interfețe seriale UART, acestea trebuiesc configurate identic. Viteza de transmisie a biților, lungimea caracterelor, paritatea și biții de stop trebuie să aibă aceleași valori.

Microcontroller-ul LPC2294 suportă în momentul de fața două interfețe UART.

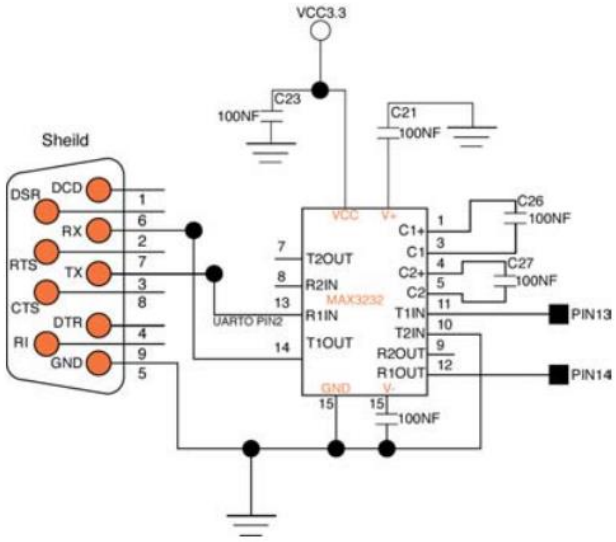


Figura 7-4. Arhitectura UART LPC2294 [18]

Pentru a putea comunica cu un calculator, trebuie inițializată interfața serială a microcontroller-ului. Această inițializare a fost prezentată anterior.

Pentru a putea transmite caractere pe interfața serială, datele trebuiesc scrise în registrul THR (**T**ransmit **H**olding **R**egister). THRE este un flag cu ajutorul căruia se poate verifica dacă stiva ce se folosește pentru transmiterea de caractere este goală. În registrul U0LSR este prezent statusul de transmisie. Dacă acesta este pe 1, se poate iniția transmisia, dar doar dacă stiva de date este goală.

Cod 7-3. Secvența de cod pentru transmiterea unui caracter pe interfața UART

```
uint8_t U0Write(char data)
{
    while ( !(U0LSR & THRE ) ); /* Wait till the THR is empty */
    U0THR = data;
    return data;
}
```

Pentru a primi date pe interfața serială, se așteaptă până în momentul în care în registrul RDR (**R**eceive **D**ata **R**egister) există date. Acest registru ocupă același zonă de memorie cu registrul THR. Pentru a face diferența dintre cei doi regiștrii, microcontroller-ul verifică statusul biților de transmisie și recepție. Dacă bit-ul de transmisie (Tx) este activ, se folosește registrul de transmisie de date, iar dacă bit-ul de recepție (Rx) este activ, se folosește registrul de recepție de date.

Cod 7-4. Secvența de cod pentru recepția unui caracter pe interfața UART

```
char U0Read(void)
{
    while( !(U0LSR & RDR ) ); /* Wait till any data arrives into Rx
FIFO */
    return U0RBR;
}
```

Aceast tip de interfață serială este foarte utilă deoarece poate fi configurată în mai multe moduri. În cazul în care se utilizează două dispozitive cu putere de procesare de date ridicată, interfața serială poate fi configurată la o viteză ridicată, însa, dacă se folosesc dispozitive mai puțin performante, interfața poate fi de asemenea configurată să funcționeze la viteze reduse.

7.5 Implementarea protocolului de comunicație I2C

Protocolul de comunicație I2C este un protocol de comunicație sincron, capabil să realizeze comunicația dintre unul sau mai multe dispozitive de tip master și unul sau mai multe dispozitive de tip slave. Cel mai des este folosit la conectarea dispozitivelor de tip slave la un microcontroller. Pentru a fi folosit eficient, este important ca dispozitivele de tip slave să se afle în imediata apropiere a dispozitivelor de tip master.

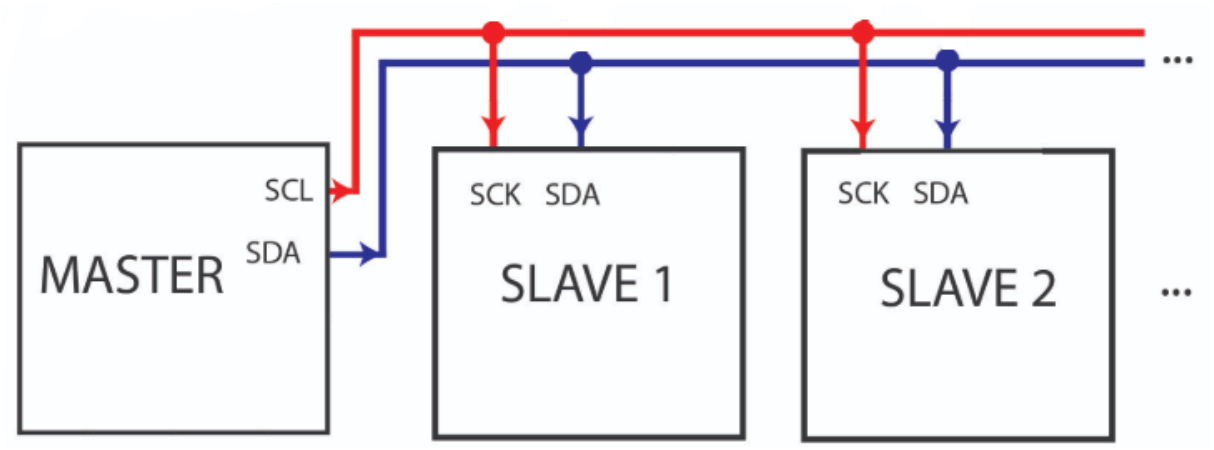


Figura 7-5. Conectarea dispozitivelor la magistrala I2C [18]

Pentru a realiza comunicația dintre master și slave, se folosesc liniile SCL (**S**erial **C**lock **L**ine) și SDA (**S**erial **D**ata **L**ine).

Senzorul LIDAR folosit de către mine este capabil să se conecteze la un dispozitiv master folosind o conexiune de tip 2-wires (sunt folosite liniile SDA și SCL). Poate fi conectat doar ca și dispozitiv de tip slave, fiind controlat de un dispozitiv master, în cazul meu, microcontroller-ul. Frecvența maximă suportată de acest dispozitiv este de 400kHz. In funcție de această frecvență, se calculează și tact-ul clock-ului din registrul SCL.

Pentru a comunica cu master-ul, dispozitivul slave trebuie să aibă o adresa, în funcție de care dispozitivul master știe cui trebuie să îi transmită datele. Adresa default a senzorului LIDAR este 0x62 dar aceasta poate fi configurată cu ajutorul registrilor I2C_ID_HIGH și I2C_ID_LOW. În implementarea facută de către mine s-a considerat adresa 0x62 pentru senzorul LIDAR.

Acțiunile cele mai importante care se pot realiza pe magistrala I2C sunt de scriere și de citire. Indiferent de acțiune, protocolul impune mai întai o scriere pe magistrală de la master la slave, iar mai apoi fiind posibilă și executarea unei citiri.

Protolul I2C al senzorului LIDAR nu fuctioneaza pe principiul “repeated start conditions”, care ar presupune că dispozitivul nu are nevoie de o condiție de oprire înaintea unei noi condiții de start. Astfel, au trebuit implementate funcții de inițiere de comunicație și de oprire de comunicație.

7.5.1 Transmisia datelor de la master la slave

Pentru a realiza o scriere pe magistrala I2C, dispozitivul master inițiază transferul de date prin stabilirea condiție de start, care este reprezentată printr-o tranziție a linie SDA din high în low, atât timp cât linia de SCL rămâne high. Primul byte care este setat este byte-ul de adresă, care constă din 7 biți de adresă și un bit care reprezintă operațiunea ce va avea loc. În cazul în care acel bit este 0, operațiunea este de scriere, iar în cazul în care bit-ul este 1, se va executa o operație de citire. Operația de scriere este folosită ca și primă etapă atât în cazul în care se dorește o scriere, cât și daca se dorește o citire de date. În cazul în care biții de adresă corespund cu adresa dispozitivului cu care se dorește inițierea comunicației, dispozitivul slave raspunde prin setarea liniei SDA pe low și prin transmiterea unei confirmări (acknowledge). În acest moment, toate dispozitivele de tip slave rămân în modul de așteptare, dispozitivul care a fost selectat anterior fiind singurul care comunică pe magistrală cu dispozitivul master.

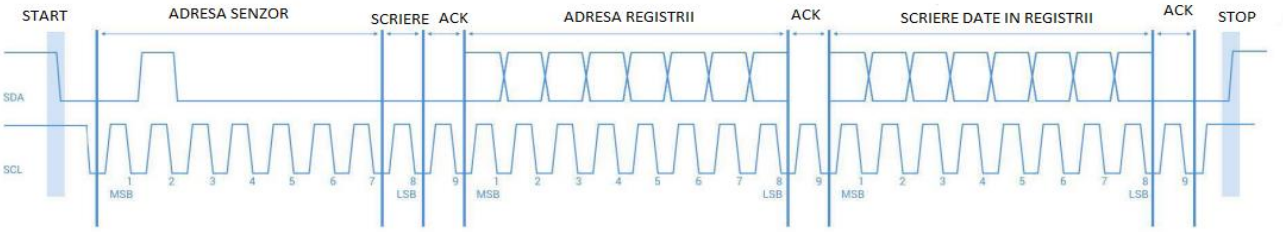


Figura 7-6. Inițierea secvenței de scriere de date [9]

Pentru inițierea secvenței de start, se setează liniile SDA și SLC cu ajutorul registrului I2CONSET, iar mai apoi trebuie scris în registrul I2DAT adresa dispozitivului slave cu care se dorește comunicarea, așteaptându-se confirmarea de la acesta.

Cod 7-5. Secvența de cod pentru inițierea condiției de start

```
void Lidar_start_condition()
{
    I2CONSET |= 1 << 5;
    while (I2STAT != 8); /* Wait for ack */
    I2CONCLR=0x28;/* Address shift and write */
    I2DAT = (LIDAR_ADDRESS << 1);
    while (I2STAT != 0x18);   /* Wait for ack */
}
```

După executarea secvenței de start, în cazul executării unei operații de scriere, datele se transmit în blocuri de 8 biți urmate de căte un bit de acknowledge, primit de la dispozitivul slave către care se transmit datele. Aceste date se transmit pe magistrala

I2C pe linia SDA. Această linie trebuie să execute tranziții atât timp cât linia SCL este în starea de low, iar atât timp cât linia de SCL este HIGH, linia SDA trebuie să rămână stabilă.

Pentru a iniția o scriere, se execută secvența de start, apoi se copiază în registrul de date I2DAT adresa dispozitivului de tip slave către care se dorește transmiterea de date, se așteaptă confirmarea acestuia, apoi se copiază datele ce urmează să fie transmise în registrul I2DAT. În cazul în care datele au fost transmise cu succes, se primește din nou un acknowledge de la dispozitivul slave, urmând ca dispozitivul master să inițieze secvența de stop.

Cod 7-6. Secvența de cod pentru scrierea de date pe magistrala I2C

```
uint8_t LidarWrite_8bits(uint8_t address, uint8_t data,uint8_t
numBytes)
{
    Lidar_start_condition();
    I2DAT=address;
    I2CONCLR=0x8;
    if(wait_for_response(0x28)==0) /* Wait for ack */
    {
       I2CONCLR=0x28;
       I2DAT = data;
       if(wait_for_response(0x28)==0)
       {
          I2CONCLR=0x28;
       }
       else
       {
            printf ("Ack2 in write not received." );
       }
    }
    else
    {
         printf ("Ack1 in write not received." );
    }
    Lidar_stop_condition();
    return 0x00;
}
```

În cazul în care transmiterea de date s-a realizat cu succes, dispozitivul master trebuie să inițieze o secvență de stop prin care oprește comunicația cu dispozitivul de tip slave și, totodată, permite din nou accesul altor dispozitive pe magistrală.

Cod 7-7. Secvența de cod pentru inițierea condiției de stop

```
void Lidar_stop_condition()
{
    I2CONSET=0x10;
    I2CONCLR=0x8;
}
```

7.5.2 Citirea datelor primite de la slave

Pentru a putea realiza o citire de date, prima dată trebuie executată o secvență de scriere la adresa dispozitivului slave. Această scriere trebuie executată în același mod în care a fost prezentată anterior. După ce secvența de scriere a fost realizată, dispozitivul master transmite un bit de stop după prima secvență de date din operația de scriere, urmat de o nouă secvență de start, care va conține adresa dispozitivului de tip slave de la care se va face citirea, urmată de un bit setat pe 1 care reprezintă operația de citire. Dacă dispozitivul slave trimite confirmarea către dispozitivul master, acesta inițiază secvența de citire de blocuri de câte 8 biți de date. După fiecare bloc de date, trebuie trimisă o confirmare de citire. Fiecare citire de bit se face pe frontul crescător al liniei SCL.

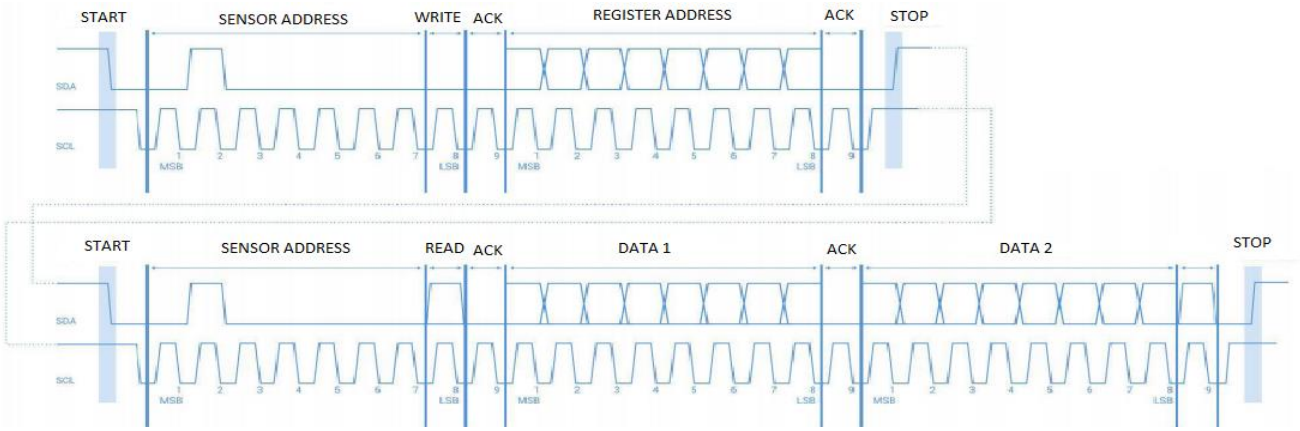


Figura 7-7. Inițierea secvenței de citire de date [9]

Pentru a executa o operație de citire, prima dată se pornește comunicația pe magistrala I2C, apoi se scrie în registrul de date adresa dispozitivului de tip slave de la care se va face citirea, iar după adresă urmeaza un bit de 1 care reprezintă operația de scriere. Următorul byte reprezintă adresa registrului din care se va face citirea. După această adresă, urmează un bit de stop apoi din nou o secvența de start, urmată de un

număr de bytes de date citite din registrul dispozitivului slave. Toate datele se citesc în registrul I2DAT. Fiecare bloc de date citit este urmat de o confirmare. În cazul în care această confirmare nu este primită, se încearcă din nou transmisia ei, iar dacă nici a doua oară nu se primește confirmarea, citirea este oprită.

Cod 7-8. Secvența de cod pentru citirea datelor

```
void Lidar_stop_condition()
{
uint8_t LidarRead_8bits(uint8_t address,uint8_t numBytes)
{
    uint8_t data_read;
    Lidar_start_condition();
    I2DAT=address;
    I2CONCLR=0x8;
    if(wait_for_response(0x28)==0) /* Wait for ack */
    {
      /* Stop */
      I2CONSET=0x10;
      I2CONCLR=0x8;
      /* Start */
      I2CONSET |= 1 << 5;
      if(wait_for_response(0x08)==0) /* Wait for ack */
      {
           /* Read from address*/
           I2CONCLR=0x28;
           I2DAT = (LIDAR_ADDRESS << 1) | 1;
           /* Wait for ack */
        if(wait_for_response(0x40)==0)
        {
          I2CONCLR=0x28;

          /* Wait for ack */
          if(wait_for_response(0x58)==0)
          {
          /* Save data */
            data_read = I2DAT;
          }
```

```
            else
            {
            if(DEBUG_MODE == 1)
              printf ("Ack4 in read not received." );
            }
          }
          else
          {
            if(DEBUG_MODE == 1)
              printf ("Ack3 in read not received." );
          }
        }
        else
        {
          if(DEBUG_MODE == 1)
            printf ("Ack2 in read not received." );
        }
    }
    else
    {
      if(DEBUG_MODE == 1)
        printf ("Ack1 in read not received." );
    }
    Lidar_stop_condition();
    return data_read;
}
```

7.5.3 Mecanismul de tratare al acknowledge-urilor

Folosind protocolul de comunicație I2C, trebuie implementat un mecanism de tartare a confirmărilor primite, pentru a asigura că transmiterea și primirea datelor se va face fără a pierde blocuri de date importante. Pentru aceasta am implementat o serie de funcții de tratare de confirmări pentru funcțiile de scriere și citire de date din regiștrii dispozitivului de tip slave.

În cazul operației de scriere de date în regiștrii dispozitivului de tip slave, sunt necesare 3 tipuri de confirmări. Prima confirmare, primită ca urmare a confirmării adresei slave-ului, a doua confirmare se primește după confirmarea adresei registrului în care se scriu date, iar următoarele confirmări se primesc la fiecare bloc de date scris

în registrul respectiv. Pentru a asigura că operația de scriere se execută cu succes, am implementat următorul mecanisc de tartare al acknowledege-urilor:

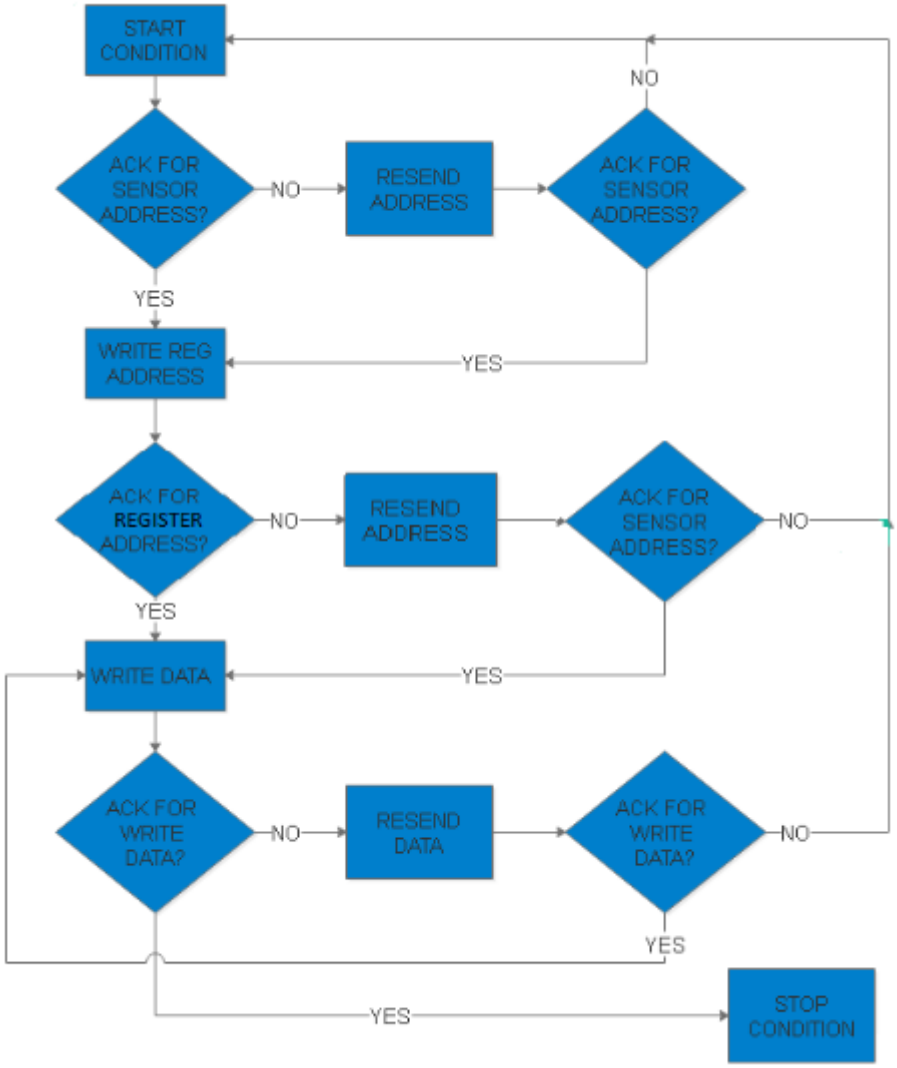


Figura 7-8. Mecanism tratare acknowledge-uri pentru scriere

Pentru implementarea mecanismului de tratare a confirmărilor pentru operaţia de citire, au fost folosite aceleaşi funcţii pentru partea de scriere de adresă a dispozitivului slave şi pentru scrierea adresei registrului din care se face citirea, iar pentru partea de citire a fost implementată altă functie pentru tratarea confirmărilor primite în urma citirii blocurilor de date. Această implementare are urmatorul flow:

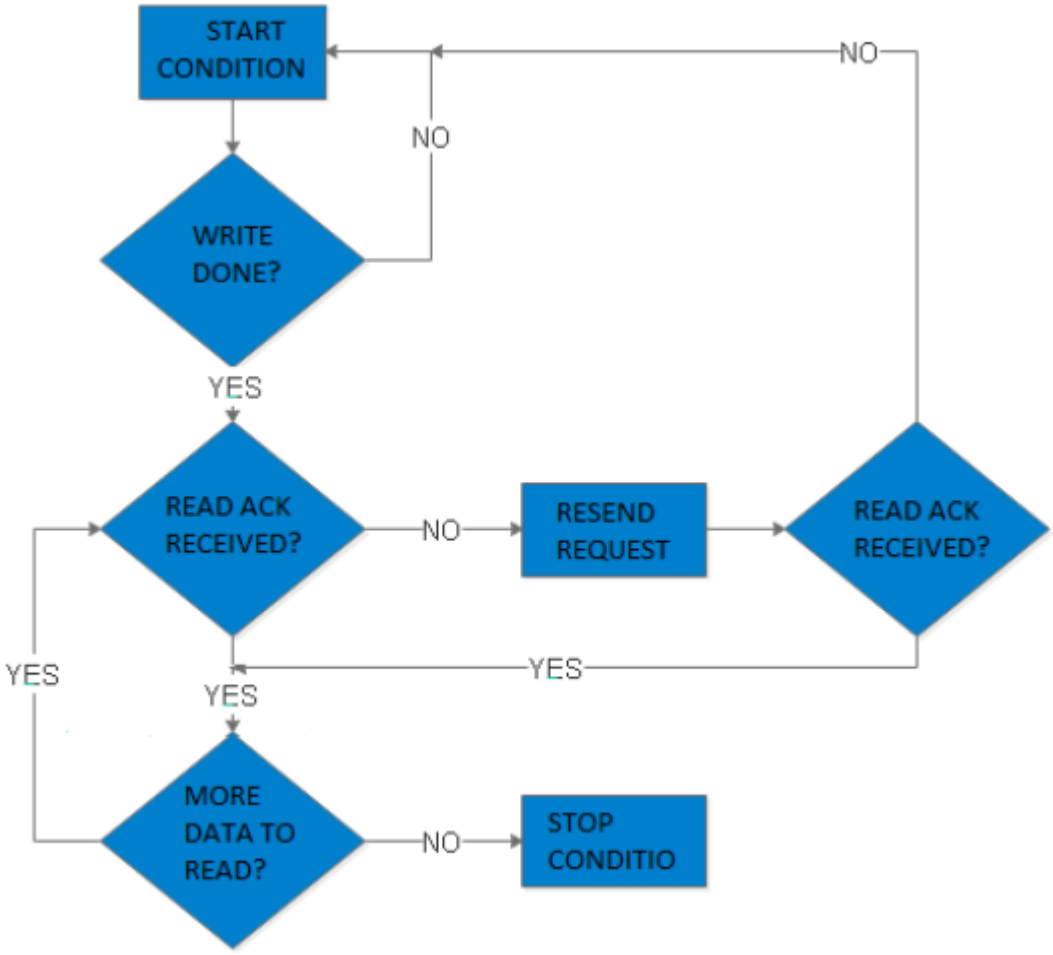


Figura 7-9. Mecanism tratare acknowledge-uri pentru citire

7.5.4 Măsurarea distanţei cu ajutorul senzorului LIDAR

Pentru a putea executa o măsurătoare cu ajutorul senzorului LIDAR, am implementat o funcţie care execută citiri din regiştrii în care se salvează datele citite. În urma executării unei măsurători, datele sunt salvate în regiştrii LAST_DELAY_HIGH şi LAST_DELAY_LOW. Aceşti doi regiştrii sunt pe 8 biţi fiecare şi pentru a putea afla valoarea reală citită trebuie concatenaţi cei doi bytes. Valoarea măsurată este în cm.

Având în vedere că măsurarea distanţei presupune o citire dintr-un registru, trebuie respectat tot procesul de citire de date, de la condiţia de start, la trimiterea adresei slave-ului şi a registrului din care se face citirea, până la citirea blocurilor de date. Pentru a putea implementa această operaţie de măsurare, am folosit funcţiile descrise anterior.

Cod 7-9. Secvența de cod pentru executarea unei măsurători

```
uint16_t read_distance()
{
    uint8_t lidar_data;
    uint8_t lidar_data_low;
    uint8_t lidar_data_high;
    uint8_t lidar_address;
    uint16_t lidar_response;
    /* Write 0x04 in reg 0x00*/
    lidar_address = 0x00;
    lidar_data = 0x04;
    lidar_response = LidarWrite_8bits(lidar_address,lidar_data,1);
    lidar_address = 0x01;
    while(LidarRead_8bits(lidar_address,1)==0x01);
    lidar_address = 0x14;
    lidar_data_high = LidarRead_8bits(lidar_address,1);
    if(DEBUG_MODE == 1)
      printf ("High data: %02X\n",lidar_data_high );
    lidar_address = 0x15;
    lidar_data_low = LidarRead_8bits(lidar_address,1);
    if(DEBUG_MODE == 1)
      printf ("Low data: %02X\n",lidar_data_low );
    /* Compute distance */
    lidar_response = lidar_data_high;
    lidar_response = lidar_response << 8 ;
    lidar_response += lidar_data_low;
    return lidar_response;
}
```

7.6 Configurarea senzorului LIDAR

Pentru a folosi în cât mai multe moduri acest dispozitiv, a fost implementată o funcţie de configurare a senzorului LIDAR. Această funcţie suportă mai multe moduri de configurare, dintre care 6 sunt predefinite şi un mod ce presupune o configuraţie dinamică, permiţând configurarea senzorului în funcţie de necesităţi, fiind luaţi în calcul mai mulţi factori.

Primul registru ce se poate configura este registrul ACQ_CONFIG_REG. Cu ajutorul acestui registru se poate activa terminarea rapidă de măsurătoare în cazul în care senzorul anticipează că distanţa măsurată este mai mare decât cea mai mare valoare ce poate fi măsurată.

Al doilea registru ce se poate configura este registrul THRESHOLD_BYPASS. În cazul în care se foloseşte valoarea default a registrului, distanţa maximă ce se poate măsura ţine cont de valoarea de vârf, de zgomotul şi de puterea semnalului. În cazul în care se modifică valoarea din acest registru, orice distanţă măsurată care este peste acel threshold, este ignorată.

Al treilea registru ce se poate configura este registrul OUTER_LOOP_COUNT. Valoare default din acest registru este 0x01. Această valoare reprezintă numărul de sample-uri. Cu cât această valoare este mai ridicată, cu atât timpul de măsurare va creşte, dar, totodată, precizia de măsurare este îmbunătăţită considerabil. În cazul în care se execută mai multe sample-uri pentru o măsurătoare, în registrul MEASURE_DELAY se configurează durata de timp dintre două sample-uri consecutive. În funcţie de valoarea salvată în acest registru, frecvenţa sample-urilor este mai mare sau mai mică.

Un alt registru ce poate fi configurat este registrul VELOCITY. În acest registru se salvează valoarea diferenţei dintre sample-ul curent şi sample-ul precedent, dar doar în cazul în care, în registrul OUTER_LOOP_COUNT se salvează o valoare mai mare de 0x01.

Pentru a configura adresa I2C a senzorului, trebuiesc aduse mai multe modificări unor regiştrii. Pentru început, trebuie citită adresa curentă a senzorului din regiştrii UNIT_ID_HIGH şi UNIT_ID_LOW. A doua etapă din configurarea adresei constă în scrierea noii adrese în regiştrii I2C_ID_HIGH şi I2C_ID_LOW. Ultima etapă constă în scrierea noii adrese cu ajutorul registrului I2C_SEC_ADDR şi dezactivarea adresei default din registrul I2C_CONFIG.

Ultimul registru ce se poate configura este registrul POWER_CONTROL. Prin acest registru se poate configura modul de funcţionare a senzorului. Valoarea default din acest registru este 0x80. Această valoare presupune că senzorul, în momentul în care acesta nu este folosit, se dezactivează, scăzând consumul cu aproximativ 40 mA. Tot cu ajutorul acestui registru se poate configura şi un mod de “always on” sau “sleep”.

Din punct de vedere al configuraţiilor, au fost implementate 7 moduri de configurare.

Prima configuraţie reprezintă modul de operare default. Acesta are ca şi principale caracteristici viteză medie de procesare şi o performanţă balansată. Pentru această configuraţie default, în regiştrii SIG_COUNT_VAL, ACQ_CONFIG_REG, REF_COUNT_VAL şi THRESHOLD_BYPASS au fost scrise valorile default ale

acestora. Valoarea default a registrului SIG_COUNT_VAL este 0x80 și reprezintă numărul maxim de achiziţii făcute în urma unei măsurători. În registrul ACQ_CONFIG_REG se scrie valoarea 0x08 şi reprezintă valoarea default a registrului. În registrul folosit ca referinţă pentru numărul de achiziţii, REF_COUNT_VAL, se scrie de asemenea valoarea default. Threshold-ul nu este setat, astfel că măsurătorile se fac în funcţie de valoarea maximă detectată.

Cod 7-10. Secvența de cod pentru configurarea default a senzorului

```
case default:
             sigCountMax     = 0x80;
             acqConfigReg    = 0x08;
             refCountMax     = 0x05;
             thresholdBypass = 0x00;
break;
```

În următoarele moduri de configurare am prezentat în detaliu cum influenţează fiecare valoare scrisă în aceşti regiştrii, performanţele senzorului LIDAR.

Primul mod de configurare implementat are ca principale caracteristici măsurări pe distanţe mici dar cu viteză mare. Acest mod este foarte util în cazul în care precizia de măsurare nu este atât de importantă şi nici nu se fac măsurători pe distanţe lungi. Având în vedere că măsurătorile se execută cu o viteză mare, există şanse foarte mari ca undele laser transmise de senzorul LIDAR să nu se reflecte şi să nu poată fi receptate. Pentru a putea face posibilă această configuraţie, trebuie modificat numărul maxim de achiziţii. Acest lucru se face modificând valoarea din regiştrii SIG_COUNT_VAL şi REF_COUNT_VAL.

Cod 7-11. Secvența de cod pentru modul 1 de configurare a senzorului

```
case mode1:
             sigCountMax     = 0x1d;
             acqConfigReg    = 0x08;
             refCountMax     = 0x03;
             thresholdBypass = 0x00;
break;
```

Al doilea mod de configurare presupune modificarea caracteristicilor senzorului, astfel încat să se facă măsurători pe o distanţă medie cu viteză medie, iar, în cazul în care distanţa măsurată este mică, viteza creşte. Acest lucru este posibil deoarece se activează

terminarea rapidă a detecţiei. Acurateţea acestui mod de configurare este de asemenea scazută.

Cod 7-12. Secvența de cod pentru modul 2 de configurare a senzorului

```
case mode2:
            sigCountMax     = 0x80;
            acqConfigReg    = 0x00;
            refCountMax     = 0x03;
            thresholdBypass = 0x00;
break;
```

Al treilea mod de configurare este folosit pentru a realiza măsurători la distanţă mare. Acest tip de configuraţie este cea mai lentă dintre toate şi presupune o acurateţe scăzută, dar, distanţa de măsurare este foarte mare. Acest lucru este posibil dacă în registrul SIG_COUNT_VAL se scrie valoarea maximă.

Cod 7-13. Secvența de cod pentru modul 3 de configurare a senzorului

```
case mode3:
            sigCountMax     = 0xFF;
            acqConfigReg    = 0x08;
            refCountMax     = 0x05;
            thresholdBypass = 0x00;
break;
```

Un alt mod de configurare este cel în care senzitivitatea de detecţie este crescută, dar erorile de măsurare sunt mari. Acest lucru e posibil prin stabilirea unui threshold cu o valoare mai mica.

Cod 7-14. Secvența de cod pentru modul 4 de configurare a senzorului

```
case mode4:
            sigCountMax     = 0x80;
            acqConfigReg    = 0x08;
            refCountMax     = 0x05;
            thresholdBypass = 0x80;
break;
```

Dacă în modul anterior de configuraţie, eroarea de măsurare era mare, în cazul în care se doreşte ca eroarea de măsurare să fie cât mai mică, trebuie crescută valoarea threshold-ului, astfel încat zgomotul din mediul în care se execută măsurători să poată fi acoperit. Setandu-se o valoarea mare a threshold-ului, senzitivitatea scade.

Cod 7-15. Secvența de cod pentru modul 5 de configurare a senzorului

```
case mode5:
            sigCountMax     = 0x80;
            acqConfigReg    = 0x08;
            refCountMax     = 0x05;
            thresholdBypass = 0xB0;
break;
```

Ultimul mod de configurare a senzorului LIDAR este unul în care valorile din regiştrii pot fi setate după propriul plac. Este importantă o configuraţie foarte bună pentru a putea folosi cât mai bine acest senzor.

Din aceste configuraţii s-a realizat următorul tabel:

Tabel 7-1. Tabel comparaţie configuraţii

Mod configurare	**Viteză**	**Sensibilitate**	**Distanţă**	**Performanţă**	**Eroare măsurare**
Default	Medie	Medie	Medie	Medie	Medie
Modul 1	Mare	Medie	Mică	Medie	Medie
Modul 2	Mare	Medie	Medie	Medie	Medie
Modul 3	Mică	Mică	Mare	Medie	Medie
Modul 4	Medie	Mare	Medie	Mică	Mare
Modul 5	Medie	Mică	Medie	Mare	Mică

8 Funționarea sistemului de măsurare

În cele ce urmează am prezentat întreg parcursul programului, din momentul inițializării modulului până în momentul în care datele sunt transmise către calculator, prin interfața serială.

După alimentarea sistemului sau după un reset, se execută funcțiile de inițializare a modulelor. Dacă se dorește o scriere, prima dată se trimite secvența de start. În cazul în care aceasta a avut succes, se primește acknowledege-ul cu codul 0x08. În continuare, aplicația software trebuie să scrie adresa dispozitivului slave și să seteze bitul de scriere/citire pe 1, în registru de date I2DAT. Această operație de scriere va fi scrisă pe magistrala I2C și se va primi o confirmare de la slave. Dacă scrierea a fost realizată cu succes, se generează o întrerupere, iar în registrul de status se setează valoarea 0x18. În acest moment, legătura dintre master și slave a fost realizată, iar dispozitivul slave este pregătit să primească date. Pentru a transmite aceste date, se scrie în registrul I2DAT blocuri de câte 8 biți. Fiecare bloc va primi o confirmare de scriere după ce datele au fost scrise. În momentul în care se transmite confirmarea, este generată o întrerupere, iar în registrul de status se va regăsi valoarea 0x28 dacă scrierea s-a executat cu success. În cazul în care nu se primește confirmarea, se generează o întrerupere, în registrul de status setânduse valoarea 0x20, fiind nevoie o retransmitere de date. După ce toate datele au fost scrise, se inițiază o secvență de stop prin scrierea în regisrul de control a acestei comenzi.

Pentru a executa o operație de citire, condiția de start va fi aceași, însă, bitul de scriere sau citire trebuie setat pe 0. Fiecare bloc de date citit se salvează în registrul de date I2DAT și fiecare citire este confirmată cu statusul 0x28. După ce toate datele au fost citite, se inițiază secvența de stop, comunicația fiind oprită.

Flow-ul programului este prezentat în Figura 8-1**.**

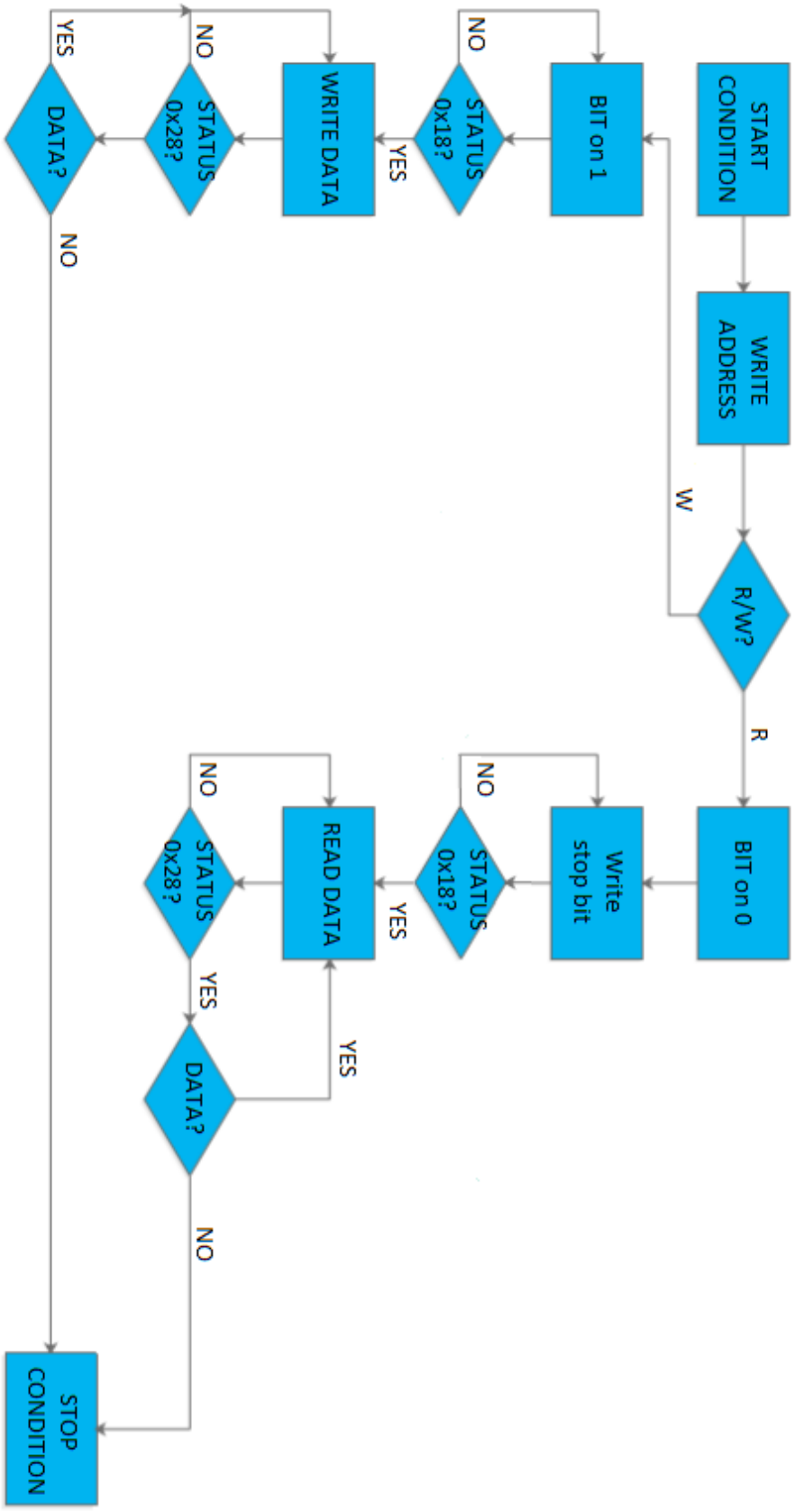


Figura 8-1. Control Flow Graph

9 Rezultate experimentale

După finalizarea proiectului, anumite teste experimentale au fost executate de către mine, urmând ca acestea să fie prezentate în acest capitol.

9.1 Distanțe măsurate și performanțele senzorului LIDAR

După ce senzorul LIDAR a fost conectat cu succes la placa de dezvoltare, am rulat codul cu fiecare configurație posibilă. Din aceste execuții a rezultat tabelul 9.1.

Din acest tabel, coloana de "viteză" reprezintă viteza cu care se face o măsurătoare. Aceasta este calculată din momentul în care este emisă o undă laser până în momentul în care unda este reflectată și recepționată.

Coloana "Distanță maximă" reprezintă distanța maximă ce poate fi măsurată cu ajutorul senzorului.

Coloana "Performanță" reprezintă o analiză proprie asupra performanțelor senzorului. Au fost luați în calcul toți factorii determinanți ai performanței sistemului.

Rezultatele experimentale sunt următoarele:

Tabel 9-1. Tabel performanțe senzor LIDAR

Mod configurare	**Viteză**	**Distanţă maximă**	**Performanţă**	**Eroare măsurare**
Default	5 µs	20 m	Medie	±3 cm
Modul 1	0,5 µs	4,5-5 m	Medie	±3 cm
Modul 2	0,5 µs	20 m	Medie	±5 cm
Modul 3	10 µs	40 m	Medie	±5 cm
Modul 4	5 µs	20 m	Mică	±10 cm
Modul 5	5 µs	20 m	Mare	±1 cm

Performanța senzorului a fost calculată ținând cont de viteză, distanța maximă ce poate fi măsurată și eroare de măsurare. Din punctul meu de vedere, cel mai satisfăcător mod de configurare este modul 5. În acest tip de configurație, măsurătorile se execută rapid, distanța maximă de măsurare, fiind suficient de mare pentru a putea măsura distanțe considerabile, însă, cea mai importantă caracteristică este "Eroarea de măsurare". Dacă în celelalte moduri de configurație, erorile de măsurare puteau să ajungă și la ±10 cm, în acest mod de configurare, eroarea maximă fiind de 1cm. Acest 1cm apare din cauza rezoluției senzorului care măsoară în centimetrii. O comparație detaliată a măsurătorilor este prezentată în tabelul 9.2.

Tabel 9-2. Tabel rezultate măsurători

Mod configurare	Distanță 20 cm	Distanță 40 cm	Distanță 100 cm	Distanță 500 cm
Default	22cm	41cm	102cm	503 cm
Modul 1	20 cm	41 cm	103 cm	1 cm
Modul 2	21 cm	42 cm	103 cm	505 cm
Modul 3	21 cm	38 cm	102 cm	504 cm
Modul 4	15 cm	31 cm	106 cm	493 cm
Modul 5	20 cm	40 cm	100 cm	501 cm

9.2 Verificarea implementării protocolului de comunicație I2C

Pentru a verifica funcționarea protocolului de comunicație I2C, liniile SDA și SCL au fost monitorizate static, cu ajutorul unui analizor logic.

Pentru executarea operației de scriere a unui bloc de date, sunt necesare 8 tranziții pe linia de clk, fiecare bit fiind transmis pe câte un front crescător al semnalului de clk. Fiecare tranziție a semnalului de clk durează aproximativ 30µs.

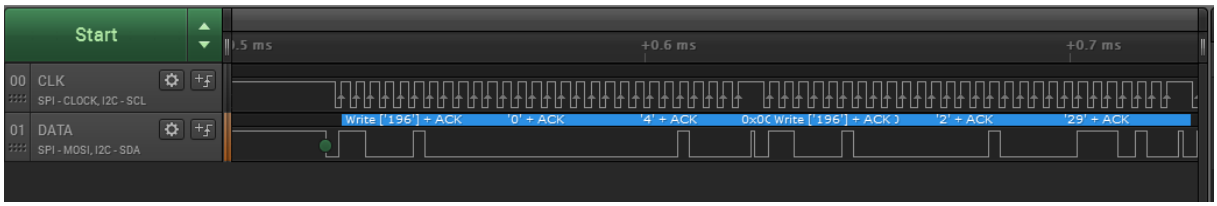


Figura 9-1. Captură analizor logic

10 Concluzii și perspective de dezvoltare

10.1 Concluzii

S-a realizat cu succes implementarea modulului de măsurare. Sistemul poate realiza atât măsurători la distanţă cât şi măsurători foarte precise. Comunicaţia sistemului cu un calculator este foarte utilă deoarece se poate testa în timp real funcţionarea programului.

Au fost implementate atât funcţii simple pentru comunicaţia serială şi pentru utilizarea timerelor software, cât şi funcţii mai complexe, folosite la implementarea protocolului de comunicaţie I2C cât şi funcţii de configurare a senzorului LIDAR.

În concluzie, partea practică a acestei lucrări este funcţională şi poate fi folosită atât la implementarea unui dispozitiv de mapare a spaţiului cât şi ca şi driver compatibil cu orice senzor cu laser capabil să comunice cu ajutorul protocolului I2C.

10.2 Sinteza contribuțiilor

În acest subcapitol sunt enumerate contribuţiile principale ale mele în realizarea acestei lucrări:

1. Am realizat conectarea la un calculator a sistemului, prin interfaţa UART.
2. Am conectat cu succes senzorul LIDAR la placa de dezvoltare Olimex.
3. Am implementat protocolul de comunicaţie dintre senzor şi placa de dezvoltare.
4. Am creat un prototip hardware pentru testarea senzorului LIDAR.
5. Am creat funcţii de măsurare a distanţei.
6. Am implementat cu succes funcţii de configurare a senzorului LIDAR.
7. Am generat documentația codului sursă cu ajutorul programului Doxygen.

10.3 Perspective de dezvoltare

Ca în orice proiect care presupune implementarea unui sistem embedded, există întotdeauna perspective de dezvoltare, prin urmare, şi proiectul implementat de către mine poate primii anumite îmbunătăţiri.

1. Prima parte ce ar putea fi îmbunătăţită ar fii implementarea hardware finală a dispozitivului de măsurare, deoarece a fost implementat doar un prototip hardware.
2. Se pot monta anumite motoare pentru a forma o turelă cu ajutorul căreia să se poată mişca senzorul pentru a putea face posibilă maparea spaţiului.
3. Se poate adăuga un modul wireless pentru a comunica mai uşor cu un calculator.
4. Implementarea unui mecanism de detecţie şi protecţie împotriva unor posibile erori hardware.
5. Implementarea unui mecanism eficient de tratarea erorilor primite în cazul în care nu se primesc acknowledge-uri. În prezent, în cazul în care nu se primeşte un acknowledge, se retransmit datele de la început.

În orice proiect se pot aduce îmbunătăţiri. Nici proiectul implementat de către mine nu duce lipsă de astfel de îmbunătăţiri posibile. Sper ca pe viitor să pot îmbunătăţii acest proiect cât mai bine şi să ajung la un produs final.

Referințe bibliografice

[1] K. Zhang, "A progressive morphological filter for removing nonground measurements from airborne LIDAR data", 2003.

[2] D. S. Hall, „High definition lidar system," [Interactiv]. Available: https://patents.google.com/patent/US7969558B2/en.

[3] A. Nastase, "Cartografie-Topografie", Bucuresti: Editura Didactica si Pedagogica, 1983.

[4] Technomic Publishing Company, "Ceramic Sensors", Lancaster, 1996.

[5] D. Vlad, „Senzori de temperatura," [Interactiv]. Available: https://www.scribd.com/doc/45979130/6-senzori-de-temperatura.

[6] M. Popa, "Suport de curs: Sisteme Incorporate".

[7] A. Dumitru, Mecatronica, Bucuresti: Universitatea Politehnica Bucuresti, 2003.

[8] „Principiul senzorului Hall," [Interactiv]. Available: http://www.e-automobile.ro/categorie-electronica/106-senzor-hall.html.

[9] „LIDAR Lite v3 Operation Manual and Tehnical Specification," [Interactiv]. Available: https://static.garmin.com.

[10] M. V. Micea, Introducere in Prelucrarea Numerica a Semnalelor, Timisoara, 2003.

[11] L. Prodan, Curs - Structuri tolerate la defecte.

[12] B. Vesely, "Fault Tree Analysis", NASA HQ.

[13] „Curs Fiabilitate - Ciontu Marian," [Interactiv]. Available: http://elth.ucv.ro/student1/Cursuri/Ciontu%20Marian/Fiabilitate/Curs%20Fiabilitate.pdf.

[14] Keil Software Inc., "IDE Keil uVision 5 Software".

[15] M. Marcu, "Curs - Proiectarea Driverelor", Universitatea Politehnica Timisoara.

[16] "LPC-H2294 schematic".

[17] NXP Semiconductors, "LPC21xx and LPC22xx User Manual", NXP Semiconductors, 2012.

[18] T. Martin, "The Insider's Guide To The Philips ARM7-Based Microcontrollers", Hitex, 2005.