Tampere University

Riku Törmä

LIDAR USAGE EVALUATION IN IOT DEVICE

Master of Science Thesis
Faculty of Management and Business
Examiner: Professor Tarmo Lipping
Examiner: University Lecturer Jari Turunen
June 2023

ABSTRACT

Riku Törmä: LiDAR usage evaluation in IoT device
Master of Science Thesis
Tampere University
Management and Information Technology
June 2023

The first objective of this thesis is to find out if it is possible to use LiDAR (light detection and ranging) with the Epec 6200 Remote Access Unit variant, Epec 6200-268. The second objective of this thesis is to find out if it is possible to execute application that utilizes LiDAR with the Epec 6200-268 and to see if the performance level of the Epec 6200-268 is sufficient.

To achieve the objectives, a system was built that connects the Epec 6200-268 unit and the LiDAR device to the same Ethernet network where the devices communicate via UDP (User Datagram Protocol). Epec 6200-268 unit has a tailored Linux operating system that runs ROS (Robot Operating System) in a Docker container. The used LiDAR has a ready-made ROS compatible LiDAR driver that was used to verify that LiDAR works with Epec 6200-268. The actual LiDAR utilizing application was developed as a separate ROS application. The application is based on DATMO (Detection and Tracking of Moving Objects) algorithm that can be used to detect and track moving objects. The application has parameters that can be adjusted to change the amount of load the application causes to the Epec 6200-268.

After the system was built, the system was used to run Epec 6200-268 performance tests. The Epec 6200-268 performs well when it is running only the LiDAR driver. In this case the CPU (central processing unit) average load does not exceed 10% of CPU maximum capacity. When LiDAR driver and DATMO algorithm are executed simultaneously with parameter set that limits the needed performance, the CPU average load does not exceed 24% of the CPU maximum capacity.

When the same test is run without limiting parameter set, the CPU average load does not exceed 35% of the CPU maximum capacity. This is a good result since the allowed maximum for CPU average load is set to 70%. However, the CPU single core load occasionally reaches 100% in this test, which is not acceptable. In addition, the DATMO algorithm execution time surpasses 200 ms, which is too much when the acceptance limit is 40 ms or less.

The results of the performance measurements indicate that Epec 6200-268 can be used with LiDAR and that it is possible to run simple LiDAR applications with Epec 6200-268. Based on results, it would be beneficial to study especially how to load the CPU more evenly with the DATMO algorithm, since now the DATMO algorithm is loading single core most of the time.

Keywords: Epec 6200, LiDAR, DATMO, ROS, Docker

TIIVISTELMÄ

Riku Törmä: LiDARin käyttämisen evaluointi IoT-laitteella
Diplomityö
Tampereen yliopisto
Johtamisen ja tietotekniikan DI-tutkinto-ohjelma
Kesäkuu 2023

Tämän työn ensimmäisenä tavoitteena on selvittää, pystyykö Epec 6200 Remote Access Unit IoT-laitteen variantilla Epec 6200-268 käyttämään LiDARia (light detection and ranging). Työn toisena tavoitteena on selvittää, pystyykö Epec 6200-268 laitteella ajamaan LiDAR-sovelluksia ja onko laite tarpeeksi tehokas tähän käyttötarkoitukseen.

Tavoitteiden saavuttamista varten rakennettiin testausjärjestelmä, jossa Epec 6200-268 ja LiDAR kommunikoivat Ethernet-yhteyden välityksellä käyttäen UDP-protokollaa (User Datagram Protocol). Epec 6200-268 käyttöjärjestelmänä toimii räätälöity Linux, jonka päällä pyöritetään ROS-ympäristöä (Robot Operating System) Docker-kontissa. Työssä käytetylle LiDARille löytyy valmiina ROS-pohjainen ajuri, jota käytettiin todentamaan LiDARin toimivuus Epec 6200-268 laitteen kanssa. Varsinainen LiDARia hyödyntävä sovellus toteutettiin osana työtä erillisenä ROS-sovelluksena. Sovellus pohjautuu DATMO-algoritmiin (Detection and Tracking of Moving Objects), jolla voidaan tunnistaa ja seurata liikkuvia kohteita. Sovellusta on mahdollista parametroida siten, että sovelluksen aiheuttama kuormitus Epec 6200-268 laitteelle vaihtuu parametroinnin mukaan.

Valmiilla testausjärjestelmällä ajettiin Epec 6200-268 laitteen suorituskykytestit. Epec 6200-268 suoriutuu hyvin, kun sitä kuormitetaan ainoastaan ajamalla LiDARin ajuria. Tällöin prosessorin kuormituksen keskiarvo ei ylitä 10 prosenttia prosessorin maksimikapasiteetista missään vaiheessa. Kun LiDAR-ajurin lisäksi ajetaan myös DATMO-algoritmia suorittavaa sovellusta kuormitusta rajaavalla parametroinnilla, niin prosessorin kuormituksen keskiarvo ei ylitä 24 prosenttia prosessorin maksimikapasiteetista.

Kun parametroinnilla ei enää rajata kuormitusta niin prosessorin kuormituksen keskiarvo ei ylitä 35 prosenttia prosessorin maksimikapasiteetista. Tämä on hyvä tulos, sillä prosessorin kuormituksen keskiarvon sallituksi maksimiarvoksi on asetettu 70 prosenttia. Sen sijaan tässä testissä yksittäisen ytimen kuormitus kipuaa hetkittäin 100 prosenttiin, mikä ei ole suotavaa. Lisäksi DATMO-algoritmin suoritusaika ylittää 200 ms, kun hyväksyttävänä rajana on 40 ms tai alle.

Suorituskykytesteistä voidaan päätellä, että Epec 6200-268 laitteella on mahdollista käyttää LiDARia ja suorittaa yksinkertaisia LiDAR-sovelluksia. Tuloksien perusteella voidaan suositella jatkotutkimusta liittyen erityisesti sovelluksen parempaan säikeistämiseen, sillä nyt DATMO-algoritmia ajava sovellus kuormittaa pääasiassa yksittäistä prosessorin ydintä.

Avainsanat: Epec 6200, LiDAR, DATMO, ROS, Docker

PREFACE

At first, I would like to thank my fiancé Sini-Tuulia Mäkelä for all the support she has given me during my university studies and my thesis writing process. I would like to also thank the examiners of this thesis, university lecturer Jari Turunen and professor Tarmo Lipping for guiding me through this process and giving me valuable feedback. From Epec Oy I would like to thank my mentor Mr. Kristian Vaajala who has given me valuable feedback throughout the thesis writing, I would like to also thank Mrs. Lisa Venäläinen, Mr. Eero Pärnänen and Mr. Petri Aaltonen for their counselling and support. Finally, I would like to thank Mr. Marko Aalto for assisting me with the tests that were conducted for this thesis.

Seinäjoki, 19 June 2023

Riku Törmä

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	application programming interface
BSD	Berkeley Software Distribution
CAN	Controller Area Network
CPU	central processing unit
DATMO	Detection and Tracking of Moving Objects
GPS	Global Positioning System
GPU	graphics processing unit
HMI	human-machine interface
IoT	internet of things
laser	light amplification by stimulated emission of radiation
LiDAR	light detection and ranging
MPE	Media Processing Engine
NDA	non-disclosure agreement
NPU	neural processing unit
PCL	Point Cloud Library
RAM	random-access memory
ROS	Robot Operating System
SIMD	single instruction, multiple data
SLAM	simultaneous localization and mapping
ToF	time of flight
UDP	User Datagram Protocol
VFPv3	Vector Floating-Point v3
VPU	video processing unit

1. INTRODUCTION

Epec Oy is a Finnish company that manufactures control systems for non-road mobile machinery. The control systems of non-road mobile machinery are constantly evolving thus increasing the requirements that a modern control system needs to fulfil. One example of this is the automation level of modern non-road mobile machinery that is increasing and thus requiring more performance from the control system itself. In addition, familiar technologies from the automotive industry are transferring into non-road mobile machinery as well. These technologies, being for example, *electrification* and *assistance and autonomous systems*, which add even more requirements to control system manufacturers.

One example of electrification in non-road mobile machinery is the *Ponsse EV1* which is powered by *Epec Flow* electrification system [13]. According to the Ponsse press release, Ponsse EV1 is a forest machine concept that has an electric powertrain [35]. The press release also states that for now, the Ponsse EV1 is still using a combustion engine as a range extender, but the powertrain is fully driven with battery energy thus enabling the possibility for a fully electric forest machine in the future.

In addition to electrification, the assistance and autonomous systems enable new possibilities in non-road mobile machinery. Sensors like *LiDAR* (light detection and ranging) are acting as a key role to achieve such features. LiDAR can be used to make range measurements of the surroundings of a non-road mobile machinery thus producing a point cloud of range measurements that can be utilized in assistance and autonomous systems. Since Epec is providing control systems for non-road mobile machinery, it is also an interest of Epec to study how the LiDAR can be utilized with both current and future products of Epec.

1.1 Objectives of the thesis

This thesis focuses on an already released Epec product called *Epec 6200-268*, which is part of the *Epec 6000 Series* product family. There are two main objectives that are studied in this thesis.

The first objective is to simply find out if it is possible to use LiDAR with Epec 6200-268. To achieve the first objective of this thesis, a system is built that can be utilized to

test if it is possible to use LiDAR with Epec 6200-268. This system is built around two main hardware components: Epec 6200-268 and LiDAR, which is manufactured by a third-party company. Before the system is implemented, the LiDAR technology is first studied. In addition to hardware, software is also needed. Epec 6200-268 uses a tailored *Linux* operating system. On top of Linux, a system is built that can run *ROS* (Robot Operating System), which is commonly used in robotics related application development. With ROS it is possible to utilize a ready-made LiDAR driver that is provided by the LiDAR manufacturer. Epec 6200-268 does not have native support for ROS, so for this *Docker* is used to run a *Docker Image* that can run ROS.

The second objective is to find out if Epec 6200-268 can run applications that utilize LiDAR, and furthermore to see if the performance level of Epec 6200-268 is sufficient in such applications. To achieve the second objective a LiDAR application is developed. The application is based on the LiDAR algorithm called *DATMO* (Detection and Tracking of Moving Objects). The DATMO algorithm is first studied before implementing the actual application. The application is developed on top of ROS like the LiDAR driver and the application has parameters that can be changed to vary the load of the Epec 6200-268 CPU (central processing unit) to see how the unit performs with different load conditions.

1.2 Structure of the thesis

The thesis is structured as follows. Chapter 2 covers the technology overview that is necessary to implement the test setup that is used in this thesis to evaluate the performance of Epec 6200-268. This technology overview consists of three sections. The first section covers LiDAR technology to achieve a basic understanding of how the LiDAR works. The second section discusses ROS, which is a key piece in the system that is used to perform the performance tests with Epec 6200-268. The third section introduces Epec 6200-268, including the processor architecture to give a clear picture of the unit and its capabilities.

In chapter 3 a theoretical overview of the DATMO algorithm is covered. DATMO is an algorithm that can be used to detect objects from LiDAR point cloud and then track the movement of these objects. DATMO acts a crucial role in the performance tests of Epec 6200-268.

Chapter 4 introduces the actual system that is implemented around Epec 6200-268. The different components of the system are covered including LiDAR that is used to perform all the tests and software components that are used in the system. Also, the system monitoring is covered in the chapter 4.

Chapter 5 covers the performance tests that are run with the implemented system. The test plan including all the test cases is represented in this chapter. Also, the results of every test conducted in this thesis are represented in this chapter.

The final chapter is chapter 6, which contains the conclusions of this thesis. Within the conclusions, the summary of results is presented and the recommendations for further study are given.

2. TECHNOLOGY OVERVIEW

This chapter digs into the technology that is needed in this thesis. Section 2.1 contains information of LiDAR in general, including an overview of LiDAR history, LiDAR measurement principle, different LiDAR types, point clouds and LiDAR applications. Section 2.2 introduces ROS, including a general presentation of ROS and information of the most important ROS features related to this thesis. Section 2.3 contains information of the Epec 6200-268 unit and its processor architecture.

2.1 LiDAR technology

LiDAR is an active sensor with its own artificial light source. Like the LiDAR acronym suggests, the LiDAR is used to detect light and calculate range between the LiDAR and the measured surface by sensing the backscattered light of the LiDAR's own illumination. LiDAR can sense different properties from the backscattered light like brightness, angular position, and timing. With this information, LiDAR is capable of not only measuring the distance of the measured surface, but also other things like, for example, earth surface vegetation features in aerial canopy measurement applications. [6]

2.1.1 LiDAR history

In 1961, the first LiDAR was built by Hughes Aircraft Company [15]. This happened shortly after one of the most important LiDAR components, a laser (light amplification by stimulated emission of radiation), was invented in 1960 [5]. The first applications of LiDAR focused mostly on terrain mapping [32].

LiDAR technology became more known by the public in the 1971 when the LiDAR was used first time in space in NASA's Apollo 15 mission that landed to the moon. LiDAR was used to take multiple measurements of lunar surface height to map the moon surface. [1]

The LiDAR development was somewhat stagnated until the 1980s when some of the important technologies matured [6][32]. One of these technologies was the GPS (Global Positioning System) that started to be available in the 1980s [10] and enabled accurate positioning that was needed in aerial LiDAR applications [6].

Nowadays, LiDAR technology has many new applications. LiDARs are used e.g., in household items like robotic vacuums [3] and automotive applications to enable the future of autonomous driving [32].

2.1.2 LiDAR measurement principle

The measurement principle of LiDAR is known as time of flight (ToF) [17]. The time of flight, *t* is measured between the emission and the detection. Since the speed of light *c* is known, the distance *d* between the sensor and target object can be calculated with the following formula [18]:

$$d = \frac{1}{2}ct. \tag{2.1}$$

LiDAR uses a laser as its light source that it emits towards the measured surface. The laser produces coherent light that can travel large distances. The laser beam itself is narrow. By utilizing laser in LiDAR, it is possible to get accurate measurements, since the emitted light does not diverge much. This makes it possible to produce detailed imagery. [6]

LiDAR's laser radiates waves with near-infrared, visible or near-ultraviolet wavelength. The most suitable wavelength depends on the measured target. To achieve eye safe operation, the emitted laser beam is usually shaped with optics. This shaping includes increasing the diameter and manipulating the divergence of the laser beam. [7]

Returning reflection is detected with a photodetector like a photodiode that converts the light into current signal [17]. Before detection, the light flux is collected and filtered with a combination of optics [7].

There are two different techniques to illuminate the measured scene: *pulsed-light* and *continuous-wave*. This thesis focuses on the pulsed-light LiDARs where the LiDAR illuminates the scene with a pulsed light and simply calculates the round-trip time between emission and detection. The pulsed-light LiDAR is suitable for outdoor applications, which is an advantage with non-road mobile machinery. [17]

The LiDAR range measurement principle is shown in Figure 1 below. The figure shows the light emitter that emits light towards an object and light detector that detects the backscattered light.

In addition to light emitter and light detector, LiDAR also has some sort of beam steering components that enable the LiDAR measurement on a wider area instead of just one point in space. There are at least two types of beam steering technologies that are used in LiDARs: mechanical LiDAR and solid-state LiDAR [20]. These are further discussed in the following subsection 2.1.3.

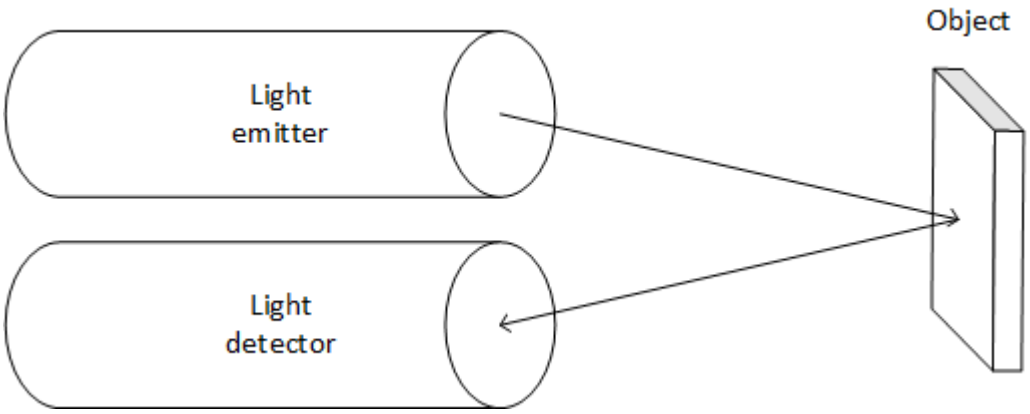

Figure 1. *LiDAR range measurement principle. The arrows represent the light beam that travels from the light emitter towards the object and then reflects to the light detector.*

2.1.3 LiDAR types

Mechanical LiDAR usually contains some sort of rotating mechanism that is used to steer the laser beam to different angles. With this approach it is possible to make measurements in a wide area and the field of view can be, for example, 360 degrees. [20]

One of the downsides of mechanical LiDAR is that the laser pulses are not emitted to all directions at once so the whole field of view is not updated at the same time. Also, moving parts tend to be more prone to break in harsh conditions like under vibration. This must be kept in mind, especially if LiDAR is used in more physically demanding applications like in non-road mobile machines.

Solid-state LiDAR does not have any moving components, thus making it more suitable for harsh conditions, since it is more robust when it comes to vibrations. One example of solid-state LiDAR is a flash LiDAR that is quite like a digital camera in operation. The main difference between the flash LiDAR and digital camera is that the flash LiDAR uses artificial light source to emit the light to the surroundings that is then detected by the flash LiDAR. The flash LiDAR illuminates a large area with a single laser pulse that is diverged with optics and thus illuminates the whole field of view at once, like shown in Figure 2 below. The backscattered light of this single pulse is then detected with an array of photodetectors. The size of the array defines the image resolution of the LiDAR. With this approach, a flash LiDAR creates a three-dimensional image of the measured surface with a single laser pulse. One disadvantage of the flash LiDAR compared to mechanical LiDAR is that the flash LiDAR requires a powerful laser to see far since the laser beam is diverged into a wider area compared to the mechanical LiDAR's laser beam, which is more directional. [20]

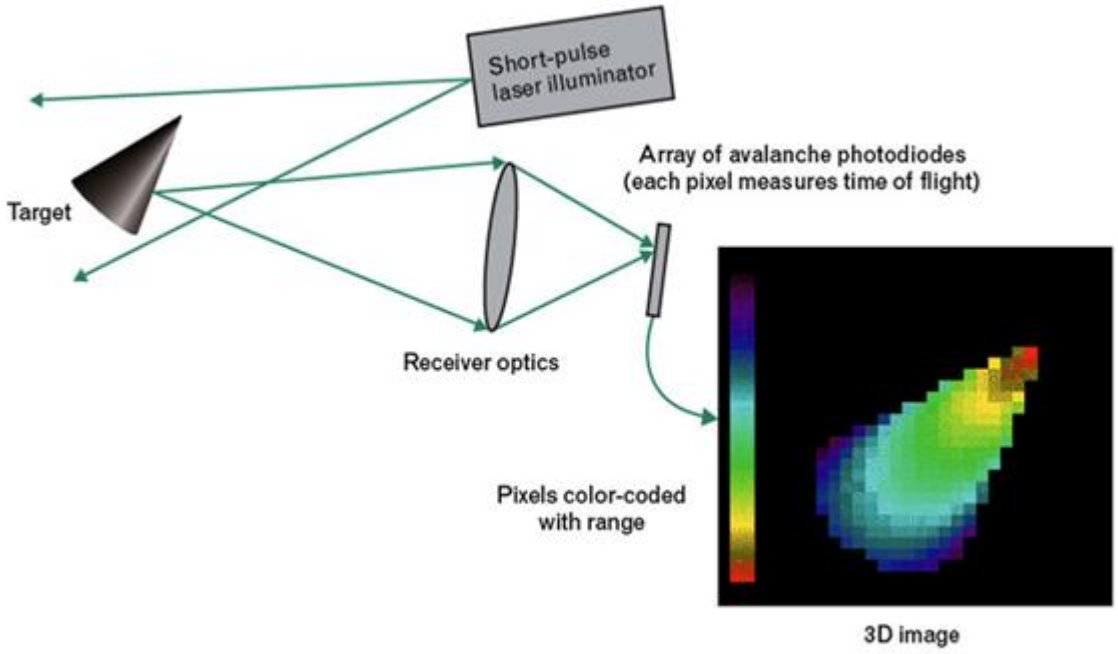

Figure 2. *Flash LiDAR basic principle [17]. Reproduced with permission from Springer Nature.*

2.1.4 Point cloud data

LiDAR measurements form a three-dimensional image of the LiDAR's surroundings, and this three-dimensional image is commonly known as a point cloud [17]. Depending on the resolution of the LiDAR the point cloud can consist of thousands of points with the minimum information of x-axis, y-axis, and z-axis position for every point. The high resolution of point cloud data with a high sampling frequency can make a lot of data that has to be processed by the processing device that is running LiDAR algorithms to refine point cloud data to reasonable information. In other words, LiDAR data handling requires a lot more processing power compared to control signal processing that is usually handled with programmable control unit.

2.1.5 LiDAR applications

Like mentioned in subsection 2.1.1, the first applications of LiDARs focused mostly on terrain mapping. In today's world, the LiDAR usage is widely spread and can be found in a lot of different applications from automotive industry to household appliances.

A robotic vacuum cleaner is a good example of a home appliance that utilizes LiDAR [3] to make the appliance a lot "smarter". Nowadays, some models of robotic vacuum cleaners come with a LiDAR that is used to measure the distance between the robot and its surroundings to determine the position of the robotic vacuum cleaner compared to its surroundings. When the position is known, it is a lot easier to keep track of already

cleaned areas and those that still need some vacuum-cleaning. In Figure 3 below is an example of a Roborock S5 robotic vacuum cleaner that has a laser distance sensor mounted on top of the unit [3].

Figure 3. *Image of a Roborock S5 which has laser distance sensor mounted on top of the unit.*

Some examples of automotive LiDAR applications include assistance systems like lane-keep assistance and adaptive cruise control, which can increase the safety of driving and make the driving easier for the driver [20]. According to Volvo, the LiDAR is also an important component in the development of autonomous driving [37].

2.2 Robot Operating System (ROS)

ROS is a meta-operating system that is designed for robot applications according to ROS documentation [29]. The ROS documentation states that the ROS runtime is a peer-to-peer network of processes that can communicate with each other in various ways, including running the processes in one or multiple machines, thus enabling distributed communication. The basic idea of a peer-to-peer system is shown in Figure 4 below.

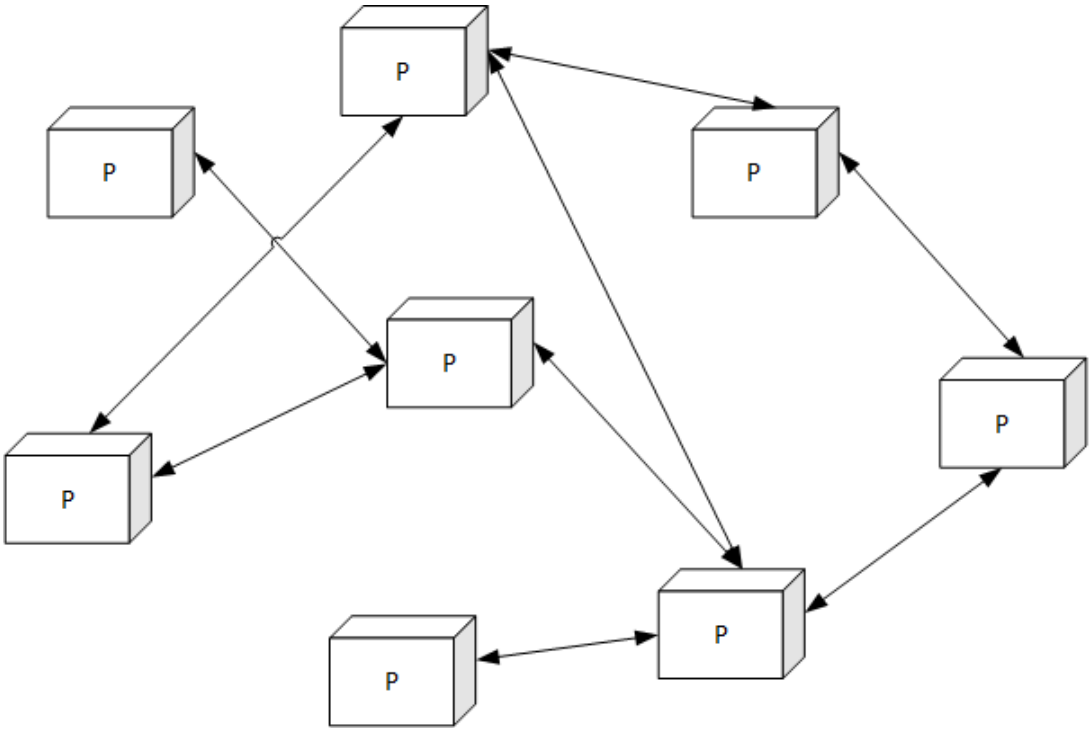

Figure 4. *Illustration of a peer-to-peer system adapted from [36]. Reproduced with permission from Springer Nature.*

This section focuses on the ROS features that are used in this thesis. The covered version of ROS is ROS 1. There is also a newer version of ROS available called ROS 2, but to maintain compatibility with the LiDAR ROS driver, the older version is used.

According to ROS documentation, ROS processes are called *nodes* and each node has its own task [28]. For example, one node can act as a LiDAR driver, and another node can run LiDAR related algorithms. The ROS documentation also states that the communication between nodes is handled with messages that can contain many types of data. According to ROS documentation it is possible to transfer messages via *topics* that work with a publish and subscribe principle: a node can publish a topic and another node can then subscribe to that topic to receive the published data. In this thesis two ROS nodes are used, and these are discussed in more detail in the upcoming sections 4.3 and 4.4.

ROS also supports a parameter server [27]. The parameter server is used by nodes to save and retrieve parameters. The parameter server is best used for static data like configuration parameters [27]. In this thesis ROS parameters are used to configure LiDAR algorithm specific parameters that can be changed to adjust the properties of the algorithm execution to match the environment features and performance requirements.

To analyse and visualize the ROS related data, a three-dimensional visualization tool for ROS is available. This tool is called *rviz* and it is specifically designed for ROS [31]. With

rviz it is possible to visualize, for example, point clouds that are published as topics by ROS nodes as seen in Figure 5 below. This is particularly important to be able to monitor the operation and performance of the system that is developed in this thesis.

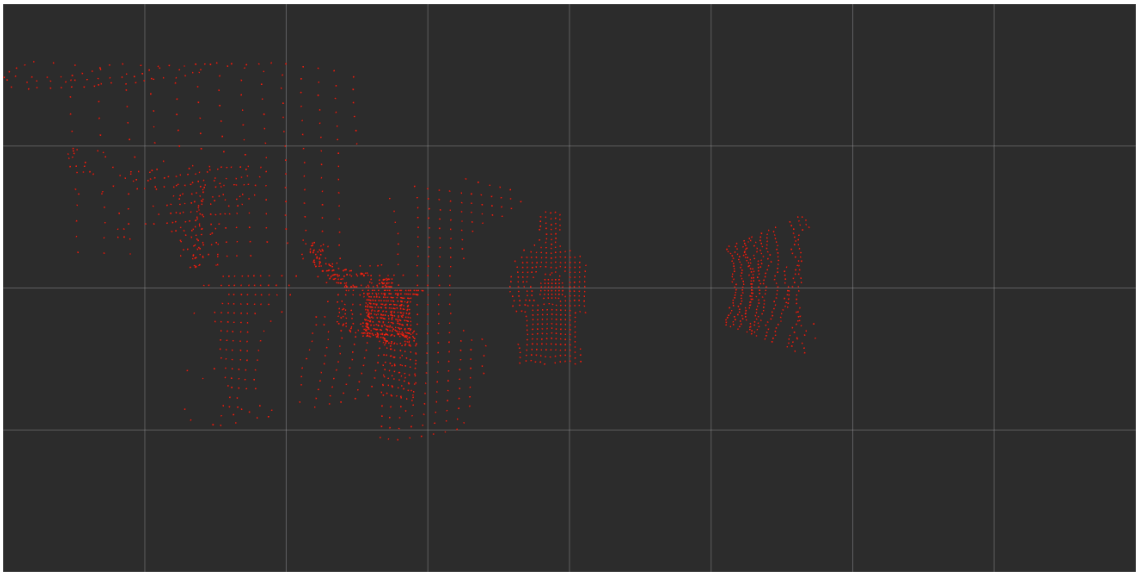

Figure 5. *Screenshot of a point cloud projection shown in rviz.*

2.3 Epec 6200-268

Epec 6200 Remote Access Unit shown in Figure 6 below is a unit that is part of the *Epec 6000 Series* product family that consists of displays, central units, and remote access units. Epec 6000 Series displays come with various touch screen sizes from 5 inch to 12.1 inch [11] and are used as a user interface in many Epec customer applications. The Epec 6000 Series central unit is designed to be used as a master unit in systems with centralized intelligence where most of the application code is run in the master unit thus offering multiple CAN (Controller Area Network) bus interfaces and a lot of computational capacity [14]. The Epec 6000 Series Remote Access Unit is an IoT (internet of things) unit that has WLAN and mobile interfaces to access the internet wirelessly [12].

The Epec 6000 Series is running a tailored *Linux* operating system. On top of the Linux, the Epec 6000 Series also has support for tools like *Docker*, which enables support for containers [9].

Although there are many different variants available in the Epec 6000 series, this thesis focuses on one of the most high-performing models when it comes to computational capacity. The unit used is the Epec 6200 Remote Access Unit variant called *Epec 6200-268*, which has a 32-bit quad core CPU that is running on a clock speed of 792 MHz, has 32 gigabyte of flash memory and a total of 1024 megabyte RAM (random-access memory).

Figure 6. *Image of Epec 6200 Remote Access Unit [12].*

The Epec 6200-268 variant supports six CAN buses, one RS-232 (Recommended Standard 232) interface, one USB (Universal Serial Bus), two Ethernet ports and one SIM slot. The Epec 6200-268 variant also has connectors for multiple different antennas including WLAN and LTE. In this thesis the first one of the two Ethernet ports is used to connect Epec 6200-268 to the same Ethernet network with the LiDAR and Ubuntu laptop.

2.3.1 i.MX 6Quad processor architecture

The processor of the Epec 6200-268 is an *i.MX 6Quad* 32-bit quad core processor. According to the processor data sheet, the processor has 64-bit memory interface that has support for DDR3 [23]. The data sheet also states that each core of the i.MX 6Quad is based on Arm Cortex-A9 MPCore that has 32 Kbyte L1 instruction cache and 32 Kbyte L1 data cache on each core, as can be seen from Figure 7 below. According to the data sheet each core also includes a Cortex-A9 NEON MPE (Media Processing Engine) co-processor.

One way to utilize the Cortex-A9 NEON MPE is to use *Arm Compute Library* which is a library that contains functions for machine learning and computer vision and is available

in ARM-software GitHub page [16]. According to ARM-software GitHub page, the library is optimized, for example, for Arm CPUs that are using SIMD technologies. Based on Arm documentation, the Cortex-A9 NEON MPE co-processor provides support for ARM v7 Advanced SIMD (single-instruction, multiple data) instruction set [2].

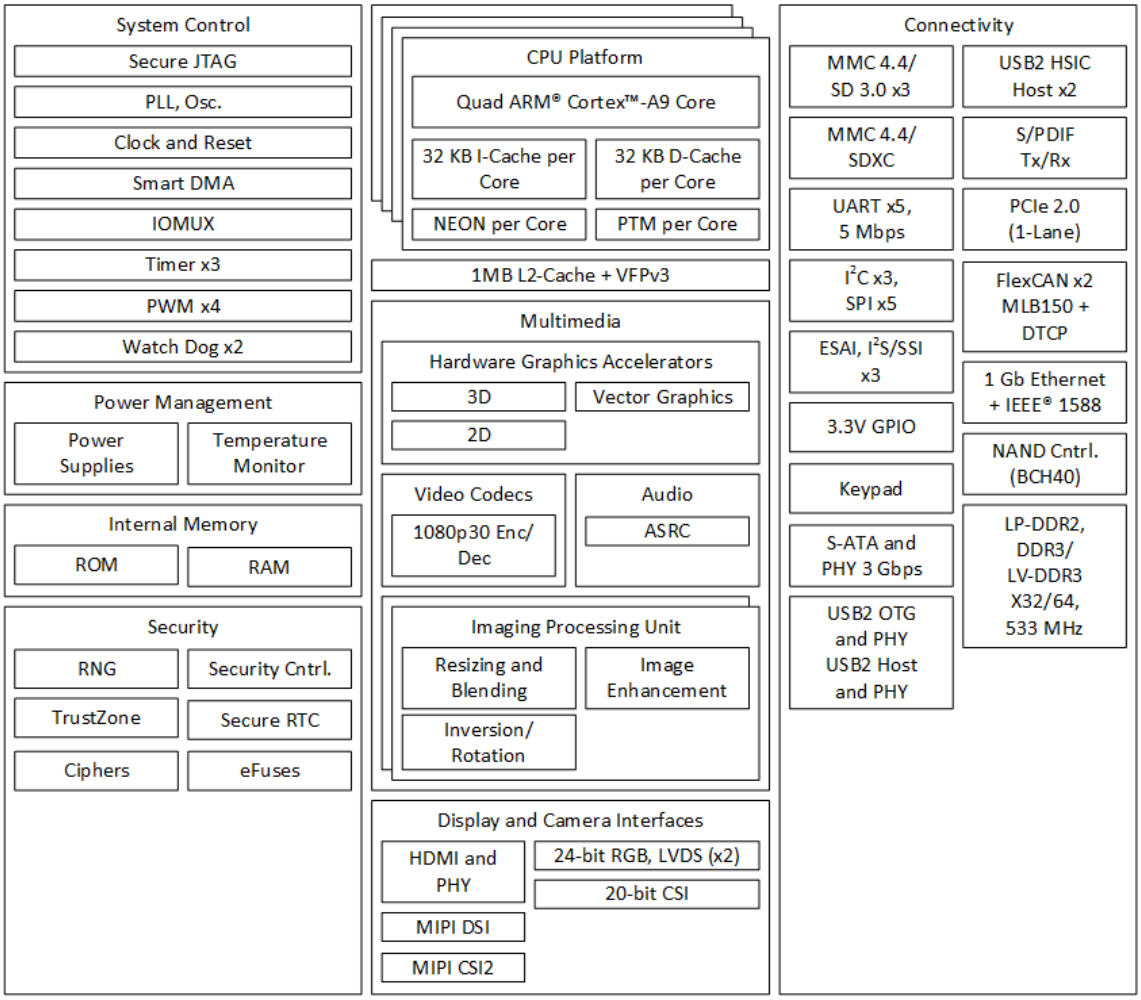

Figure 7. *Block diagram of i.MX 6Quad Multimedia Applications Processor adapted from [24].*

Although the processor has some support for machine learning and computer vision the processor data sheet states that the processor itself is designed for automotive and infotainment applications [23]. Therefore, the processor is not specifically designed for assistance and autonomous systems applications. The processor is also quite dated since i.MX6 was originally released in 2012 [4]. With this information it is quite safe to say that the processor might not be best suited for modern control system that is dealing with assistance and autonomous systems like LiDAR applications. For example, the newer model from NXP called *i.MX 8M Plus* has dedicated NPU (neural processing unit) and support for machine learning and machine vision [25]. This makes it a better candidate to be utilized in assistance and autonomous systems related tasks.

According to the processor data sheet, the specific model of the processor used in Epec 6200-268 incorporates two additional hardware accelerators: VPU (video processing unit) and GPU (graphics processing unit) [23]. The processor data sheet further adds that by using the hardware accelerators it is possible to have high performance level with low power consumption. There has been previous research in Epec Oy and it seems that there is no ready-made library support available for these hardware accelerators when it comes to computer vision applications, thus this would require quite a lot of work to fully harness the power of the VPU and GPU in assistance and autonomous systems related applications. The GPU itself is more suitable for graphics related computation since it has hardware acceleration support for two-dimensional and three-dimensional graphics algorithms [23].

3. DETECTION AND TRACKING OF MOVING OBJECTS (DATMO)

This chapter focuses on DATMO. The first section 3.1 discusses DATMO in general and the following section 3.2 goes deep into the object-based DATMO that is used in the actual performance tests of the Epec 6200-268 unit.

3.1 DATMO in general

Detection and tracking of moving objects (DATMO), which is also known as detection and tracking of moving obstacles is an approach that aims to track dynamic objects, for example, in the surroundings of a robot. This is necessary to achieve capability of safe navigation in an unknown environment. For this purpose, it is important to be able to track any dynamic object in the scene and predict its future position. The tracking must be a continuing effort with a high level of reliability. [22]

To be able to gather the needed information of the environment to implement DATMO, usually range sensors or cameras are used. The range sensor can be, for example, LiDAR or sonar. Usually, LiDAR is used as a main sensor since it is very accurate and has a high resolution, which is beneficial when detecting objects. [22]

DATMO can be an off-line or on-line process. In the off-line process the surroundings are known beforehand and the tracks and velocities of all the dynamic objects are known. The off-line process is good for e.g., industrial applications where the environment is constant. On the other hand, the on-line process is an approach where the environment is changing all the time and thus the dynamic objects and their tracks or velocities are not known in advance. In the on-line process the DATMO usually consists of two steps. The first step is *data association phase* and the second one is *multi-object tracking algorithm*. [22]

According to Llamazares et al. [22] DATMO methods can be further divided to *object-based DATMO* and g*rid-based DATMO*. This thesis focuses on object-based DATMO and thus the grid-based DATMO is not further discussed. Llamazares et al. also state that the object-based DATMO can be model-based or model-free [22].

The model-based approach knows beforehand the type of objects that it needs to detect. The detection is based on a parametric model which describes the shape of the object that needs to be detected. The model-based DATMO can be used, for example, to detect people or cars. Since the model-based DATMO is aware of the type of objects it needs

to detect, it can also detect currently static objects that might move in the future. The weakness of the model-based DATMO is that it can detect only the type of objects that match its parametric model. [22]

The model-free approach does not need any parametric model since it is not bound to any specific object shape. Instead, the model-free DATMO aims to detect all the dynamic objects in the scene. The weakness of this approach is that it can only detect currently moving objects and cannot predict if some of the static objects might move in the future, since it does not know the type or class of each object in the scene. [22]

3.2 Object-based DATMO

In object-based DATMO the actual algorithm can be divided into a few different steps that are discussed in this section. Based on the object-based DATMO description of Llamazares et al., the algorithm can be roughly divided into three steps: *data segmentation*, *data association* and *target tracking* [22].

According to Llamazares et al., in the **data segmentation** step, the data is segmented and the simplest way to do this is called clustering, where the LiDAR point cloud is grouped into clusters [22]. Kumar Rath et al. state that one way to perform clustering is called Euclidean clustering in which LiDAR point cloud points are clustered based on the Euclidean distance between them [21].

According to Kumar Rath et al., when performing Euclidean clustering, the points that belong to same cluster can be decided e.g., by parameters like the maximum allowed distance between two points and the minimum number of points that the cluster must have [21]. In a three-dimensional space the Euclidean distance D_E between two points p and q is calculated with the following equation:

$$D_E = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}. \tag{3.1}$$

In the **data association** stage, the detected objects are associated with the objects that were detected from the previous LiDAR frame. According to Llamazares et al., this is one of the more challenging tasks of DATMO since the data association step needs to define if the found object corresponds to an already existing object [22]. Llamazares et al. also state that data association step needs to determine when the objects are created as new, updated as an already existing object, or removed entirely.

To associate the current LiDAR frame objects to the previous LiDAR frame object a nearest neighbor approach can be used [21]. Darrell et al. defines the nearest neighbor search problem in the following way.

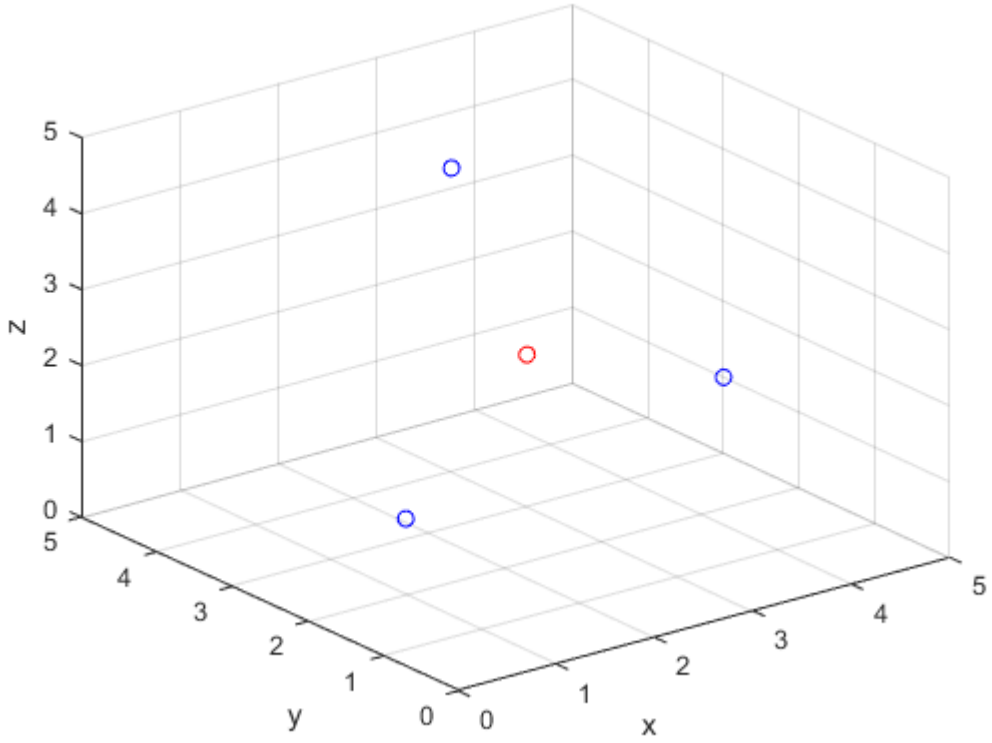

Figure 8. *Red circle is illustrating the centroid of three blue points.*

"Given a set P of points in a d-dimensional space $\mathbf{R}^d$, construct a data structure which given any query point q finds the point in P with the smallest distance to q." [8]

To use the nearest neighbor approach, it is beneficial to simplify the object data since each object consists of multiple point cloud points. One way to simplify the object data is to calculate the centroids of the objects. This way there is only one point (centroid) presenting each object. Though to be able to use the centroid information accurately it is required that the spatial position of an object does not differ too much between two consecutive LiDAR frames. [21]

The centroid of an object is calculated by using all the points of the object. The equation 3.2 is a simple way to find the centroid *C* of a three-dimensional object that consists of *n* points, and this is the way it is calculated in the *PCL* (Point Cloud Library), which is utilized in the actual DATMO algorithm implementation that is discussed in section 4.4 [33]. The equation calculates the arithmetic mean of each axis x, y and z and uses the mean values as the centroid of the given object.

$$C = \frac{1}{n}(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i, \sum_{i=1}^{n} z_i). \tag{3.2}$$

Figure 8 above illustrates an example of a centroid that is calculated from the object that consists of three blue points. The centroid is displayed as a red point.

The third and final step of the DATMO algorithm is **target tracking**. Usually, the tracking utilizes some variation of Kalman filter (KF) to achieve reliable object detection and tracking even in occluded environments [22]. Kalman filter is not further discussed because it is out of the scope of this thesis since the actual DATMO algorithm that is implemented for this thesis does not utilize Kalman filter in its tracking algorithm. The tracking part of the implemented DATMO algorithm is based on data association and works as expected, since the tracking task that is applied to the DATMO algorithm is quite simple and does not need to handle occlusion.

4. SYSTEM IMPLEMENTATION

This chapter contains information of the system that was implemented to run the needed performance tests for the Epec 6200-268. Section 4.1 describes the system architecture of the implemented system. Section 4.2 introduces the LiDAR that was used. Section 4.3 discusses LiDAR ROS driver which is delivered by a third-party company. Section 4.4 describes ROS node that implements the DATMO algorithm. Section 4.5 covers the ROS Docker Image that is run in the Epec 6200-268 and section 4.6 explains how the system can be monitored.

4.1 System architecture

The main goal of this thesis is to find out if the Epec 6200-268 is capable of handling LiDAR applications and for that purpose a system is built that can be used to test Epec 6200-268 performance with LiDAR applications. The system consists of hardware and software components.

The hardware setup to run needed tests is built around the Epec 6200-268 and a LiDAR that is manufactured by a third-party company. The Epec 6200-268 was discussed earlier in section 2.3. The following section 4.2 represents the LiDAR. In addition to the Epec 6200-268 and LiDAR a laptop with *Ubuntu* operating system is used. The main purpose of the laptop is to visualize point cloud data with the ROS visualization tool *rviz* that was discussed in section 2.2. Communication between the hardware equipment is handled with Ethernet communication. The illustration of the Ethernet network is shown in Figure 9 below.

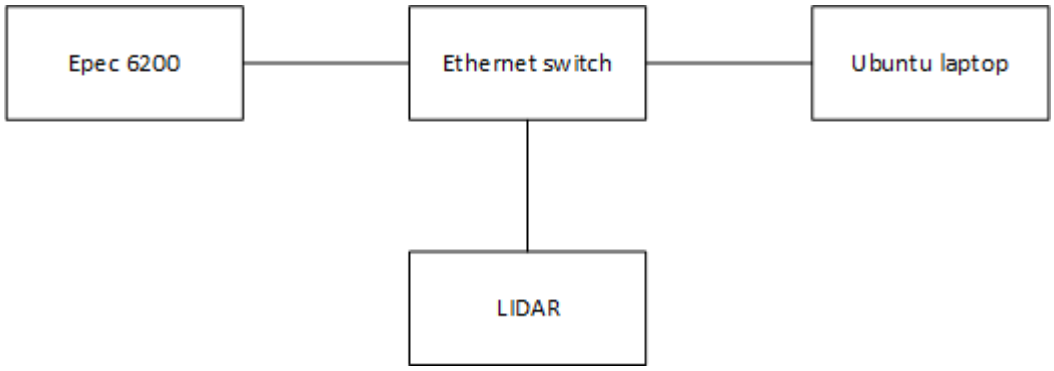

Figure 9. *Illustration of the Ethernet network that is connecting the hardware equipment.*

The software setup that was built to measure the Epec 6200-268 performance with LiDAR applications is based on the DATMO algorithm and its execution (DATMO was

discussed in more detail in chapter 3). To achieve this ROS was selected as an approach since the LiDAR manufacturer supports ROS and supplies a readymade ROS compatible LiDAR driver (ROS was discussed in section 2.2). The LiDAR driver runs as a ROS node and can communicate with other ROS nodes. Thus, it was decided to implement another ROS node that will handle all the computation related to DATMO algorithm and receive the LiDAR point clouds from the LiDAR driver ROS node.

The Epec 6200-268 embedded Linux does not have native support for ROS. To tackle this problem a Docker container is used to run ROS (this is further discussed in section 4.5). The Docker container executes the two ROS nodes: LiDAR driver and DATMO algorithm that are discussed in the following sections 4.3 and 4.4. The deployment diagram of the system can be found in Figure 10 below.

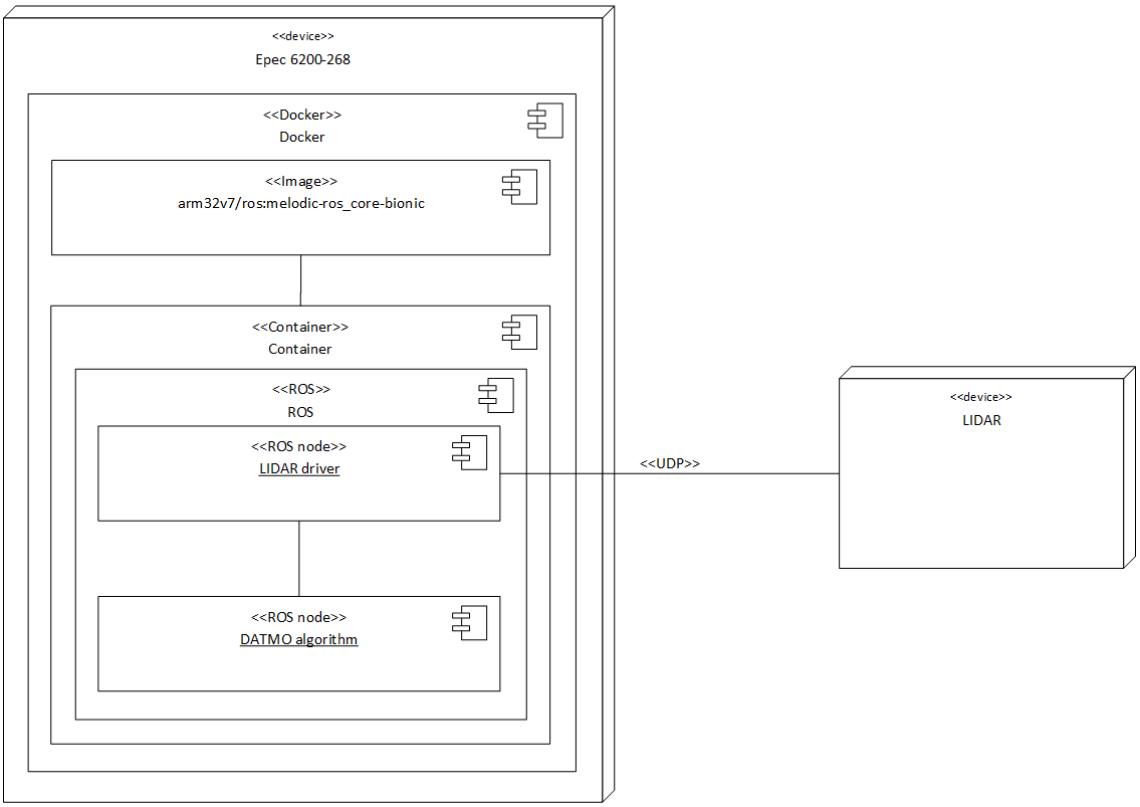

Figure 10. *Deployment diagram of the system*

4.2 Flash LiDAR

The LiDAR used in the hardware setup of this thesis is a flash LiDAR. More information of the flash LiDARs in general can be found from subsection 2.1.3 above. The LiDAR has a 25 Hz repetition rate and a resolution of 4096 pixels. Due to an NDA (non-disclosure agreement) between Epec and the LiDAR manufacturer, no further information can be provided in this thesis.

4.3 ROS node: LiDAR driver

The LiDAR driver node is provided by the LiDAR manufacturer. The LiDAR driver interprets the actual LiDAR data that is received via UDP (User Datagram Protocol) communication from the LiDAR hardware. After interpretation, the LiDAR data is published as ROS topics, which are available for other ROS nodes.

4.4 ROS node: DATMO algorithm

The ROS node for the DATMO algorithm is implemented with *C++* programming language. The algorithm is built for a simple scenario where there are objects in the provided point cloud that do not interfere with one another. This approach simplifies the algorithm implementation significantly. The algorithm detects each object and colours the object red when the object is moving and white when the object is static. An example of a moving object visualization can be seen from Figure 11 below.

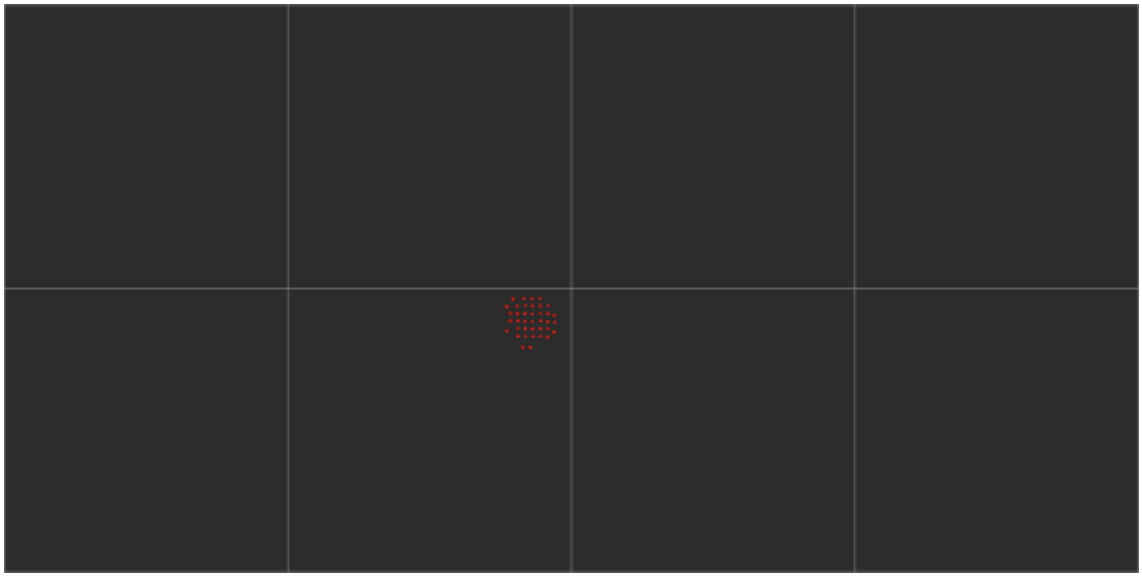

Figure 11. *DATMO algorithm has detected a round object that is moving thus it is coloured red. The screenshot is captured from ROS visualization tool rviz.*

Most of the DATMO algorithm's point cloud handling is implemented utilizing PCL. PCL is a C++ library that is released under the terms of BSD (Berkeley Software Distribution) license [34]. The library offers a lot of tools to help in the development of point cloud applications. In this DATMO algorithm, PCL is utilized, for example, in the plane model segmentation and Euclidean cluster extraction. The usage of PCL is well documented including tutorials and API (application programming interface) documentation. Also the source code is available in a GitHub repository called PointCloudLibrary. More information of the library is available on the PCL website https://pointclouds.org/.

Table 1. *DATMO algorithm parameters.*

Parameter	Data type	Description
enable_pass_through	bool	Enable pass-through filter that filters out all the points that are in the defined pass-through area.
pass_through_min_x	double	Pass-through area x-axis min value (m).
pass_through_max_x	double	Pass-through area x-axis max value (m).
pass_through_min_y	double	Pass-through area y-axis min value (m).
pass_through_max_y	double	Pass-through area y-axis max value (m).
pass_through_min_z	double	Pass-through area z-axis min value (m).
pass_through_max_z	double	Pass-through area z-axis max value (m).

The DATMO algorithm consists of several steps that are executed consecutively on each execution cycle. At the initialization phase, the algorithm subscribes to the LiDAR's ROS topic that contains the point cloud information. The algorithm creates its own ROS topic that will be used to publish a point cloud that contains all the detected static and moving objects. The algorithm utilizes ROS parameters that can be used to configure the algorithm to different scenarios. These parameters are also read in the initialization phase of the algorithm. All the parameters are presented in Table 1 above.

After the initialization phase, the actual point cloud handling starts. At first, the received point cloud is trimmed to a wanted size if pass-through filtering is enabled with the parameter *enable_pass_through*. Figure 12 below shows an example of pass-through filtering. In the example, the pass-through area is 2 meters in width, 2 meters in height and 1 meter in depth.

With the pass-through filter it is possible to reduce the amount of point cloud points that must be processed in the next steps thus making the filter an excellent tool for optimization. This optimization will be utilized in the performance tests that are covered in chapter 5.

After the pass-through filtering, all the point clusters are extracted from the trimmed point cloud with the Euclidean cluster extraction algorithm that is available in PCL. The extracted clusters are the objects that the DATMO algorithm detects and tracks. To track the movement of an object, the centroid of the object's cluster is calculated with a PCL algorithm. On the next cycle, the clusters are again extracted, and centroids calculated, then the centroids from earlier are compared to the new centroids to see if they match to earlier objects. If the match is found and the centroid has moved at least a specified amount, then the object is detected as moving, otherwise static.

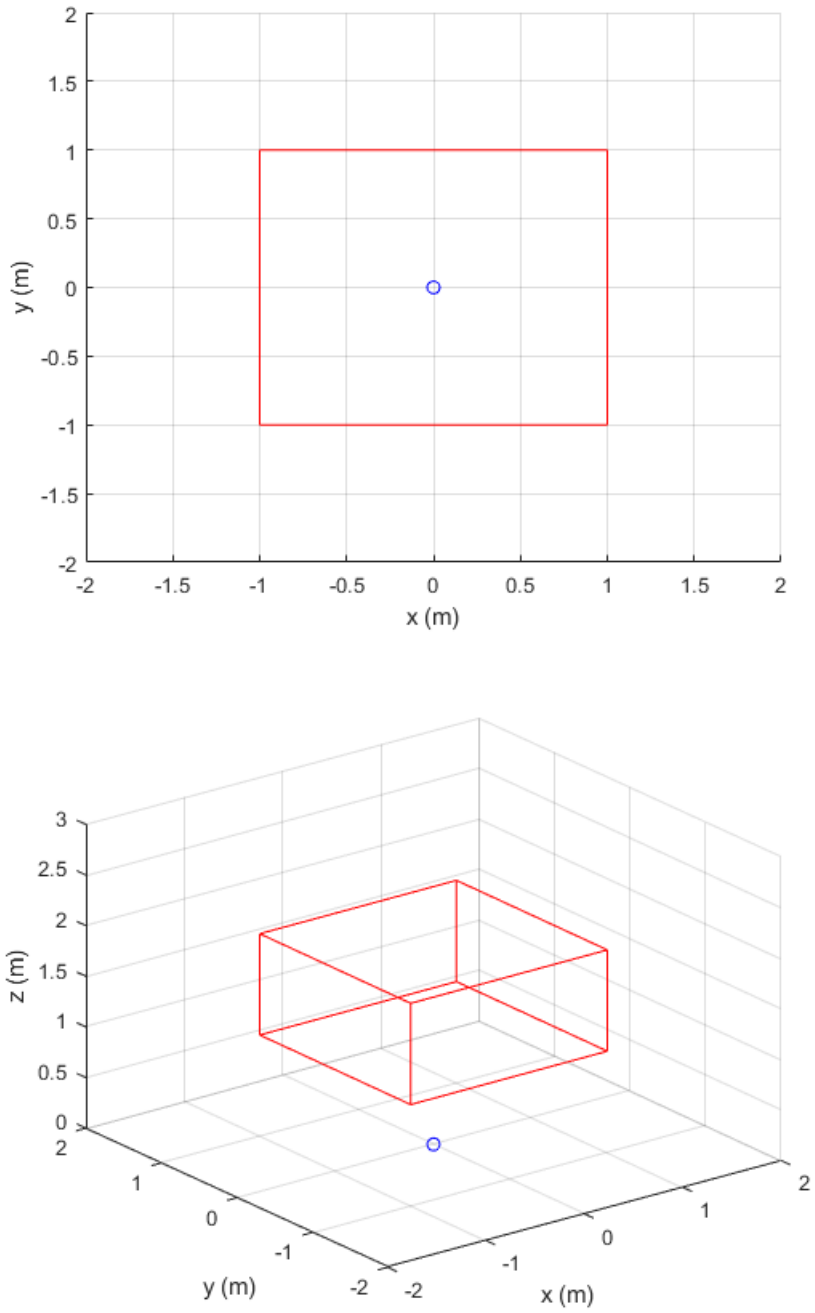

Figure 12. *The image on top is an illustration from LiDAR point of view when using pass-through filtering. The LiDAR is represented as a small blue circle and the area containing all the points after pass-through filtering is represented as a red box. In this case the pass_through_min_x is set to -1, the pass_through_max_x is set to 1, the pass_through_min_y is set to -1, the passthrough_max_y is set to 1, the passthrough_min_z is set to 1 and the passthrough_max_z is set to 2. The bottom image represents the same case but in three-dimensional format.*

4.5 ROS docker image

Since the Epec 6200-268 is based on a tailored Linux, it is not possible to directly install the needed ROS distribution on it, since *ROS Melodic Morenia* is made for Ubuntu Linux [30]. Because of this, a docker image is used that contains the needed ROS distribution and all the dependencies that are needed with the used ROS nodes.

The base image of the docker image is arm32v7/ros:melodic-ros-core-bionic, which contains ROS Melodic Morenia distribution. The image is available in the Docker Hub (https://hub.docker.com/r/arm32v7/ros/). ROS Melodic Morenia is a ROS 1 distribution that was originally released on May 23rd, 2018 [30]. The reason for using an older distribution instead of the newest is to guarantee the compatibility with the LiDAR ROS driver that is provided by the LiDAR manufacturer.

The target environment is based on a 32-bit ARM processor so the ROS nodes must be compiled with that in mind. For this purpose, a docker container is used on a Windows PC and all the nodes are compiled inside this container. The base image for the container is arm32v7/ros:melodic-ros-base that is available in the Docker Hub (https://hub.docker.com/r/arm32v7/ros/).

ROS Melodic Morenia uses catkin as its build system [26]. The installable packages are built with catkin command: *catkin_make install*. The installable package contains */build* and */install* directories that are copied to the catkin workspace of the ROS container that is run in the Epec 6200-268. After this, the package is sourced with the following Bash command in the ROS container: *source /{path_to_ros_workspace}/install/setup.bash*

After all the ROS nodes are built it is possible to run them in the ROS container of the Epec 6200-268 unit. Each ROS node can be launched with the *roslaunch* command. In this thesis' test setup, every ROS node is launched in its own ROS container terminal to be able to monitor the output of each ROS node separately.

4.6 Monitoring

The monitoring of the whole implemented system is done with container terminals and rviz. Container terminals output information of the ROS node states and execution times of the DATMO algorithm. The execution times are used in the test phase of this thesis to determine if the execution is fast enough. The execution times are discussed in more detail in the following chapter 5 that covers the performance tests of the system.

Rviz is used to visualize the point clouds of the LiDAR ROS node and the DATMO algorithm ROS node. With rviz it is possible to determine e.g., if the detected object is

static or moving. The DATMO algorithm colours all the moving objects red and static objects white. An example of this can be found from Figure 11 above.

5. PERFORMANCE TESTS

This chapter covers all the tests that were run to measure the performance of the Epec 6200-268. Section 5.1 specifies the test plan that includes all the different test cases. Section 5.2 introduces the performance measurement principle that was used to get all the measurements. Section 5.3 presents the test results of all the test cases and finally section 5.4 summarizes all the results.

5.1 Test plan

The Epec 6200-268 performance is evaluated with various tests to see if it can manage LiDAR applications. Indicators of the performance are CPU load and DATMO algorithm execution time. The tests are divided into four major test cases:

1. idle
2. LiDAR
3. LiDAR and DATMO algorithm with trimmed view
4. LiDAR and DATMO algorithm with full view.

5.1.1 Test case 1: Idle

In the idle test, the performance is measured in an idle state when none of the ROS nodes are running. The idle performance is measured as a reference point to see how much the system is stressed by the performance measurement script alone. More information of the performance measurement script is available in section 5.2.

5.1.2 Test case 2: LiDAR

In the LiDAR test, the LiDAR ROS driver is running and publishing ROS topics related to LiDAR. The LiDAR ROS driver was discussed in more detail in section 4.3.

The main purpose of this test is to see how much the LiDAR alone stresses the Epec 6200-268. Requirements for this test are that the CPU total load average does not go over 70% and single core load does not peak to 100%. The 70% limit is set as an upper boundary to preserve reasonable load levels in the long run.

5.1.3 Test case 3: LiDAR and DATMO algorithm with trimmed view

In this test, the DATMO algorithm ROS node is also run simultaneously with the LiDAR ROS node. The DATMO algorithm is executed only on a trimmed view that is achieved

by trimming the point clouds points along x-axis, y-axis, and z-axis, with a pass-through filter that was discussed in section 4.4. The purpose of this test is to see if the Epec 6200-268 can handle simple DATMO algorithm together with the LiDAR ROS node. This test consists of three slightly different scenarios to see if varying circumstances affect performance. All the three scenarios are listed below:

- In the first scenario, the trimmed view contains no LiDAR returns. In other words, the point cloud that is passed to the DATMO algorithm is empty.
- In the second scenario, a static object that is both round and flat is introduced in the LiDAR's trimmed area, thus returning points reflected from this object. DATMO algorithm receives the point cloud and tracks the round static object.
- The third scenario is like the second test, but this time the round flat object is constantly moving.

This test used DATMO algorithm parameters that can be found from Table 2 below. The meaning of each parameter was discussed in more detail in section 4.4

Table 2. *DATMO algorithm parameters in Test case 3.*

Parameter	Value
enable_passthrough	true
pass_through_min_x	-1.0
pass_through_max_x	1.0
pass_through_min_y	-1.0
pass_through_max_y	1.0
pass_through_min_z	1.0
pass_through_max_z	2.0

This test has three requirements that shall be passed:

1. CPU total load average does not exceed 70%.
2. None of the single CPU thread loads do not peak to 100%.
3. DATMO algorithm execution time must be so fast that the algorithm can handle all the LiDAR's published point cloud ROS topics. LiDAR publishes its point cloud ROS topic with a frequency of 25 Hz.

5.1.4 Test case 4: LiDAR and DATMO algorithm with full view

In Test case 4, the DATMO algorithm ROS node is also run simultaneously with the LiDAR ROS node, like in Test case 3. This time the pass-through filter is disabled and the DATMO algorithm receives the full view of LiDAR point cloud. The purpose of this test is to find the limit of the Epec 6200-268 by increasing the size and complexity of the received point cloud. This test case is run in an office environment where the point cloud that the LiDAR produces, contains multiple elements including walls, office furniture and

various other items that can be found from an office space. The implemented DATMO algorithm is not designed for such a complicated scenario so the algorithm operation is not flawless, but it can still be used to load the CPU even though it is not performing its designated task perfectly.

This test used DATMO algorithm parameters that can be found from Table 3 below. The meaning of each parameter was discussed in a more detail in section 4.4.

Table 3. *DATMO algorithm parameters in Test case 4.*

Parameter	Value
enable_passthrough	false
pass_through_min_x	-1.0
pass_through_max_x	1.0
pass_through_min_y	-1.0
pass_through_max_y	1.0
pass_through_min_z	1.0
pass_through_max_z	2.0

This test has three requirements that shall be passed:

1. CPU total load average load does not exceed 70%.
2. None of the single CPU thread loads do not peak to 100%.
3. DATMO algorithm execution time must be so fast that the algorithm can handle all the LiDAR's published point cloud ROS topics. LiDAR publishes its point cloud ROS topic with a frequency of 25 Hz.

5.2 Performance measurement principle

The CPU load of each Epec 6200-268 CPU core is logged to a log file with an interval of approximately 1000 ms to find out how much the CPU is stressed during different test cases. The CPU load is parsed from Linux kernel statistics that are available in the Epec 6200-268 file system in file */proc/stat*. The file contains the following metrics for every CPU core: *user*, *nice*, *system*, *idle*, *iowait*, *irq*, *softirq*, *steal*, *guest,* and *guest_nice* [19]. All the metrics are explained in Table 4 below.

Table 4. *Kernel/system statistics like described in [19].*

Value	Description
user	(1) Time spent in user mode.
nice	(2) Time spent in user mode with low priority (nice).
system	(3) Time spent in system mode.
idle	(4) Time spent in the idle task. This value should be USER_HZ times the second entry in the */proc/uptime* pseudo-file.
iowait	(since Linux 2.5.41) (5) Time waiting for I/O to complete. This value is not reliable, for the following reasons: 1. The CPU will not wait for I/O to complete; iowait is the time that a task is waiting for I/O to complete. When a CPU goes into idle state for outstanding task I/O, another task will be scheduled on this CPU. 2. On a multi-core CPU, the task waiting for I/O to complete is not running on any CPU, so the iowait of each CPU is difficult to calculate. 3. The value in this field may *decrease* in certain conditions.
irq	(since Linux 2.6.0) (6) Time servicing interrupts.
softirq	(since Linux 2.6.0) (7) Time servicing softirqs.
steal	(since Linux 2.6.11) (8) Stolen time, which is the time spent in other operating systems when running in a virtualized environment
guest	(since Linux 2.6.24) (9) Time spent running a virtual CPU for guest operating systems under the control of the Linux kernel.
guest_nice	(since Linux 2.6.33) (10)Time spent running a niced guest (virtual CPU for guest operating systems under the control of the Linux kernel).

Total CPU core time is calculated using equation 5.1, where subscript *x* stands for the CPU core number:

$$total_x = user_x + nice_x + system_x + idle_x + iowait_x + irq_x + softirq_x + steal_x + guest_x + guestNice_x. \quad (5.1)$$

The actual CPU core specific time that is spent running user or system processes is calculated using equation 5.2, where the subscript *x* stands for the CPU core number:

$$usage_x = user_x + system_x\,. \quad (5.2)$$

The CPU utilization percentage is calculated for each core with equation 5.3, by using the two consecutive total and usage measurements of the CPU core, here noted by postfix *new* and *old*. In equation 5.3, the subscript *x* stands for the CPU core number.

$$CPU_x = \frac{usageNew_x - usageOld_x}{totalNew_x - totalOld_x} * 100\%. \quad (5.3)$$

The CPU total utilization percentage is the arithmetic mean of all the four CPU cores (denoted from 0 to 3) utilization percentages. The CPU total utilization percentage is calculated by equation 5.4.

$$CPU_{total} = \frac{1}{4}(\sum_{x=1}^{4} CPU_{x-1}). \quad (5.4)$$

The CPU average utilization percentage is the arithmetic mean of all *n* measurement samples CPU_x (or CPU_{total}) that are taken during a test run and saved into a set of measurements *M*. The average of the set *M* values is calculated by equation 5.5:

$$average = \frac{1}{n}(\sum_{i=1}^{n} M_i). \quad (5.5)$$

The CPU max utilization percentage is the greatest value of all *n* measurement samples CPU_x (or CPU_{total}) that are taken during a test run and saved into a set of measurements *M*.

The DATMO algorithm execution time measurement is implemented in the algorithm itself and the execution times are saved to a separate log file.

5.3 Test results

This section introduces all the results of each test case. Every test case has its own named subsection.

5.3.1 Test case 1

Test case 1 for idle performance that was discussed earlier in subsection 5.1.1 was performed first and showed that the total CPU load was 2.38% on average and peaked to maximum of 4.01%. The maximum value for a single core was 5.66%. All the measured CPU load average and maximum values can be found from Table 5 below.

Table 5. *CPU load measurements in Test case 1.*

measurement	**CPU_0**	**CPU_1**	**CPU_2**	**CPU_3**	**CPU_{total}**
average	2.71%	2.37%	2.29%	2.41%	2.38%
maximum	5.66%	4.72%	4.72%	4.72%	4.01%

CPU total load stayed quite steady throughout the whole test period as can be seen from Figure 13 below. It seems that the idle load and the performance measurement that is triggered approximately every 1 second does not load the CPU too much.

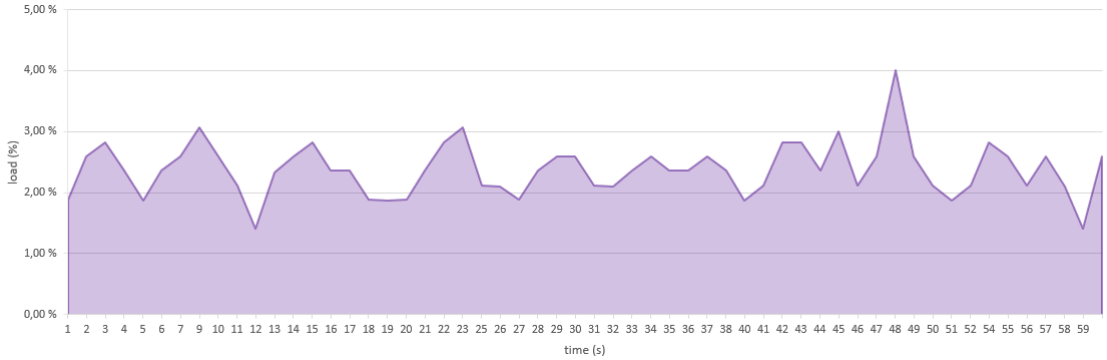

Figure 13. *CPU total load in Test case 1.*

5.3.2 Test case 2

In Test case 2 that was discussed with more detail in subsection 5.1.2, the CPU average load was 9.60% and this time the single core maximum peaked at 29.63%. The single core maximum value compared to the CPU total load average indicates that the LiDAR ROS driver is executed in single core. All the CPU load measurements can be seen from Table 6 below.

Table 6. *CPU load measurements in Test case 2.*

measurement	**CPU_0**	**CPU_1**	**CPU_2**	**CPU_3**	**CPU_{total}**
average	19.62%	6.79%	5.87%	6.11%	9.60%
maximum	29.63%	16.82%	17.76%	19.63%	11.68%

As in Test case 1, this time also the CPU total load stayed quite steady throughout the whole test like seen in Figure 14 below. The single core loads were fluctuating quite heavily but it seems that most of the time the LiDAR ROS driver was loading the first core CPU_0, as seen in Figure 15 below that represents all cores' load values separately.

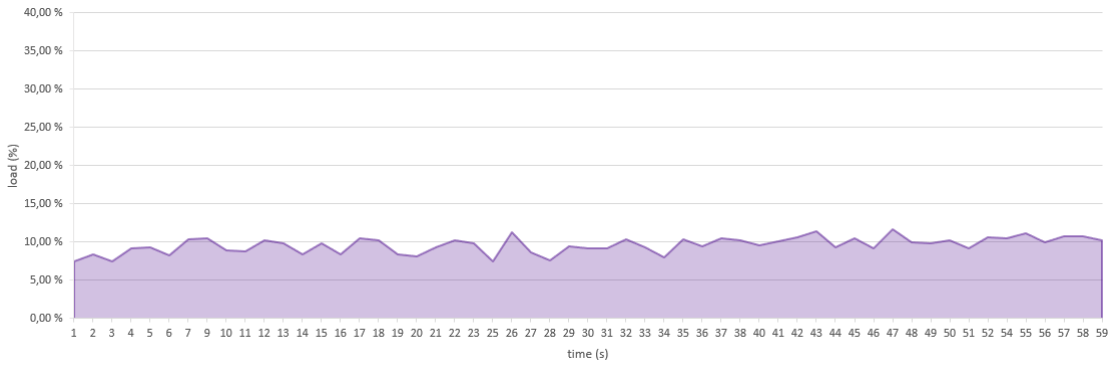

Figure 14. *CPU total load in Test case 2.*

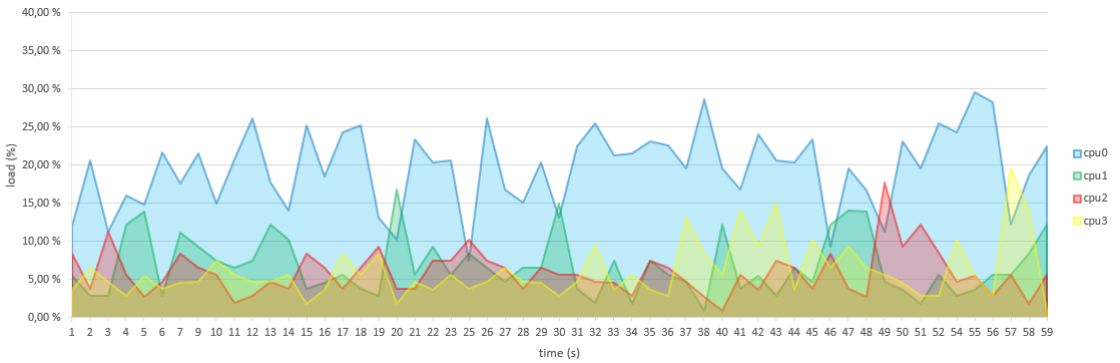

Figure 15. *CPU load of every core in Test case 2.*

5.3.3 Test case 3

Test case 3 was divided into three scenarios that were discussed earlier in subsection 5.1.3. Each individual scenario is presented in this subsection.

Test case 3 part 1:

In the first part, the DATMO algorithm was tested without any object. CPU load averaged at 21.34% and the single core maximum peaked at 68.81% as shown in Table 7.

Table 7. *CPU load measurements in Test case 3 part 1.*

measurement	CPU_0	CPU_1	CPU_2	CPU_3	CPU_{total}
average	26.29 %	21.38%	16.44%	21.26%	21.34%
maximum	68.81%	52.83%	54.63%	53.27%	24.76%

In the first part of Test case 3, the CPU total load was once again steady throughout the test, as seen in Figure 16 below.

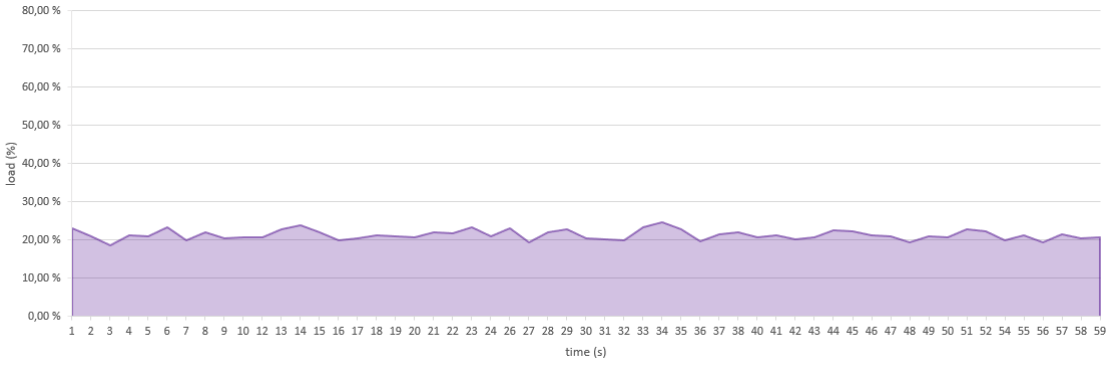

Figure 16. *CPU total load in Test case 3 part 1.*

In this test, the single core loads were fluctuating a lot more and there was not one specific core that had a significantly higher load than the others, like Table 7 above suggested. All the single core loads are shown in Figure 17 below. It seems that the

DATMO algorithm is run alternating on every core and the LiDAR ROS driver is run most of the time in the CPU_0 (blue) since it seems to have the highest base load.

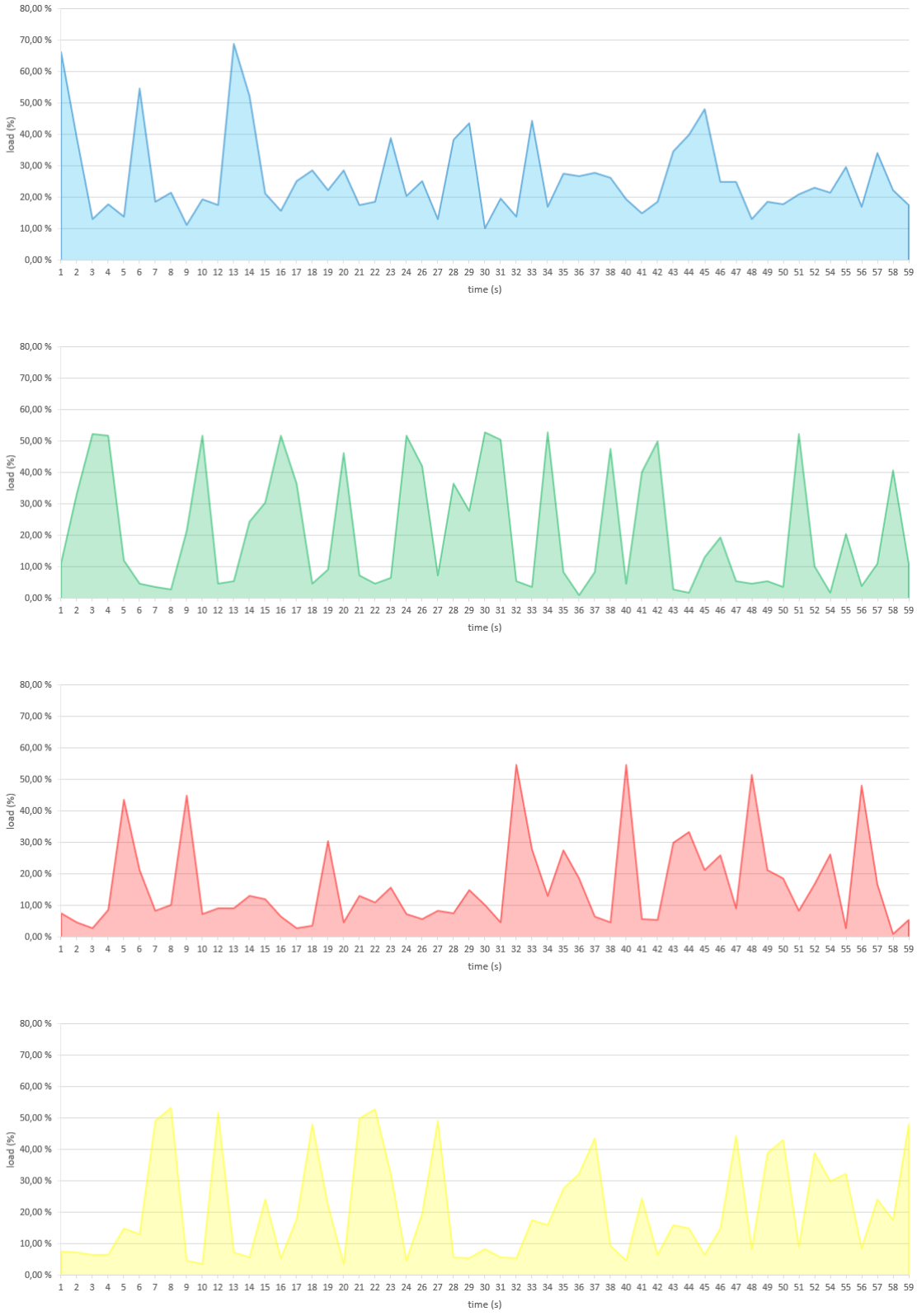

Figure 17. *CPU_0, CPU_1, CPU_2 and CPU_3 loads from top to bottom in Test case 3 part 1.*

Test case 3 part 2:

When running the DATMO with a static object in sight, the results did not differ that much compared to DATMO without any object. However, the maximum single core CPU load peaked at 58.33% which is about 10 percentage points less compared to DATMO without any object in Test case 3 part 1. This could be just a coincidence since the CPU_0 peaked over 60.00% only two times within the one minute test period in Test case 3 part 1, as can be seen from Figure 17 above. All the CPU load measurements of Test case 3 part 2 can be seen in Table 8 below.

Table 8. *CPU load measurements in Test case 3 part 2.*

measurement	**CPU_0**	**CPU_1**	**CPU_2**	**CPU_3**	**CPU_{total}**
average	16.12%	29.52%	20.42%	15.08%	20.28%
maximum	46.73%	55.56%	58.33%	58.33%	23.84%

The CPU total load did not fluctuate too much, as can be seen from Figure 18 below. Also, the single core loads seem to follow the same trend as in Test case 3 part 1. It still seems that DATMO algorithm is run on a random core and that one of the cores is alternately peaking higher than the others, as can be seen in Figure 19 below.

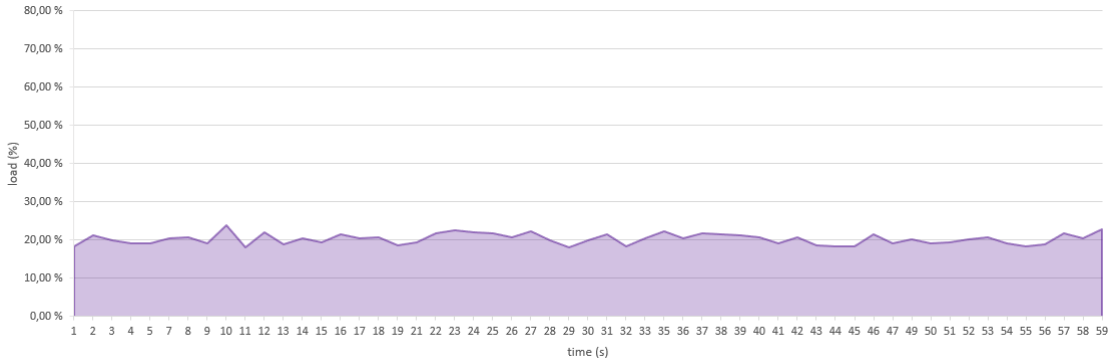

Figure 18. *CPU total load in Test case 3 part 2.*

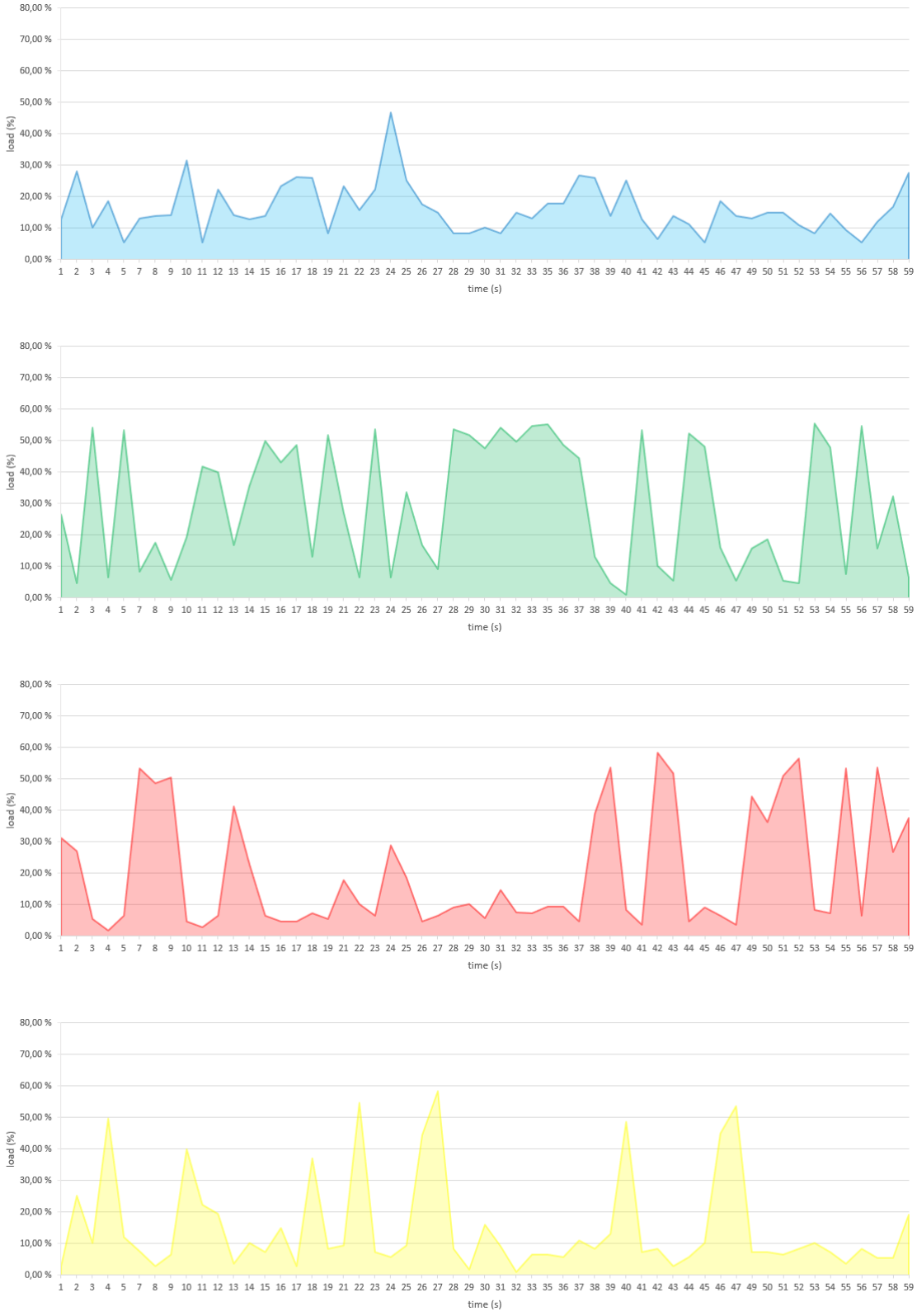

Figure 19. *CPU_0, CPU_1, CPU_2 and CPU_3 loads from top to bottom in Test case 3 part 2.*

Test case 3 part 3:

Part 3 of Test case 3 was like the two ones before, but this time the object was moving. The CPU total average and maximum loads seem to increase slightly compared to the previous tests. The results are listed in Table 9 below.

Table 9. *CPU load measurements in Test case 3 part 3.*

measurement	CPU_0	CPU_1	CPU_2	CPU_3	CPU_{total}
average	30.64%	19.20%	19.68%	24.28%	23.45%
maximum	56.07%	50.00%	53.70%	57.41%	27.55%

Otherwise, the CPU total and single core loads seemed to follow the same pattern as in Test case 3 part 1 and Test case 3 part 2. All the CPU load trends can be seen in Figure 20 and Figure 21 below.

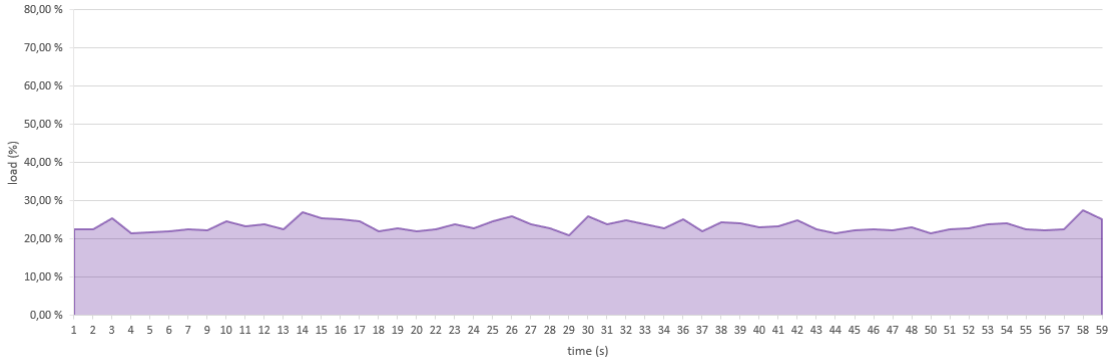

Figure 20. *CPU total load in Test case 3 part 3.*

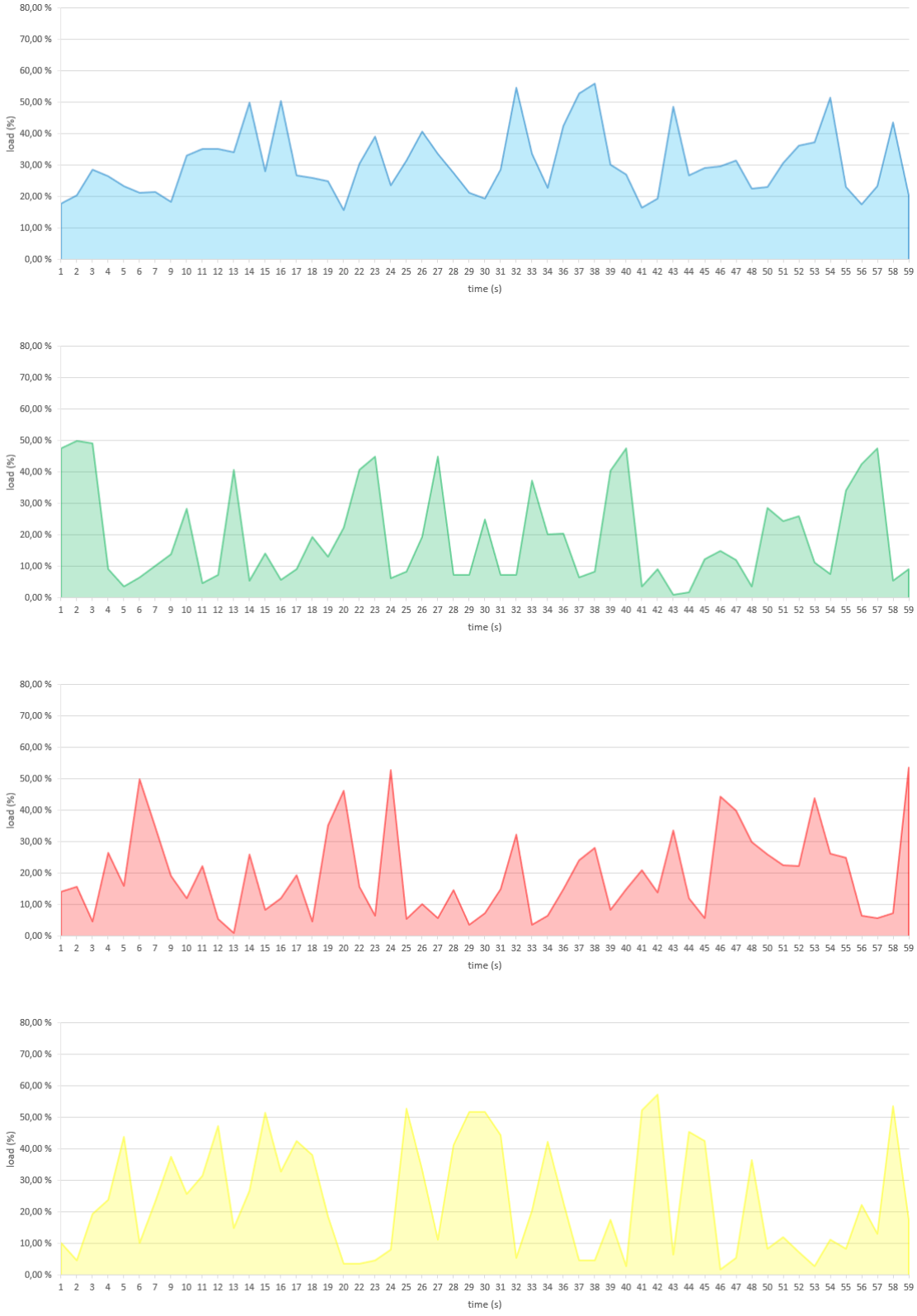

Figure 21. *CPU_0, CPU_1, CPU_2 and CPU_3 loads from top to bottom in Test case 3 part 3.*

Test case 3 summary:

The algorithm execution in all three tests always stayed below 30 ms, so the algorithm can be executed 33 times per second, like shown in formula (6):

$$\frac{1000\ ms}{30\ ms} \approx 33. \tag{6}$$

This indicates that the algorithm can process all the incoming point clouds from the LiDAR that is generating point clouds with a frequency of 25 Hz like was discussed in section 4.2. LiDAR and DATMO with trimmed view didn't surpass the threshold limits that were defined in subsection 5.1.3 and thus passed all the tests in Test case 3.

5.3.4 Test case 4

The last test was performed with LiDAR and DATMO algorithm with full view that was described in subsection 5.1.4. The CPU total load averaged 34.63% and peaked at 38.07%. Even though the average load seemed to be at acceptable levels, the single core performance was not enough to pass this test since the single core load peaked at 100.00% on three out of four cores. All the CPU load results can be seen in Table 10 below.

Table 10. *CPU load measurements in Test case 4.*

measurement	CPU_0	CPU_1	CPU_2	CPU_3	CPU_{total}
average	33.32%	42.26%	23.43%	39.52%	34.63%
maximum	96.26%	100.00%	100.00%	100.00%	38.07%

CPU total load graph can be seen in Figure 22. The CPU core specific loads are represented in Figure 23.

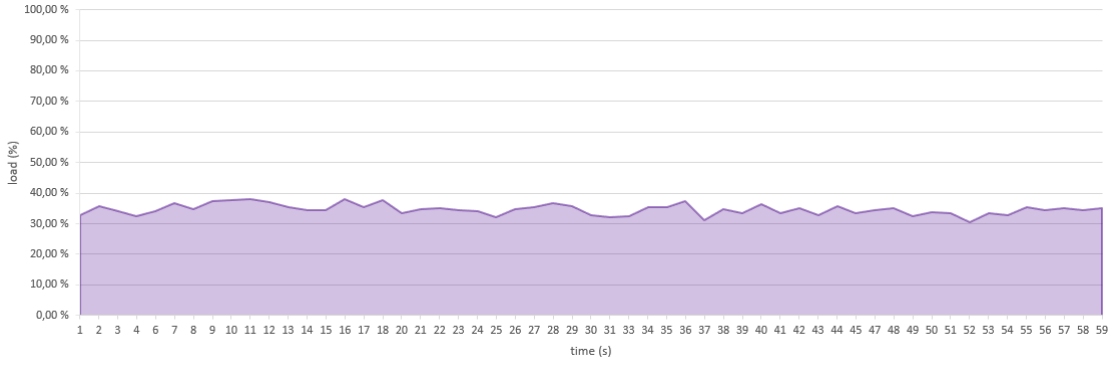

Figure 22. *CPU total load in Test case 4.*

Figure 23. *CPU_0, CPU_1, CPU_2 and CPU_3 loads from top to bottom in Test case 4.*

In Test case 4, the algorithm execution time also surpassed 200 ms which, means that the algorithm cannot process all the LiDAR's point clouds that are generated with a frequency of 25 Hz. One reason for the long execution time could be that in this test the algorithm tracked approximately 16 objects compared to just one object in Test case 3.

5.4 Test results summary

As seen from the test results that are summarized in Table 11, most of the tests passed. The only failed tests occurred in Test case 4 that was designed to find the limits of the Epec 6200-286 CPU performance, like described in subsection 5.1.4. All in all, the Epec 6200-268 seems to have adequate CPU performance for simple LIDAR applications.

Table 11. *Test results summary.*

Test case	CPU total load	CPU single core load	DATMO algorithm execution time
1	-	-	-
2	PASSED	PASSED	-
3	PASSED	PASSED	PASSED
4	PASSED	FAILED	FAILED

6. CONCLUSIONS

The results of this thesis showed that it is possible to use LiDAR with the Epec 6200-268 and that it is possible to run a simple LiDAR application with the Epec 6200-268, although it is not specifically designed to be used with perception applications. Instead, the Epec 6200-268 is designed to be used as an IoT device and its processor is designed for automotive and infotainment applications [23], as was discussed in subsection 2.3.1. Although the results were promising for a simple LiDAR application, as shown in subsection 5.3.3, it was clearly shown in subsection 5.3.4 that the Epec 6200-268 cannot keep up with a more demanding LiDAR application, where the amount of data to be processed is increased.

In the Test case 3 (that was discussed in subsections 5.1.3 and 5.3.3) the DATMO algorithm was run in a trimmed view thus presenting a *simple LiDAR application*. The point cloud was trimmed with the help of a group of parameters that were implemented in the DATMO algorithm thus allowing to trim the received point cloud to a needed size. In this test case all the test requirements were reached since the CPU total load average, which represents the average load of all four CPU cores, did not exceed 70%, but instead stayed below 24% in all three parts of Test case 3. Also, the CPU total load maximum value always stayed below 28%. The limit of CPU single core maximum load was not reached since the CPU single core maximum values stayed below 70% in all three parts of the Test case 3, when the single core maximum load limit was set to 100%. Also, the last requirement of DATMO algorithm execution time was fulfilled since the execution time stayed below 30 ms when the limit was 40 ms.

In Test case 4 (that was discussed in subsections 5.1.4 and 5.3.4), where the DATMO algorithm was run in a full view thus presenting the *demanding LiDAR application*, all the test requirements were not met. The CPU total load average test was successful since the CPU total load average stayed below 35% when the limit was 70%. Also, the CPU total maximum load just lightly surpassed 38% with a maximum value of 38.07%. Unfortunately, the remaining two requirements were not met since the CPU single core maximum level reached the limit of 100% in three out of four cores, thus failing the second requirement. Also, the DATMO algorithm execution time exceeded the set limit of 40 ms with a wide margin surpassing 200 ms.

As can be seen above from the summary of the two most important test cases the LiDAR application used in this thesis does not load the CPU evenly since the CPU total load

maximum was significantly lower than the CPU single core maximum load. In Test case 3, the CPU total load maximum stayed below 28% when the CPU single core load maximum stayed just below 70%, thus making the difference between the two about 40 percentage points. In Test case 3, on the other hand, the CPU total load maximum stayed below 35% when the CPU single core load maximum reached 100% resulting in an even higher difference between the two with about 65 percentage points.

From the results, a conclusion can be drawn that the DATMO algorithm used in this thesis was loading only single core of the CPU most of the time. With this information it is safe to say that if the Epec 6200-268 is used with a LiDAR application in the future it is beneficial to further study and investigate the possibilities of ROS on how to distribute the load to all four CPU cores more evenly. With this approach it might be possible to get more performance out of the Epec 6200-268 since as shown in the results the margin between the CPU total load maximum and CPU single core load maximum is quite wide and suggests that there is quite a lot more unused CPU performance available to be further utilized.

Another way to improve the Epec 6200-268 performance with DATMO algorithm especially is to further study the algorithm implementation and see if there are possibilities to increase the efficiency of it. As can be seen from the results, the parametrization acts a major role when it comes to performance of the algorithm. The simple LiDAR application was parametrized in a way that it only receives a trimmed view of the point cloud, thus reducing the amount of data it must process, thus improving the performance of the algorithm. Although, it is good to keep in mind that trimming the point cloud is quite a harsh way to reduce the needed computational capacity of the DATMO algorithm and this may not be a suitable way to optimize the algorithm in many LiDAR applications. Other possible ways to parametrize the algorithm include adjusting the inputs of PCL functions, which are used by the DATMO algorithm. Adjustments of the PCL functions inputs offers a wide variety of possibilities to further optimize the actual DATMO algorithm itself.

In addition, further studying the possibilities of the Epec 6200-268 CPU might offer some possibilities to further optimize the implementation. It might have a positive impact on the performance of the overall system if every bit of CPU power is harnessed on the use of LiDAR application. However, there is also a limit of time that is reasonable to use in the optimization of the algorithm since time is money.

However, it might not be possible to implement more demanding LiDAR applications like SLAM (simultaneous localization and mapping) in a way that the Epec 6200-268 can

handle them seamlessly, at least not with a three-dimensional point cloud. It is good to keep in mind that the DATMO algorithm that was implemented for this thesis is run in a way that the LiDAR remains static. If the LiDAR sensor position would change dynamically this would add more requirements to the DATMO algorithm itself. In a dynamic scenario the DATMO algorithm should be aware, for example, of the position and the angle of the LiDAR, thus adding more computation to the mix. This would increase the amount of code that must be executed in the Epec 6200-268, thus reducing the overall performance.

To summarize, the objectives of this thesis were met, thus forming a better image of the capabilities of the Epec 6200-268 with LiDAR applications. All the research questions defined in chapter 1 were answered and suggestions and recommendations for future research were introduced.

REFERENCES

[1] J. Abshire, NASA's Space Lidar Measurements of Earth and Planetary Surfaces, NASA, Jan 2010. Available (accessed on 4.6.2023): https://ntrs.nasa.gov/citations/20100031189

[2] Cortex-A9 NEON Media Processing Engine, Technical Reference Manual, Revision: r4p1, ARM. Available: https://documentation-service.arm.com/static/5e8e176afd977155116a3c45?token=

[3] Roborock Robotic Vacuum Cleaner Manual, Beijing Roborock Technology Co. Ltd., 5 p. Available (accessed on 4.6.2023): https://support.roborock.com/hc/en-us/article_attachments/360045870291/S5_CE_EN_manual.pdf

[4] i.MX6 is officially launched, Boundary Devices, Nov 2012. Available (accessed on 31.5.2023): https://boundarydevices.com/i-mx6-is-officially-launched/

[5] F. Bretenaker, N. Treps, Laser : 50 Years of Discoveries, World Scientific Publishing Co. Pte. Ltd., Singapore, 2015, 1 p.

[6] J. Campbell, Introduction to Remote Sensing, Fourth Edition, Taylor & Francis, Abingdon, UK, 2006, pp. 239–249.

[7] P. Chazette, J. Totems, L. Hespel, J. Bailly, Optical Remote Sensing of Land Surface, 2016, pp. 207–217. Available: https://www.researchgate.net/publication/308941555_Principle_and_Physics_of_the_LiDAR_Measurement

[8] T. Darrell, G. Shakhnarovich, P. Indyk, Nearest-neighbor methods in learning and vision : theory and practice, The MIT Press, Cambridge, Massachusetts, USA, 2005, 1 p.

[9] Use containers to Build, Share and Run your applications, Docker Inc. Available (accessed on 15.4.2023): https://www.docker.com/resources/what-container/

[10] A. El-Rabbany, Introduction to GPS : The Global Positioning System, Artech House, Boston, Massachusetts, USA, 2002, 4 p.

[11] Displays, Epec Oy. Available (accessed on 4.6.2023): https://epec.fi/epec-oy-products/displays/

[12] Epec 6200 Remote Access Unit, Epec Oy. Available (accessed on 15.4.2023): https://epec.fi/epec-oy-products/connectivity/telematics/control-system-products-for-iot-6200/

[13] Epec Flow Mobile Electrification Systems, Epec Oy. Available (accessed on: 30.5.2023): https://epec.fi/systems/epec-flow/

[14] Epec XS6C Central Unit, Epec Oy, Available (accessed on 15.4.2023): https://epec.fi/products/control-system-products-xs6c/

[15] A short story about LiDAR technology, Geo-Plus inc. Available (accessed on: 15.4.2023): https://geo-plus.com/blog-a-short-story-of-lidar-technology/

[16] ARM-Software/ComputeLibrary, GitHub, Available (accessed on: 3.6.2023): https://github.com/ARM-software/ComputeLibrary

[17] R. Horaud, M. Hansard, G. Evangelidis, C. Ménier, An overview of depth cameras and range scanners based on time-of-flight technologies, Machine Vision and Applications, Vol. 27, Iss. 7, Springer Nature, 2016, pp. 1005–1009. Available: https://doi.org/10.1007/s00138-016-0784-4

[18] A. Incoronato, M. Locatelli, F. Zappa, Statistical Modelling of SPADs for Time-of-Flight LiDAR, Sensors, Vol. 21, Iss. 13, 2021, pp. 4481–.

[19] M. Kerrisk, proc(5) – Linux manual page, man-pages. Available (accessed on 15.4.2023): https://man7.org/linux/man-pages/man5/proc.5.html

[20] M. Khader, S. Cherian, An Introduction to Automotive LIDAR, Texas Instruments, May 2020, pp. 2–4. Available: https://www.ti.com/lit/SLYY150A

[21] P. Kumar Rath, A. Ramirez-Serrano, D. Kumar Pratihar, Real-time moving object detection and removal from 3D pointcloud data for humanoid navigation in dense GPS-denied environments, Engineering Reports, Vol. 2, Iss. 12, 2020.

[22] Á, Llamazares, E. Molinos, M. Ocaña, Detection and Tracking of Moving Obstacles (DATMO): A Review, Robotica, Vol. 38, Iss. 5, Cambridge University Press, 2019, pp. 761–765.

[23] i.MX 6Dual/6Quad Automotive and Infotainment Applications Processors, NXP B.V., Document Number: IMX6DQAEC, Rev. 6, Nov 2018, pp. 1–14. Available: https://www.nxp.com/docs/en/data-sheet/IMX6DQAEC.pdf

[24] i.MX 6Quad Processors - High-Performance, 3D Graphics, HD Video, Arm® Cortex®-A9 Core, NXP B.V., Jun 2022. Available (accessed on 4.6.2023): https://www.nxp.com/assets/block-diagram/en/i.MX6Q.pdf

[25] i.MX 8M Plus – Arm® Cortex®-A53, Machine Learning, Vision, Multimedia and Industrial IoT, NXP Semiconductors. Available (accessed on 31.5.2023): https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-8-applications-processors/i-mx-8m-plus-arm-cortex-a53-machine-learning-vision-multimedia-and-industrial-iot:IMX8MPLUS

[26] catkin, Open Robotics, Jul 2017. Available (accessed on 3.6.2023): http://wiki.ros.org/catkin

[27] Parameter Server, Open Robotics. Available (accessed on 16.4.2023): http://wiki.ros.org/Parameter%20Server

[28] ROS Concepts, Open Robotics. Available (accessed on 15.4.2023): http://wiki.ros.org/ROS/Concepts

[29] ROS Introduction, Open Robotics. Available (accessed on 15.4.2023): http://wiki.ros.org/ROS/Introduction

[30] ROS Melodic Morenia, Open Robotics. Available (accessed on 15.4.2023): http://wiki.ros.org/melodic

[31] rviz, Open Robotics. Available (accessed on 16.4.2023): http://wiki.ros.org/rviz

[32] F. Petit, The beginnings of LiDAR – A time travel back in history, Blickfeld Blog, Apr 2020. Available (accessed on 15.4.2023): https://www.blickfeld.com/blog/the-beginnings-of-lidar/

[33] centroid.hpp, Point Cloud Library. Available (accessed on 11.5.2023): https://pointclouds.org/documentation/centroid_8hpp_source.html#l00056

[34] What is PCL?, Point Cloud Library. Available (accessed on 4.6.2023): https://pointclouds.org/about/

[35] Ponsse launches new technology: an electric forest machine, Ponsse Oyj, Aug 2022. Available (accessed on 28.5.2023): https://www.ponsse.com/en/company/news/-/asset_publisher/P4s3zYhpxHUQ/content/ponsse-launches-new-technology-an-electric-forest-machine#/

[36] R. Steinmetz, K. Wehrle, Peer-to-peer systems and applications, Springer Nature, 2005, 11 p. Available: https://doi.org/10.1007/11530657

[37] Uuden tason turvallisuutta seuraavan sukupolven Volvoissa, Volvo Car Corporation, Jun 2021. Available (accessed on 16.4.2023): https://www.volvocars.com/fi/news/innovation/new-safety-standard-comes-to-the-next-generation-of-volvos/