



Autonomous Route and Mapping 2D - 3D Lidar Scanning

FINAL REPORT

CALIN DORAN

Table of Contents

Table of Figures	2
1. Introduction	3
2. Project Description.....	4
2.1 Application Screenshots.....	5
2.1.1 Command window with a warning.	5
2.1.2 Blank window due to No Data	6
2.1.3 Lidar Connection	7
2.1.4 OpenGL Window with LiDAR Data	8
2.1.5 Successful termination of the application	9
2.2 Problems Encountered	10
2.2.1 Software Issues	10
2.2.2 Hardware Issues.....	10
2.2.2 Covid-19 Lockdown and Working from Home.....	11
3. Conformity with the Original Design.....	12
3.1. Changes from the Original Design	12
4. Learning Outcomes	13
4.1 Technical	13
4.1.1 C++	13
4.1.2 SFML.....	13
4.1.3 Slamtec LiDAR SDK	14
4.2 Personal	15
5. Project Review	16
5.1 What was Achieved.....	16
5.2 What was not Archived.....	16
5.3 Future Features.....	17
Acknowledgements.....	18
Plagiarism Declaration	19

Table of Figures

Figure 1. Console Window, No Connection	5
Figure 2. Blank OpenGL Window	6
Figure 3. Console Window, Successful Connection.....	7
Figure 4. OpenGL Window, Rendering Data	8
Figure 5. Console Window, Successful Termination	9

1. Introduction

The purpose of this final report is to provide and reflect on the project as a whole and its development over the past months. The A.R.M. Lidar System is a Windows OS application that allows the user to view LiDAR data in a GUI built with SFML. There are not many applications like this on the Windows OS for developers or robot enthusiasts in the field of software development. There are some applications on Linux OS that are open source, however, they deal mainly with the mapping functionality of the LiDAR system. This means that on Windows OS it is indeed quite a niche market as most of the projects are data-driven applications based on Linux. The A.R.M. Lidar System set out with the aim of making something new, that hadn't been done before on Windows due to its less than ideal multithreaded performance in comparison to the many different Linux distro counterparts.

In section two, we will document the development process of the A.R.M Lidar System, and describe the project along with the final product and its goals and objectives. It will also provide and discuss any problems that were encountered throughout the project development cycle and how they were solved.

Section three will outline how the application has changed from the start of development and how it conforms with the initial proposed concept.

In section four the learning outcomes will be discussed, from a technical point of view and a personal one. This will be including any changes that would be made to the process if the project were started again.

Finally, in section five, this document will provide an overall review of the project and process its successes and failures, and proposed additional functionality, features and usability.

2. Project Description

This project, the A.R.M Lidar System, is an application built upon the RPLiDAR A2M8 scanner from Slamtec. The LiDAR can map objects and environments in a 2D point cloud map that can be stored to generate a 2D wireframe map of the area scanned, however, this can only be accomplished if connected to a computer set up to transform this data. Hardware similar to this LiDAR can be seen already in many regular or more advanced self-driving cars developed by most car manufactures. This technology is used in the parking sensors of newer car systems, usually describe as driver assistance. This application's intended purpose was to take full advantage of all of these functions and features of the provided scanner by Slamtec and to implement these key components into a native windows application using C++ and the SFML API with Slamtec's SDK.

The purpose of this application was to allow a user a set field of view generated from the LiDAR data that can help with avoidance, navigation and mapping of the environment up to a predefined distance of a few meters. This 3D representation of the LiDAR's 2D data is to give the user a view that is easily understood so that it can be used in a predefined set of functions that are within the key features of this application's design.

To do this we needed a fast and efficient framework, and as such C++ was and remains the best option as the core functionality of this application relies on the speed, quality and management of data taken from the LiDAR scanner. This means that the user will have a pleasant experience using this application and will not run into problems that require fast computation of data.

The application in its current form is quite restricted in several ways. It currently only displays the points of data in different coloured boxes that change in size depending on the distance to the LiDAR scanner. This was to be a field of boxes that were to be displayed constantly giving a sense of object permanence to the user, potentially making it easier to navigate around an environment. However, to do this the implementation of the A-star or (A*) was to be implemented to solve the overlapping box issue, but the developer did not have enough time to complete this task. The A-star algorithm is a graph traversal and path search algorithm, however, the one major drawback that the developer is aware of is that the algorithm could reduce the efficiency of the application by a huge amount. How this would affect our application is that it would store every node from the LiDAR to memory, this would have a huge cost to performance depending on the amount of memory the user has available and having to manage this data would take more time and as such it is still to be developed.

Due to the description above, the application currently cannot store this data in a way beneficial to a user in a functional or meaningful way regarding the mapping function, this also means that without this data to process it cannot currently provide assistance in which direction to move for the avoidance specification of the application.

2.1 Application Screenshots

2.1.1 Command window with a warning.

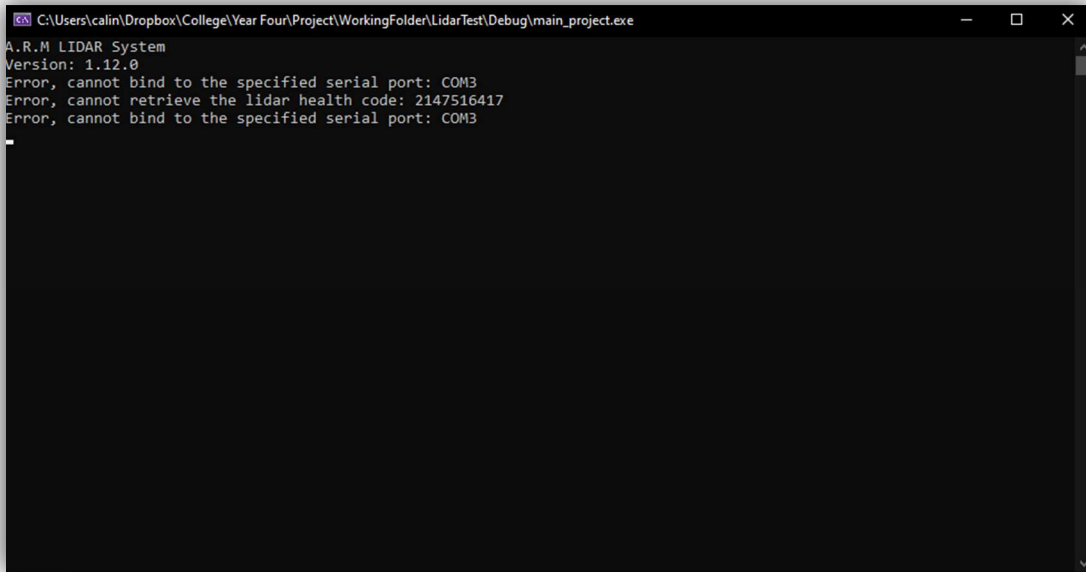


Figure 1. Console Window, No Connection

The application's console window shows that it is trying to find the LiDAR scanner on the COM3 port and that it is not getting the health check back from the LiDAR scanner. This allows us to show some functionality in exception management, it is set to show the error every second.

2.1.2 Blank window due to No Data



Figure 2. Blank OpenGL Window

As you can see, we cannot show data if there is none to display. This is resolved with a LiDAR health check once the LiDAR scanner is plugged into a computer. This allows the application to recognize the scanner via a health check of the LiDAR scanner that is searched on COM port three every second.

2.1.4 OpenGL Window with LiDAR Data

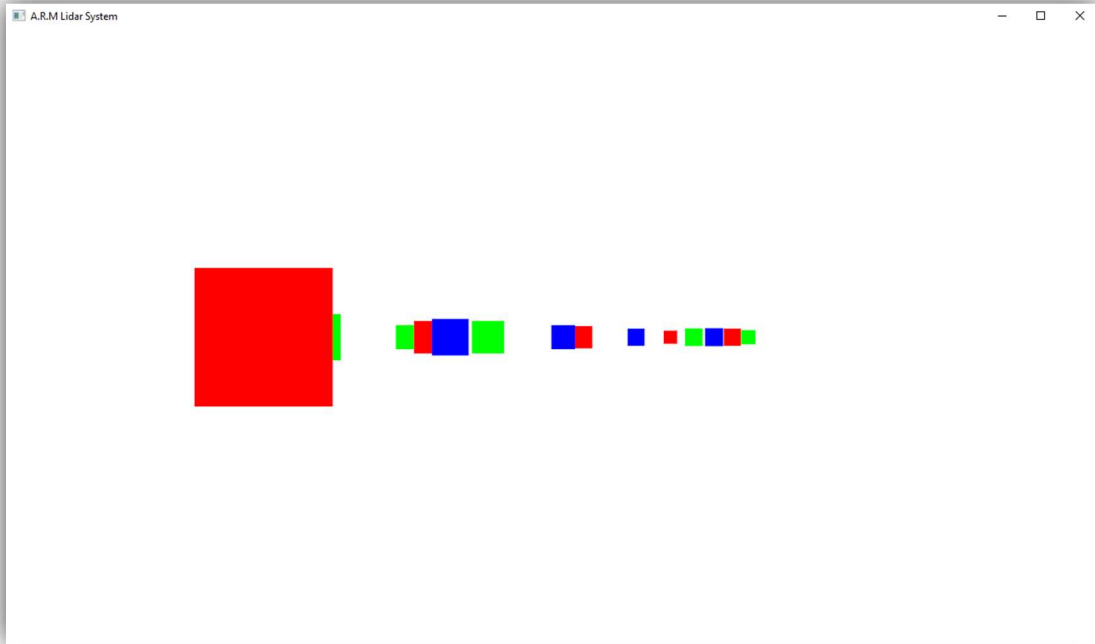
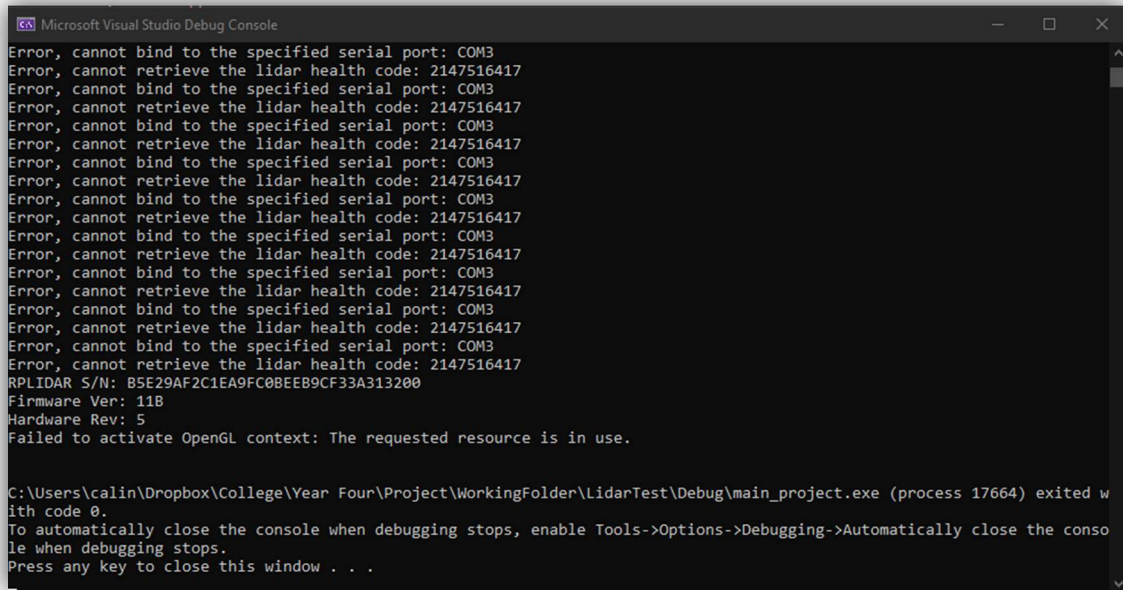


Figure 4. OpenGL Window, Rendering Data

On a successful connection, the OpenGL window will show our data that is to be displayed in the form of coloured squares that change size and position depending on the distance and angle from the LiDAR scanner. This gives the user the ability to “see” in a manner of a 2D plane represented by the coloured squares that we get from a specified field of view of the device. From here we get the scanners first point in its node array, we take thirty degrees either side from the LiDAR scanner, allowing for a full sixty degrees field of view for the user.

2.1.5 Successful termination of the application



```
Microsoft Visual Studio Debug Console
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
Error, cannot bind to the specified serial port: COM3
Error, cannot retrieve the lidar health code: 2147516417
RPLIDAR S/N: B5E29AF2C1EA9FC08EEB9CF33A313200
Firmware Ver: 11B
Hardware Rev: 5
Failed to activate OpenGL context: The requested resource is in use.

C:\Users\calin\Dropbox\College\Year Four\Project\WorkingFolder\LidarTest\Debug\main_project.exe (process 17664) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Figure 5. Console Window, Successful Termination

Successful termination of the application allows for continued testing and development of the applications functions and features that are to do developed in the future. This is a key feature as not needing to plug the device out and back in every test increases developmental productivity.

2.2 Problems Encountered

2.2.1 Software Issues

2.2.1.1 RPLIDAR standard SDK

The RPLIDAR standard SDK provides a static library for developers to integrate the SDK features into their existing projects. When developing using the SDK, developers are usually only required to use the external header files in the include folder to their source code. However, for VS developers they can also include the SDK's project into their project and set the related project dependencies as this is what was needed here. For Linux developers, all that is needed is to see the `simple_grabber`'s Makefile for detailed settings. This had proven difficult at the beginning of the project as developing on windows using the static library `c` uses `vc10 MD c's` runtime library which leads to many compilation failures with a lot of unpredictable behaviour, stifling the progress of the application and its development.

This combined with the rather steep learning curve of C++ means that the development of the application wasn't only complex and frustrating to the developer, but also meant that there was significant research needed to take advantage of the SDK in a newer version of our IDE (vs2019) roughly ten years from (vs2010) that was suggested by the documentation.

Other more frustrating problems showed very quickly as it suggested to use C++10 Standard, an extremely dated version of C++. In short, what was used to deal with this issue was the ISO C++14 Standard, as it maintained most of the requirements of the old version but still was compatible with the newest versions of the IDE (vs2019).

2.2.2 Hardware Issues

2.2.2.1 RPLiDAR A2M8 from Slamtec

During the development with the RPLiDAR A2M8 from Slamtec, an issue was raised later in the development utilizing this scanner. The main issue encountered was with the dongle that is included with the LiDAR scanner and the associated cables. Ideally, if we were to start this project over, we would have used an Arduino to connect to the device as this would have allowed us much better control over the settings available. However, due to consistent use of the USB dongle, we could see an inconsistent connection to the LiDAR on many occasions as it would get very hot and stop responding to our commands during testing.

A solution to this was to code a check for the hardware using our health check of the lidar on our specified port, this allowed us the benefit of not needing to plug in and out the hardware of our USB port reducing wear on the unit.

2.2.2.2 Laptop Issues

Initially, the development of this project was carried out on a dual boot enabled laptop, this allowed for cross-development and the ability to perform testing on the LiDAR unit via the Linux and Windows OS respectively. As the development continued the project files continued to increase as did the dependency on good hardware. With the limited amount of ram and just a dual-core processor, it struggled to consistently meet the requirements needed for this level of development.

2.2.2 Covid-19 Lockdown and Working from Home

From a personal point of view, the scheduling involved with planning out time to study and work for this project proved to be quite difficult. The time spent studying with the resource available in Carlow IT during normal hours was extremely valuable, as once lockdown happened and due co-habiting with other people and then moving operations to a family home in the country made this much more difficult than it should have needed to be. This was mainly due to a less than an ideal internet connection via a mobile phone with terrible reception, and trying to help family members with less-mobile family members during the day.

Due to this emergence of such an unprecedented global pandemic and subsequent closure of the institute, and consequently, the requirement to work from home during a critical time in the projects developmental cycle, progress was noticeably diminished due to the factors outlined above. Taking this into account, the time spent on remaining features did not go as planned.

Having these compromises in mind and trying to keep in line with the schedule set in place made these unforeseen changes a challenge to overcome. However, the final product produced is in a state that is functional and will hopefully be deemed adequate amongst peers and colleagues.

3. Conformity with the Original Design

In many respects, the A.R.M LiDAR System has stayed true to the original design and concept developed for this application. Much of the key features and options are still yet to be implemented but the key components of logic and rendering of the data were successfully implemented with minor issues. The core functionality of the application was that of the implementation of representing 2D data in a 3D manner to give the user fast updates to direct them quickly and without hesitation or incident.

The user can view the data of the LiDAR as designed, however, the further implementation of the GUI is yet to be developed that could show the user a warning and or a direction to take depending on the distance to object's on either side of the device. Another feature that is yet to be implemented is the mapping of data that is shown in realtime to the user in the GUI. The developer believes that this feature to show the mapped data to the user in the window may prove difficult on windows and vs2019 and might prove more fruitful if developed on Linux with and IDE such as CMake.

3.1. Changes from the Original Design

There were few changes made to the specification and design during the development of the project. The biggest change within the application was the approach to the showing of the squares on the display, as of now, they show an object once it is in range but once it is out of the defined view nothing is displayed to the user. This could be changed with some updates to the OpenGL window to show a diminishing view on the top and bottom of the window to give a better understanding of what we are seeing.

As the other features have yet to be developed it would be easy to say that they have not changed and will remain the same as development continues as they are to be added to the feature list in the future potentially.

4. Learning Outcomes

4.1 Technical

During the development of this project, some new technologies and programming languages were unfamiliar to the developer which required a large level of upskilling. The technologies for this project were intentionally chosen as they were unfamiliar and would greatly increase the learning outcomes of this project. The technologies used to develop the A.R.M LiDAR System application are discussed below.

4.1.1 C++

C++ is a cross-platformed language that can be used to create sophisticated high-performance applications. The language gives developers a high level of control over system resources and memory. It is one of the most widely known languages and as such has a reputation of being one of the most difficult and hard to understand in the industry, however, if you can code in C++ it is said that most all other languages become easier to learn and develop with.

While having some prior knowledge of the language the developer did not know just how much they would need to learn to develop the application in its current state. The developer being somewhat familiar with the basics of web application development using C# and other associated web development technologies, found that C++ proved to be a very difficult technology to grasp and to gain a level of familiarity needed to complete the tasks set out before them. This required the developer to do just as much in-depth research on this single language alone just to get a level of understanding good enough to know what to use alongside the other library's outlined below as it did for most of the work carried out for a single module for a final year of college.

Thankfully there is a wealth of knowledge online in the form of documentation and video that allowed the developer to improve their abilities over the past months to an adequate level. This also includes the developers own personal "library" of sorts that helped with certain aspects of C++ coding and allow the developer to call on when needed.

4.1.2 SFML

SFML provides a simple interface to the various components of your PC, to ease the development of games and multimedia applications. It is composed of five modules: system, window, graphics, audio and network. An SFML application can compile on the most common operating systems which include Windows, Linux and macOS.

Learning new technologies is extremely valuable, but can be quite challenging at times. Given the opportunity to build an application with SFML is by far no exception to this statement. Having some knowledge of this library or API that is again written in C++, was somewhat helpful as it is an object-oriented SDL and is made up of multiple modules which are quite useful and can do many different operations. Using this API to code a window that

will run on multiple operating systems was not the initial idea, but having the ability to test the application on Linux proved quite useful, even more so when used with different IDEs like CMake on Linux.

Using this library certainly proved that you are required to have a better understanding of C++ to take full advantage of what is provided, as you can easily try and code a small app for a test in C++ with SFML very quickly, you will indeed learn very quickly that it may not be usable or even maintainable down the line.

4.1.3 Slamtec LiDAR SDK

The RPLIDAR standard SDK has been quite a difficult aspect of this project to learn, as it is one thing to implement a simple C++ project with some SFML, but the addition of an SDK like this makes it much more of a task to manage.

Whilst the development with this SDK and its documentation hasn't been the easiest on Windows, as these types of projects are largely found and documented on Linux with the likes of other SDK's built for applications like the A.R.M. LiDAR System, i.e. the hector slam SDK from ROS.org, that won't wrap your main so that it may run with other robot software frameworks. As such this allows ROS to be a distributed framework of processes (aka Nodes) that enables executables to be individually designed and loosely coupled at runtime. But unlike our RPLIDAR SDK, Slamtec provides a RoboStudio that is currently only available on Windows to do the evaluation, but this is mainly used for more in-depth robotics development.

As a side note, within the past four months and before the beginning of the development of this application, Slamtec updated their SDK for new features and support for newer IDE's and as such as ours (vs2019) is now supported properly.

You can utilize this SDK effectively within most Linux IDE's such as CMake, which allows you to perform a cross-compile that can be used with Makefile systems. This was a huge help for certain aspects of development as we could generate a CMake project from vs2019 with its cross-platform module, and with what currently had been developed we could use it natively on Linux for testing.

4.2 Personal

From my time working on this project, time management, as well as learning on my own initiative, have been the two biggest skills that have greatly improved I feel. To complete all of my project's current functions and requirements, I had to implement a strict time limit on certain parts of daily, weekly and weekend tasks, and on how much time spent on each task. Having to do most of these tasks or assignment for the final year of college has proved challenging given the time given. The best option I had available was to learn to optimise my time better and as such become efficient in doing so, even more so than ever with the circumstances of this unprecedented time we live in regarding the pandemic and while working from home to a certain degree of a professional manner.

This project was an extremely difficult task to handle from start to finish, not to mention the other assignments required to be done over the year, and again the lockdown that's happened over the past few months. But, organising my time has proven invaluable non the less and with the use of google calendar and todo lists from google, most tasks proved to be manageable given the right time allotted to them.

Personally adopting this behaviour structure has been a mentality taxing requirement and has been though to implement and do but, it helped me manage to keep on track and up to date for the workload for the A.R.M. LiDAR System application as well as the allotted study on the technologies required to gain the required level of familiarity with. I believe that I have performed to the best of my ability with the time allocated and the ambitiousness and complexity of the project and work given. One of the bigger personal disappointments of this project is that if I had taken a different approach to the project, or managed my time better at the start of the year, I imagen I could have accomplished much more and feature-rich and developed a much better version of the A.R.M. LiDAR System than we currently can see.

5. Project Review

This project has been one of the most technically and personally challenging projects I have undertaken to date, counting both during my studies and on work placement, and in terms of size and new technologies and concepts that I had never implemented before. Overall the A.R.M LiDAR System project was somewhat successful when compared to the main requirements, design and specification. Of the main functions, the main data transformation and associated logic have been implemented, albeit in a slightly different manner than originally designed and discussed in previous documentation.

5.1 What was Achieved

In the applications current form, we can see that the developer has completed some of the projects original intent of enabling a user a 3D view of the data generated from the LiDAR scanner in a manner that best suits its outlined purpose. This means that the user may use this application to a minor extent to see obstacles in front of them and move away accordingly given the set distance.

Additionally, the basics for the mobile unit logic in the form of the input source file that contains basic game loop logic can take commands from a keyboard but is not implemented as of yet, this is due to errors on compile as this function currently cannot do anything with the commands given.

The basic's of logic written to implement a zoom function based on the initial amount of nodes captured was a potential feature that has been worked on. This, however, was not in line with the tasks at hand and as such, the developer decided to put this feature to the side for the time being. This was mainly because it may not prove valuable to the user or the scope of the project with the allotted time given to development.

5.2 What was not Archived

Due to the Covid-19 lockdown and the subsequent closer of the Carlow IT campus, and having to move operations to work from home, the developer did not have access to adequate resources needed to implement the remaining functionality.

These features include the avoidance algorithm (A*), a GUI that would show warnings to the user depending on if they were getting close to an object or to suggest another route. This also includes the displaying of the generated map from the LiDAR's node data in the main GUI as this may need a considerable amount of work to complete.

Given the circumstances and the developers initial level of knowledge and with the need of upskilling, the level of features outlined for this project may have been out of scope from the start. Taking this into account, the developer would have changed a lot of aspects of this project given the knowledge gained from such a steep learning curve.

5.3 Future Features

Many features outlined in the documentation can be added to the A.R.M Lidar application in future iterations, given the adequate time and patience to develop them. As mentioned above, the application would have benefited from the implementation of a Linux based operating system and the tools available for a project like this. The resources available to a developer on the Linux platform is by far more extensive, this would have allowed for faster development as the developer could have utilized this knowledge to an even greater extent than what has been done currently.

This application could also benefit from some additional hardware and an Arduino kit to help with development on a more interesting and scalable level. The initial idea of this application was to have a mobile unit, a remote control car for instance, and to give it commands via its wireless controller or a wireless version of the RPLiDAR that can take commands from the user, or hooking up a wifi antenna to the Arduino and control everything from there. The input source logic would have been used to engage motors connected to the wheels and Arduino on the mobile unit, allowing for full automation.

From here we could have had the LiDAR working in conjunction with a small camera to avoid obstacles in the environment, giving us more freedom on what way we wanted to design the avoidance aspect of the project. Having this freedom to choose what way to avoid obstacles would have given the option to view the feed from the camera and allow the LiDAR to help direct the mobile unit safely around the environment, depending on the distance from the object to LiDAR.

However, as outlined above, the level of knowledge required by the developer would have needed to be extensive and the addition of extra devices would have decreased any chance of the project to ever be completed to some extent in the time given.

Acknowledgements

First, I would like to thank my project supervisor, Oisin Cawley, for his support and guidance throughout the project. This help has been extremely valuable in helping me think of solutions I otherwise would not have found. His encouragement and ability to help with what was realistic versus what was initially conceptualized was an exceptional boon on the development of this project as I had lofty goals for what I could do in the time given and what my skill level allowed.

I wish to thank all of my lectures, as without them and their consistent support I would never have made it through these four years of college. Going back to education as a mature student certainly has been an experience, one that I won't soon forget.

Finally, I want to thank the people I have met throughout the years of college as, without them and our mutual feelings of how difficult it has been, I don't think I could have made it through to the end.

Sincerely,

Calin Doran

Plagiarism Declaration



Work submitted for assessment which does not include this declaration will not be assessed.

DECLARATION

*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) CALIN DORAN
Student Number(s): C00220175
Signature(s): Calin Doran
Date: 3/4/2020

Please note:

- a) * Individual declaration is required by each student for joint projects.
- b) Where projects are submitted electronically, students are required to type their name under signature.
- c) The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook.