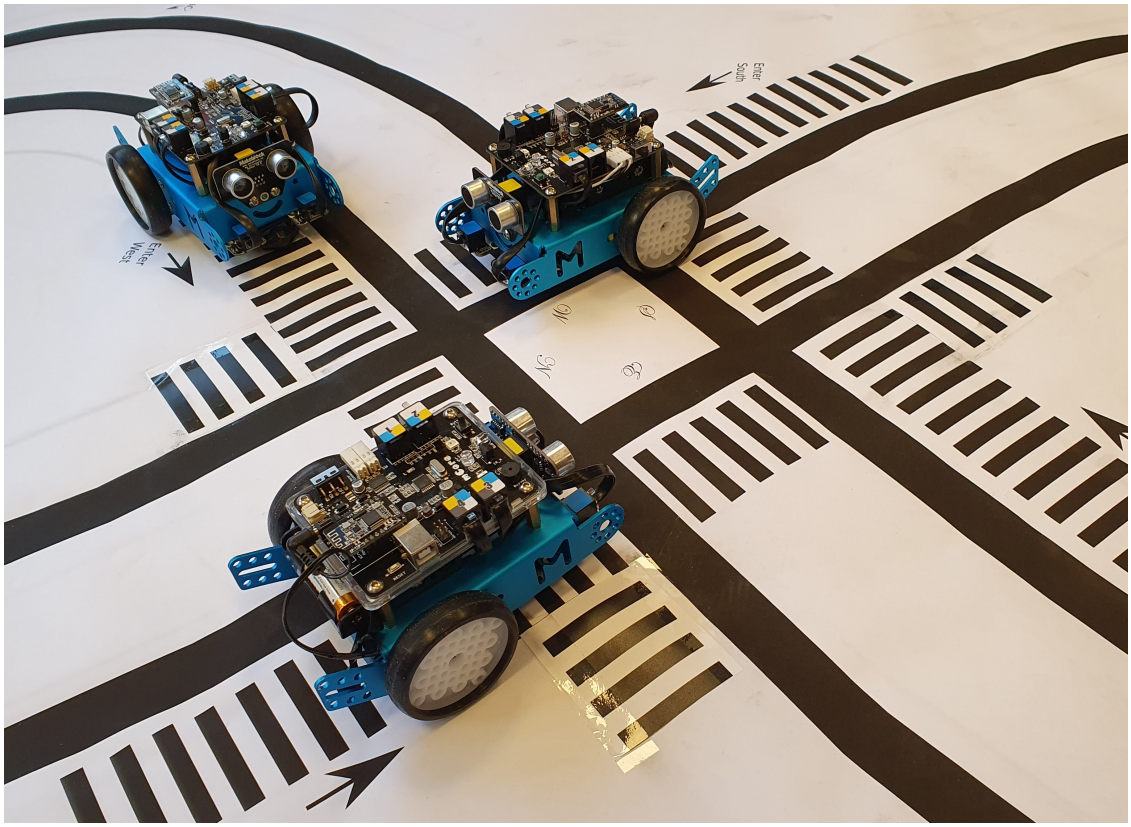

Vehicle Communication

A study of vehicle communication in a modeled environment

Bachelor's thesis in Computer Science and Engineering

Mattias Sikvall Källström
Eric Rylander

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

DEGREE PROJECT REPORT

A study of vehicle communication in a modeled environment

How robots can be used to model a traffic situation with an intersection and avoid collisions between multiple robots

Mattias Sikvall Källström
Eric Rylander

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

A study of vehicle communication in a modeled environment
How robots can be used to model a traffic situation with an intersection and avoid collisions between multiple robots
Mattias Sikvall Källström
Eric Rylander

Examiner: Jonas Duregård, Chalmers University of Technology, Department of Computer Science and Engineering

Supervisor: Sakib Sistek, Chalmers University of Technology, Department of Computer Science and Engineering
Supervisor: Henrik Lundqvist, Cybercom

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover: Photo of the modeled environment.

Typeset in LaTeX
Department of Computer Science and Engineering
Gothenburg 2020

Abstract

This report deals with the topic of vehicle communication which includes communication between vehicles and other entities such as infrastructure. The purpose of the report is to modulate a vehicle environment where communication between vehicles and infrastructure can be tested. With studies from both literature and experiments, this report aims to answer the question of how safety, efficiency and environmental impact can be improved with the help of vehicle communication. With the aid of existing hardware and proprietary software, tests have been developed to test vehicle communication. Four applications have been developed for the project, one for the integrated system in the vehicles, one system for the vehicle computers, one system for the infrastructure and finally a debugging software. These systems have made it possible to answer the research questions. The results of the tests show that in the modulated environment it is possible to eliminate collisions from occurring. The tests also show how waiting time and throughput transpires at the intersection based on the application of different algorithms. With a queue algorithm it is possible to get a better flow at the intersection than it does with an algorithm that mimics a traffic light. Conclusions that can be drawn from the project are that vehicle communication can improve road safety and efficiency. Vehicle communication can be implemented in infrastructure so that a significant reduction in accidents occurs. Implementations can also save the environment and be implemented in an ethical way.

Keywords: Vehicle Communication, ICWS, V2I, V2X, Intelligent Traffic, IoT.

Sammandrag

Denna rapport behandlar ämnet fordonskommunikation som innefattar kommunikation mellan fordon och andra entiteter så som infrastruktur. Syftet med rapporten är att modulera en fordonsmiljö där kommunikation mellan fordonen och infrastruktur kan testas. Med studier från både litteratur och experiment önskar denna rapport svara på frågan hur säkerhet, effektivitet och miljöpåverkan kan förbättra med hjälp av fordonskommunikation. Med hjälp av existerande hårdvara och egenutvecklad mjukvara har tester kunnat tas fram för att testa fordonskommunikationen. Fyra applikationer har utvecklats för projektet, en för det integrerade system i fordonen, ett system för fordonsdatorerna, ett system för infrastrukturen och slutligen en felsökningsprogramvara. Dessa system har gjort det möjligt att besvara frågeställningen. Resultaten från testerna visar att i den modulerade miljön går det att eliminera kollisioner från att inträffa. Testerna visar också på hur väntetid och genomströmning sker i korsning utifrån att olika algoritmer tillämpas. Med en kö-algoritm går det att få ett bättre flöde i korsningen än vad det gör med en algoritm som efterliknar ett trafikljus. Slutsatser som kan dras utifrån projektet är att fordonskommunikation har möjligheten att förbättra trafiksäkerhet och effektivitet. Det går att implementera fordonskommunikation i infrastruktur så att en markant minskning av olyckor sker. Implementationer kan också innebära en besparing av miljön och genomförs på etiskt sätt.

Keywords: Fordonskommunikation, ICWS, V2I, V2X, Intelligent Trafik, IoT.

Acknowledgements

This project was performed by students at the bachelor programs with a specialization in Computer Science and Engineering and we want to thank Chalmers University of Technology for our years there. We would like to give a big thank you to our supervisor Sakib who has given us valuable insights about technology and also about life in general. Thanks should also be directed to Cybercom who has provided us with the resources that enabled the project. Thanks also for the competent guidance from our Cybercom supervisor Henrik. We would also like to thank our respective partners and families who supported us during this time.

Mattias Sikvall Källström, Eric Rylander, Gothenburg, 2020

Acronyms

AWT	Average waiting time
FHSS	Frequency hopping spread spectrum
ICWS	Intersection Collision Warning System
IEEE 802.11	Standards for WLAN communication
IEEE 802.11p / WAVE	Standard transmission protocol for vehicle communication
ITS	Intelligent Transportation System
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MAC	Media access control
mCore	Custom Arduino Uno board for mBot
RPi	Raspberry Pi
RSU	Road Side Unit
RTT	Round Trip Time
UML	Unified Modeling Language
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything

Contents

1

Introduction

The driving force behind developing vehicles that can communication with the outside world comes from a desire to make traveling safer, more efficient and a better experience for the driver [1]. This type of communication is called Vehicle to Everything (V2X) [2]. When vehicles communicate with infrastructure, the infrastructure is called a Road Side Unit (RSU). This project has been made on the behalf of the company Cybercom Group AB in Gothenburg. Cybercom has worked with V2X-technology for many years and wishes to demonstrate this upcoming technology in a model car track and this is something the purpose of this project aims fulfill.

1.1 Purpose

The purpose of this report is to model a vehicle environment where communication between vehicles and infrastructure can be tested. With studies from both literature and experiments this report wishes to answer the following questions.

- How can V2X technology be used to reduce traffic accidents in modeled intersections?
- What impact does the implementation of V2X have on the efficiency of intersections?
- How can V2X technology be used to reduce environmental impact?

1.2 Scope

The project includes a track for autonomous vehicles that contains a four-way intersection. Network communication shown in Figure 1.1 enable the intelligent infrastructure to transmit drive and stop signals to the vehicles and in turn result in an effective traffic flow. Each vehicle is equipped with a computer that handles the V2X communication. The intelligent infrastructure is modeled by another computer that acts as an RSU. Each vehicle has a field of view, that is implemented with an ultrasonic sensor.

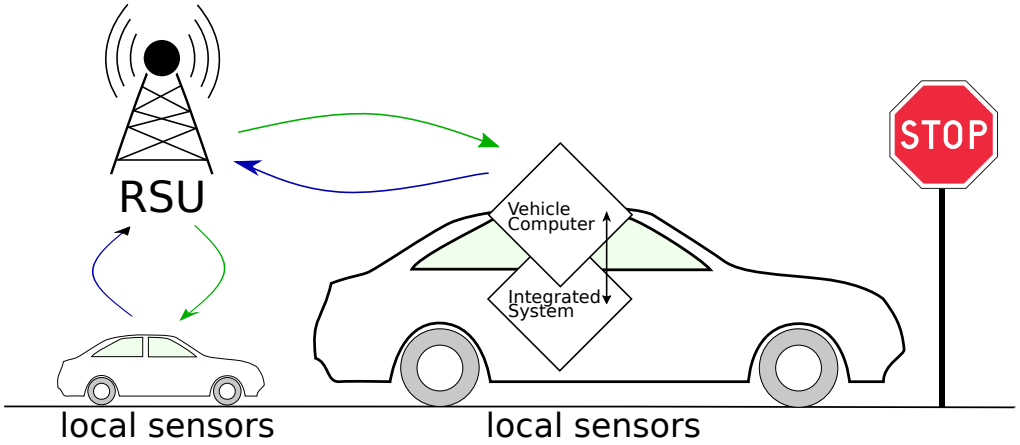

Figure 1.1: A flowchart of the system

Four applications have been developed for the project, one for, the embedded system, the vehicle computer, the RSU and the debugging software. Testes have been defined and run to test the efficiency of the intersection using different algorithms.

1.3 Delimitation

Tests in the project are not be performed on real vehicles, only on vehicles models. The vehicles can only drive straight through the intersection and neglects other crossing vehicles. Ultrasonic sensors are only used to give the vehicles vision in front of them and therefor the ultrasonic sensor can only eliminate crashes with other vehicles that are straight in front of them. No practical implementation of computer security regarding outside threats to the system, nor will it be discussed. Only preexisting hardware and drivers are used in the project.

2
Background and Theory

This chapter aims to provide an overview of the technologies and their usages in intelligent transportation. Furthermore, it describes the current research and development in this field as well as describing the technologies behind the projects system. Since 1997 the department of transportation in Sweden has aimed to totally eliminate deaths in traffic, a vision called "Nollvisionen" [3].

2.1 Vehicular Communication

The driving force behind developing vehicles that are able to communication with the outside world comes from a desire to make traveling safer, more efficient and a better experience for the driver [1]. Other cars may have information that is useful for a driver. Examples of this type information are how the conditions are further down the road, if there has been a collision or if there is an obstacle that is hidden on the road. This type of information could be vital to the driver and could be obtained by other vehicles, sensors positioned along the road or cloud connected services. Vehicle communication has been developed and researched for the last couple of decades, some recent research has begun to look in to the environmental aspects of the technology [4][5].

2.1.1 Vehicle to Everything and Vehicle to Vehicle

Vehicle to everything or V2X for short is a general term that describes the communication techniques that takes part between one vehicle and another entity of any kind [1]. These entities include but are not limited to other vehicles, pedestrians and infrastructure. To this day most development in vehicular communication has been towards technologies in the Vehicle to Vehicle (V2V) field, this is mainly due to this being the field where car manufacturers needs fewer standardization's and agreements between participants in order to actualize implementations of V2X technology. One example of V2V technology is a warning system that Volvo Cars are developing, it will be made available as a standard feature on all 2020 models [6]. The warning system comes into action when a car activates its hazard warning lights, a warning signal is then transmitted so that it can be received by all Volvo Cars connected to the cloud service. This is an example of a V2X/V2V application, safety among efficiency and driving experience are the main field of actions for V2X-communication [1].

2.1.2 Vehicle to Infrastructure and Road Side Units

Vehicle to Infrastructure or V2I for short, also known as Road to Vehicle (R2V) is the communication between vehicles and entities that are part of the infrastructure [2]. These entities are base stations or access point and are called a RSU:s. A RSU could provide vehicles with Internet access or inform vehicles about relevant traffic information. This information could be handled directly by the vehicle or the information could be passed on to the driver via a sound system or a monitor. In March 2020 Volkswagen claims to be the first car manufacturer to release a V2I capable car on the European market [7]. Volkswagen's new Golf can send and receive messages from other cars and infrastructure in a vicinity of 800 meters using WAVE technology, WAVE is a communication technology that will be explained in section 2.2.1. Other usages for RSU:s could be intelligent traffic lights, systems that provides warnings about hazardous road conditions and systems that assists the driver with finding parking spot [2].

2.1.3 Intersection Collision Warning System

A Collision Warning System (CWS) is a system that uses sensors to detect obstacles between vehicles and other entities. If such a collision risk is detected the system transmits some form of warning signal to the entities at risk. A CWS-system applied in an intersection is called an Intersection Collision Warning System (ICWS). Such a system could be created by positioning a radar in the intersection and allow vehicles to connect to it, creating a form of RSU. Test that have been made on ICWS-systems shows that collisions can be decreased by 18-26% [8].

Other studies on V2X communication in intersections with bicycle crossings showed the potential to reduce the number of accidents by 30-50% [9]. Another study has been performed by the United States department of transportation. This study shows that the use of V2I communication has the potential to reduce more than 250 000 accidents and 2 000 fatalities each year [4].

2.1.4 Environmental Impact of Vehicular Communication

Vehicle communication is not only beneficial for reducing accidents, research also shows that it could reduce pollution [4]. One example is a report from the United States Department of Transportation. This report shows that implementing traffic flow optimization using V2I has the potential reduce travel times by 6-27%. It also showed CO_2 emission and fuel consumption reductions by up to 11%. Another study on the subject shows that these types of systems could reduce CO_2 emissions by around 5% [5].

2.2 Communication technologies

Many different communication technologies can be used for V2X communication and they all come with different advantages and disadvantages. V2X creates a difficult environment for communication since transmissions must be sent between moving

destinations and sources as well through varying weather conditions. This generates the need for a robust and quick communication technology. This section present communication technologies that can be used for vehicle communication and the communication technology used in the projects system [2].

2.2.1 WAVE

WAVE is a modified version of the LAN protocol IEEE 802.11a, adapted for use in vehicular communication. The modifications were necessary in order to reduce the overhead time [2]. The modifications resulted in a new standard named IEEE 802.11p, but it is more commonly referred to as WAVE. There are two main differences between IEEE 802.11a and WAVE, these lie in the physical and Media Access Control (MAC) layers of the protocol. WAVE:s physical layer uses a reduced channel bandwidth of 10 Mhz instead of the generally used 20 Mhz in WAVE. The use of smaller bandwidth allows larger guard bands between adjacent channels. Thereby reducing the chance of interference. The MAC layer has a new mode of operation, the operation allows for transmission without association of a basic service set in emergency situations. This allows for quicker exchanges and with low overhead.

2.2.2 5G

5G is an upcoming technology that proposes great potential for V2X communication [10]. The technology is set to replace the current 4G network and will improve on it in almost every way. This means that 5G has the potential to be the network technology that can bind the Internet of Things (IoT) together. For vehicles and infrastructure this means that data concerning transport and traffic-related sensors can be shared in higher quantities and more frequently. China who is an early adopter of 5G plans on putting 5G connected sensors on 90% of their highways in the year 2020. Even though current cellular networks provides potential for many V2X functions, 5G will make it possible for vehicles to behave as smart clusters rather than individual units [11].

2.2.3 IPv4 and TCP

IPv4 and TCP are two protocols that can be used in combination to transmit data over the internet [12]. This is one of the most used communication techniques for Internet communication. IPv4 belongs to what is called the network layer. TCP is a layer on top of the network layer and is called the transport layer. Another example of techniques that uses the transport layer is the User Datagram Protocol (UDP).

IPv4

IPv4 is the fourth version of the Internet Protocol [12]. It is used to manage the transmission of packages between sources and destinations on the Internet. Every connected host is assigned an IP-address. This address identifies the destination a package would have to be assigned in order to reach the correct host. Apart from

the destinations IP-address, the IPv4 part of a package also contains the IP-address of the source, as well as other information that is required for well-functioning transmissions on the internet.

TCP

TCP is a transport communication protocol the builds on the IP layer [12]. When establishing a new TCP connection host A, sends a sync message to another host B. When host B receives the sync message it replies to A with a sync and acknowledge message. Now host A knows that it can both receive and send messages to host B and host A informs host B about this by responding with an acknowledge message. This process is known as a handshake and is described in figure 2.1.

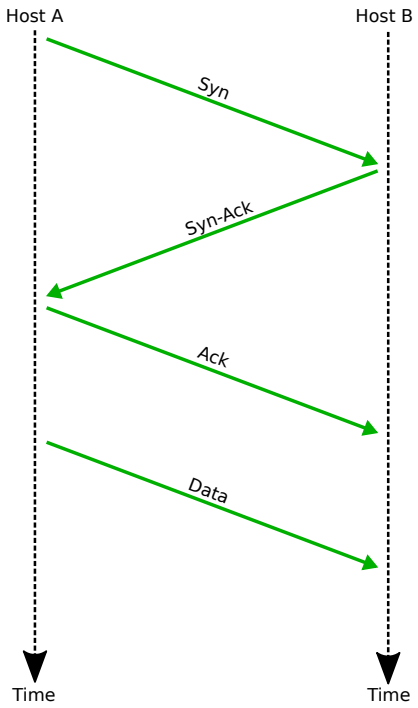

Figure 2.1: Example of handshaking in the TCP protocol

After a handshake has been performed data transmissions can be started between the two hosts. The receiving host will acknowledge every package it receives. If a sent package is not acknowledged it will be sent again in the next sending sequence. This guarantees that no data is lost. The TCP-connection needs to have predefined source and destination port. These ports are used to separate communication from different connections.

2.2.4 Bluetooth

Bluetooth is a wireless short way communication technology that allows transmission of data between devices [13]. It was conceived as an alternative to RS-232 data cables. Bluetooth uses the 2.4 GHz band, a band that is globally available for unlicensed low power use. In each network a Bluetooth device is connected to, the

device is either defined as master or slave. As master the Bluetooth devices can connect to multiple slave devices at the same time. Contrarily a slave device is only connected to the master. The architecture of Bluetooth defines two types of networks, Piconet and Scatternet. Piconet is defined as one master connected to up to seven slaves. A Scatternet is a combination of Piconets [13]. The nets are connected by a device that is slave and master at the same time. These slave-masters are often called bridge-slave.

The device connection process is divided in to three stages, inquiry, pairing and connected [14][15]. A Bluetooth device uses the inquiry request to discover other devices within range. If there are any nearby devices listening, the requesting devices will receive a response containing the other's address and name. If a connection is to be established, a paring process is initiated. During the paring process the devices exchanges secret keys that are used for future communication. After this stage the devices are said to be paired. Two paired devices can also be bonded, this means that the devices store each other's communication resources for future use. This means that they are able to auto-connect the next time they are within range of one each other. After the pairing stage the devices can connect to one another.

Bluetooth uses Frequency Hopping Spread Spectrum (FHSS) protocol to change operating frequencies during communication [13]. FHSS operates between seventy-nine predefined channels that exists in the frequency's spectrum of 2.402 GHz to 2.480 GHz. These operations occur 1600 times per second. The use of FHSS allows the connection between two devices to perform with less interference. Each channel can use 1 MHz of bandwidth.

The Bluetooth frame is divide in to three parts, access code, header and data, as shown in figure 2.2. The destination device uses the access code to identify the source of an incoming transmission. The header contains an identifier for the intended destination, an identifier for the type of data, a flow control bit, an acknowledgement bit, a sequence bit for detection of re-transmission of the data and a checksum for error detection. Furthermore, the header is repeated three times in the frame. The last part of the frame is the application data that is to be transmitted [13].

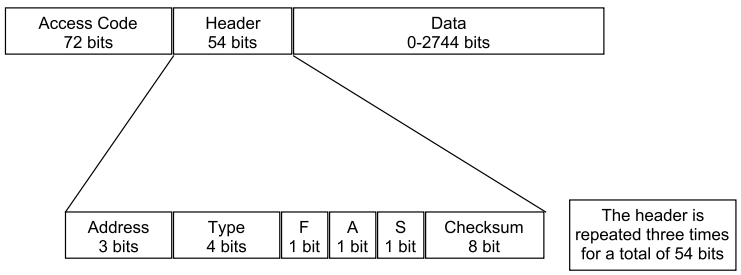

Figure 2.2: Bluetooth frame.

2.3 Hardware

The underlying sections describes the hardware used in the modeled environment. The modeled environment consists of vehicles and infrastructure. Raspberry Pis are

used for computational power and network communication on the different entities. The robots embedded systems are modeled with mBots.

2.3.1 Raspberry Pi

A Raspberry Pi (RPi) is a single-board computer built to be an educational tool for people to learn programming [16]. The first RPi launched in 2012 and contained a single-core CPU and 256 MB of RAM. The latest released RPi has a quad-core CPU and up to four gigabytes of RAM. RPi uses an ARM processor and the operating system Linux. The uses of RPi:s are many and since they are easily modified thanks to their easy to use General purpose input-output (GPIO) connectors. The GPIO connectors allows for several different accessories to be connected. LEDs, switches or analog sensors are some of the accessories that could be used. Which has made the RPi a popular computer to use in projects.

2.3.2 mBot

A mBot is a STEAM educational robot that is constructed of motors, sensors and a micro controller. [17]. The mBot used in the project consists of two line-follower sensors and an ultra-sonic sensor. It is also equipped with a battery pack, two motors, a Bluetooth module and a micro controller. The micro controller is a redesigned Arduino Uno, called mCore. It features onboard LEDs, IR receiver and RJ25 ports for expandability. An Arduino is an open-source micro controller that is made to be a tool for fast prototyping and a cheaper alternative. Arduino can be used with most operating systems and provides a simple and effective programming environment, the Arduino IDE. The programming language used for Arduino is based on AVR-C and is easily expandable with C++ libraries [18]. The Arduino IDE can be used to develop software for the mCore. The software requires both the Arduino's standard library as well as the mCore's standard library [19].

3
Methods

This chapter aims to motivate the choices made in the development process. Decision had to be made on what hardware, software, positioning, intersection handling and communication technologies that were to be used. Tests has been constructed and performed on the system.

3.1 Modeling

To achieve the purpose and scope of the project a modeled environment had to be constructed.

3.1.1 Vehicle

mBots was chosen to model the vehicles. The decision to use mBots was made based on them being available, simple to program and easy to expand upon with additional hardware[17]. This allowed the project to produce results in the early stages. With the help of sensors, it is possible for the mBot to model vehicles within an acceptable degree to reach the purpose of the project. Other vehicles were considered but due to them either being too complex or too few in quantity, they were not a viable option.

3.1.2 Line tracking and positioning

To model an intersection, a predefined track seen in Figure 3.1 and a system for positioning the vehicles is necessary. For positioning on the track, a method using two dashed lines were developed where the vehicle counts the lines it passes. Other systems considered for position were Ultra Wide Band and image recognition. However, these systems were deemed to be too complicated and time consuming for the projects scope. To keep the vehicles on the predefined track a solid lines was used, with a linefollower-sensor the solid line could be followed. The two lines were combined in to a two line system.

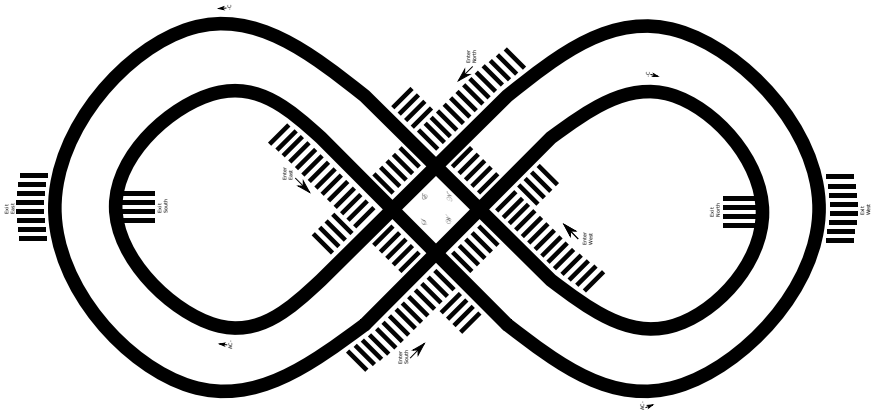

Figure 3.1: The track design with two two lanes and an intersection in the middle. The solid lines are used for keeping the vehicle on the track while the dashed lines are used for positioning.

Two line system

The two line system uses one solid line to keep the vehicle in the driving lane and one dashed line to calculate the position. The system counts the number of past dashed lines. Combining this information with the system-time makes it possible to calculate the vehicles position. This system used to design the first and following tracks.

Track design

The decision to make the track formed as an eight was based on the requirement to have an intersection that the vehicle could return to repeatedly. The track is designed in Inkscape, a vector graphics editor and printed on wide format printer. For a detailed view of the track see Appendix A.3.

During the development process different track designs were used. The first design was a single lane, one way track see Appendix A.1, this track laid the basis for the coming track designs. The single lane track didn't realize our scope of having traffic traveling in both directions. This led to the second design found in Appendix A.2. This second design was the first track to support cars in different directions. The second track designed had problems with vehicles losing their positions. These flaws were later fixed in the third and final design.

3.1.3 Vehicle Computer

A Raspberry Pi 3B+ was chosen for use as vehicle computers for the mBots. Due to its small size and cost, while still having plenty of computing power for the projects purpose. The RPi zero was a strong candidate for usage as vehicle computer, but resource limitations due to COVID-19 made it so that only RPi 3B+ were available. No other microcomputer than RPi models was considered since the authors were accustomed to working with RPi:s.

3.1.4 Vehicle state machine

In order to make the vehicles perform in a desired manner a state machine was developed. The different states represent the vehicles position relative to the line section it is currently passing. Two different state changes were constructed. One in the mBot and one in the RPi. The one used in the mBots software contained all possible states for the line-follower sensor that was used to count the dashed lines. It was used to differentiate when the sensor passed a line and if the line should be counted. These states can be seen in Figure 3.2. The HALFOUTSIDE and HALFINSIDE states was required for the one-line system to work and was kept in the two-line system since that the precision was seen as an asset. The two-line system would work just as well running only with the INSIDE and OUTSIDE states.

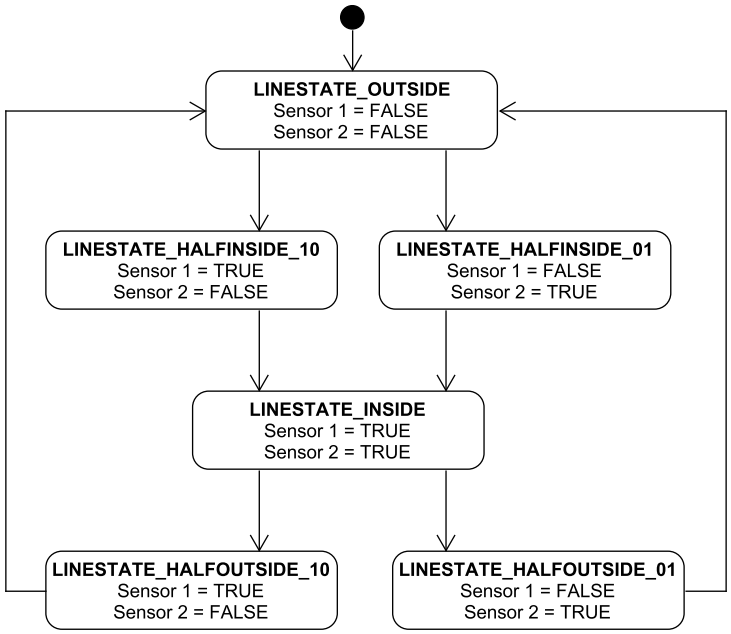

Figure 3.2: The states controlling dashed line counter.

The other state machine was implemented in the RPi:s software, these states was also used in the positioning of the mBot. These separated the track into sections and changed according to the section that the mBot were in. These can be seen in Figure 3.3. These states were divided up in two parts, enter and exit after the four cardinal directions used to divide the track. These were named and divided after the cardinal directions to make the positioning system accurate and easy to understand and debug.

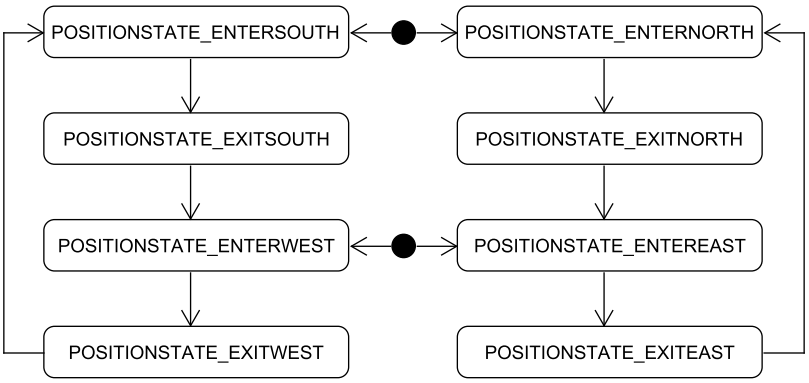

Figure 3.3: The states used for positioning.

3.1.5 Road Side Unit

To model the Road Side Unit an application that could calculate and predict if two or more vehicles were about to collide was developed. This program was made using a prediction algorithm that could define if a vehicle is allowed to drive through the intersection or not. For this algorithm to work a communication with the different vehicles were needed, as well as message protocols. Two criteria was constructed for the algorithm, no collisions were allowed in the intersection and it needed to be scalable.

Network Communication

A decision was made to let all communication with RSU be performed over WiFi with the TCP protocol. This goes against much of the conventional research that exists about RSU:s, which often uses WAVE or other specially developed protocols for vehicular communication. Communications with an RSU and a vehicle some package loss must be accepted. However, in the modeled environment that has been constructed, WiFi and the TCP protocol is used since WiFi is a cheap and accessible technology. While the technology, such as WAVE needs special hardware that was unavailable to the project.

3.2 Debugging software

A debugging software was developed to facilitate the developing during the project. It was used to test the Bluetooth transmissions between the mBot and the program, which helped to test and define protocols of transmission to save valuable time during run time. It was also used to run the system with a single computer instead of a different RPi for each mBot. This allowed for faster debugging of the code and allowed the project to proceed quicker.

3.3 Tests

To answer the project questions, three different tests were constructed. Each test ran for ten minutes and was repeated three times. Ten minutes was deemed as an acceptable time frame for the test due project limitations and the fact that the driving pattern of the vehicles became repeatable. The tests were done with one to three vehicles to ascertain what effect the amount of vehicles on the track had on the result. The tests seen in Table 3.1 where performed once for each algorithm N. Where CW and ACW shows how many vehicles should be traveling in each direction, clockwise and anticlockwise.

Algorithm	**Test #**	**# Vehicles**	**CW**	**ACW**	**# Runs**	**Period [min]**
Algorithm N	1	1	1	0	3	10
Algorithm N	2	2	1	1	3	10
Algorithm N	3	3	2	1	3	10

Table 3.1: The test settings

Two measurements were created in order to be able to assess the result, the Average Waiting Time (AWT) and throughput. When calculating the AWT time frames are used, they consist of the time it takes from when a mBot gets a stop signal till it gets a drive signal from the RSU. The average waiting time is calculated by taking the sum of all the time frames and divide it by the number of cars on the track. This is shown in Equation 3.1 where k is the number of time frames, S is the stop time, D is the drive time and C is the number of cars on the track.

$$AWT = \frac{\sum_{n}^{k}(D_n - S_n)}{C} \tag{3.1}$$

The throughput is the number of vehicles that drives through the intersection each minute. The Equation 3.2 shown how the throughput is calculated.

$$Throughput = \frac{vehicles\ passing\ through\ intersection}{test\ duration} \tag{3.2}$$

3.4 Languages

Languages that are both understood by humans and computers are vital tools in all software development projects. In the following sections a motivation of the chosen languages C, Java and JSON are given. These languages were all used to develop the system that this report aims to describe.

3.4.1 Programming Languages

Programming languages was chosen based on the requirements of the hardware and the experience of the developers. C was chosen for writing the embedded system application and Java for writing the applications for the vehicle computer and RSU.

C

C is high-level programming language that was created in the early 1970s [20]. The main features of C are that it provides low-level access to memory and the usage of simple syntax. C were used to develop the software running on the mCore, as it only supports C or Scratch, a block-based programming language. C were chosen based on that it being an industrial standard for embedded systems which scratch is not.

Java

Java is a general purpose programming language that was first released in 1994 [21]. The reasons we chose java as our developing language is that Java is an object-oriented language and it manages all memory handling, eliminating the source for many bugs and performance issues. It also has the benefits of the JVM that allows compiled code to execute on multiple platforms. Additional to this benefit the JVM also works as an extra layer of security. Java is also designed with concurrent programming as one of the main design goals. The developers of the applications were also used to working with Java. The problems with Java is that its proven to function poorly when used in real time applications resulting in poor performance and communication delays. However the benefits of using Java was considered to out weight the negative.

3.4.2 JavaScript Object Notation

JavaScript Object Notation (JSON) is an independent language based on a subset of JavaScript. It is a lightweight data-interchange format designed for human readability and writability. While being fast for computers to parse and understand [22]. These are the reasons why JSON was used as a data-interchange language for the system.

4
Architecture

The basic setup consists of mBots, a track and a RSU. The RSU communicates via WiFi and TCP with the RPi, which in turn communicates via Bluetooth with the Arduino. The Arduino is connected to sensors and motors that allows it to drive around the track and calculate its position relative to the track. The RPi interpreters the data from the Arduino and updates the RSU with the mBot's position. The RSU is used to avoid collisions in the tracks intersection and does this by performing a collision avoidance algorithm every time a mBot updates it with its position value. In order to test the system thoroughly a debugging software has been constructed. All data being sent between the different modules in the system uses messages written in JSON.

4.1 RSU software

The purpose of the RSU is to make sure that no collisions take place in the tracks intersection. It does this by communicating over LAN with every mBot on the track and performing a collision avoidance algorithm that determines if a mBot is allowed to drive or if it has to stop. The architecture of the software is described in Figure 4.1.

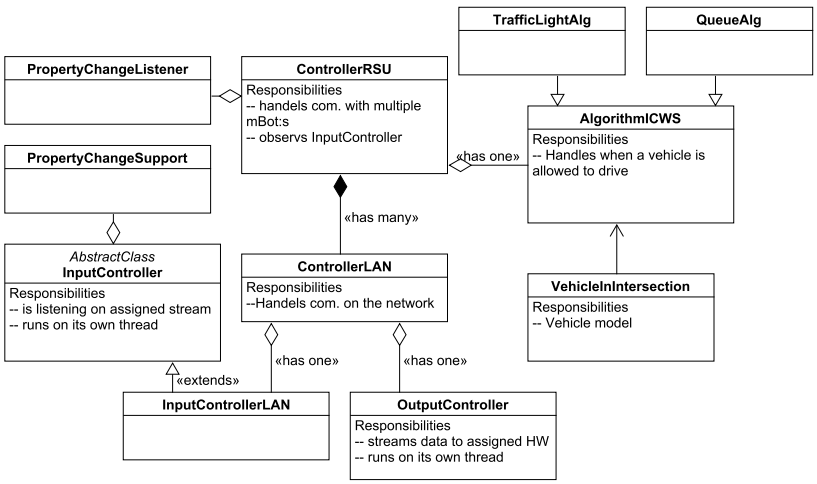

Figure 4.1: UML class diagram of the RSU software

The software consist of one main controller that handles the communication with the mBots and performs the algorithm for collision avoidance. The RSU works as a

server and continuously listens for new connections. Each server connection is using the TCP protocol and runs on its own thread in the form of the ControllerLAN class, this class makes it possible for the ControllerRSU to observe incoming messages and send messages to the mBots.

4.1.1 Collision avoidance algorithms

Two different collision avoidance algorithms were developed to be tested with the RSU. The first was an algorithm that mimic a intersection with traffic lights. This was especially used during the development and debugging of the network communication and positioning. Another algorithm using queues was also developed. The algorithms are executed on a set time interval and each execution the vehicle is informed of the result.

The traffic light has three states closed, north-south open and east-west open. The states switches from closed to north-south open to closed to east-west open and then repeats the process. Different times can be set for how long the intersection is open and closed. The traffic light Algorithm 1 uses the traffic light state to determine if an incoming vehicle is allowed to continue or not.

if *(East-West green AND Entering East-West) OR (South-North green AND Entering South-North)* **then**
| Continue driving;
else
| Stop driving;
end

Algorithm 1: Traffic light algorithm in pseudocode.

The queuing algorithm uses two queues, one for south-north and one for east-west. A mBot continuously sends updates on it whereabouts to the RSU. The algorithm uses this information to check what direction the mBot is traveling in and adds it to the right queue. It then checks if the other queue in the crossing direction is empty, if it is then the RSU sends driving messages to the mBot. If the other queue is not empty the RSU sends stop messages and the mBot awaits a drive message from the RSU. When a mBot leaves the intersection, the algorithm removes it from the queue and checks if the queue is empty. This algorithm is described in pseudo code in Algorithm 2.

```
switch Vehicle direction do
    case Enter from south/north do
        Add vehicle to south-north queue;
        if east-west queue is empty then
            Open south-north direction and close other;
        end
    end
    case Enter from east/west do
        Add vehicle to east-west queue;
        if south-north queue is empty then
            Open east-west direction and close other;
        end
    end
    otherwise do
        Make sure that vehicle exist in no queue;
    end
end
if vehicle is not in closed direction then
    Continue driving;
else
    Stop driving;
end
```

Algorithm 2: Queuing Algorithm in pseudocode.

4.2 mBot RPi software

The RPi software works in close collaboration with the Arduino software. It complement the Arduino with network communication, system timing and extra computing power. An overview of the RPi software is demonstrated in Figure 4.2.

The network card makes it possible to establish a TCP connection to the RSU and update the RSU with the current state of the mBot, this is performed each time a position change is notified by the Arduino. In this modeled environment, the mBot has a continuously open connection stream between itself and the RSU. The RPi software communicates with the Arduino via a Bluetooth connection. The Arduino updates the RPi software with information about its state, position, base speed, etc. and the RPi software stores this information in a model class called MBotModel. The position value in combination with system time makes it possible to calculate the vehicles position on the track. It does this by calculation how many tracking lines the mBot has passed within a set amount of time. The calculated position is then used to determine which of the different cardinal direction states the mBot is located in. In turn the RPi software updates the Arduino with stop and drive instructions it gets from the RSU's collision avoidance algorithm, stop messages can also be transmitted if objects are in front of the mBot.

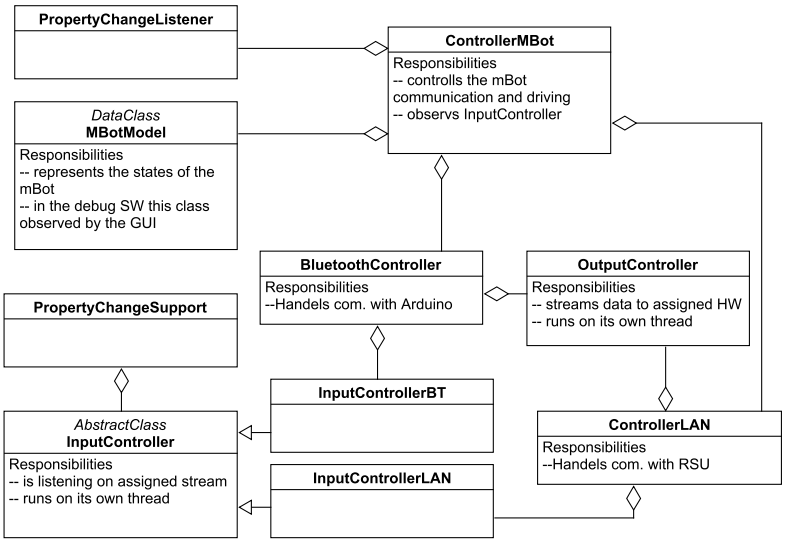

Figure 4.2: UML class diagram of the mBot RPi software

MBotModel represents the real time state of a mBot. Different states are represented by enums. A MBotModel is update via it's controller that gets Bluetooth updates from the mBot and sets the MBotModels values correspondingly. The two enums are PositionState and LineState, PositionState represents the mBot state relative to track while PositionState represents the mBot state relative to the position tracking lines on the track.

4.3 mBot Arduino software

The software that is controlling the Arduino (mCore) is written in C and its task is to run four different functions after each other in repeat. The Arduino program is split in two parts, a setup and a loop. The setup is the boot up sequence, it sets the communication and resets the LEDs. While the loop run continuously until the mBot is turned off, this is where the four functions are located.

The purpose of the mCore software is to control the mBot and handle the communication between the mBot and the RPi. These responsibilities is split up in four different functions: bluetoothReceiver, lineFollower, lineCounter and ultraSonic. The communication updates to the RPi is transmitted every 150 system-loops, with a complete update message containing all relevant information about the mBot. An activity diagram of the Ardunio software can be seen in Figure 4.3.

The bluetoothReceiver listens for messages from the RPi. The receiving process is separated into two functions, reading messages and finished reading messages. It reads messages until it receives a end of transmission character "\n". Then the message is handed over to a JSON parser that extracts all the information and the message is interpreted.

The lineFollower is responsible to make sure that the mBot stays on the line and follow the predefined path. Reading data from one of the linefollow-sensors allows the mBot to adjust after the line. The data from the linefollow-sensor has

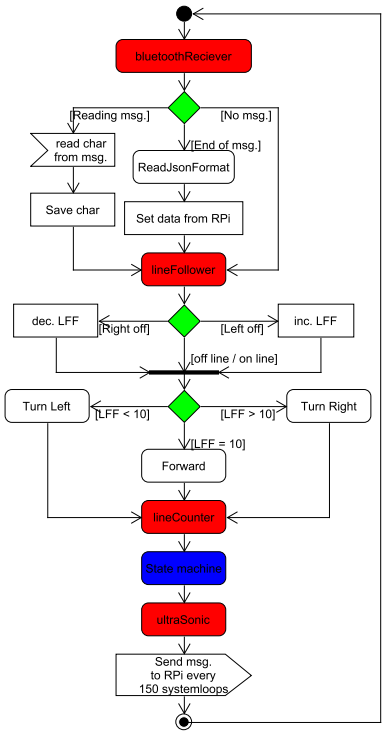

Figure 4.3: UML activity diagram of the mBot Arduino software

four states, on the line, off the line, right off and left off. By checking what state is active, the function sets the motors to the right effects until the state on the line is active.

The lineCounter uses the data from the other linefollow-sensor to count how many dashed lines it has passed. The state machine seen in Figure 3.2 can determine when and how the mBot passes over the dashed lines. A total of six different states is used to help the program determine when it has passed a dashed line. The lineCounter function decreases the communication intervals by increasing the system-loop counter, each time it determines a increase in counted lines. Thereby reducing the time between update messages to the RPi, this is done to assure that the communication happens at a higher interval when the mBot is passing dashed lines.

The ultraSonic function reads the data from the ultrasonic sensor and saves the value. The value is then transmitted to the RPi along with all other data in the update message.

4.4 Debugging software

The debugging software works by providing an actor with the opportunity to observe and modify the communication between RSUs and mBots. The application is developed in Java and meant to be run in Windows. The software structure is described in Figure 4.4.

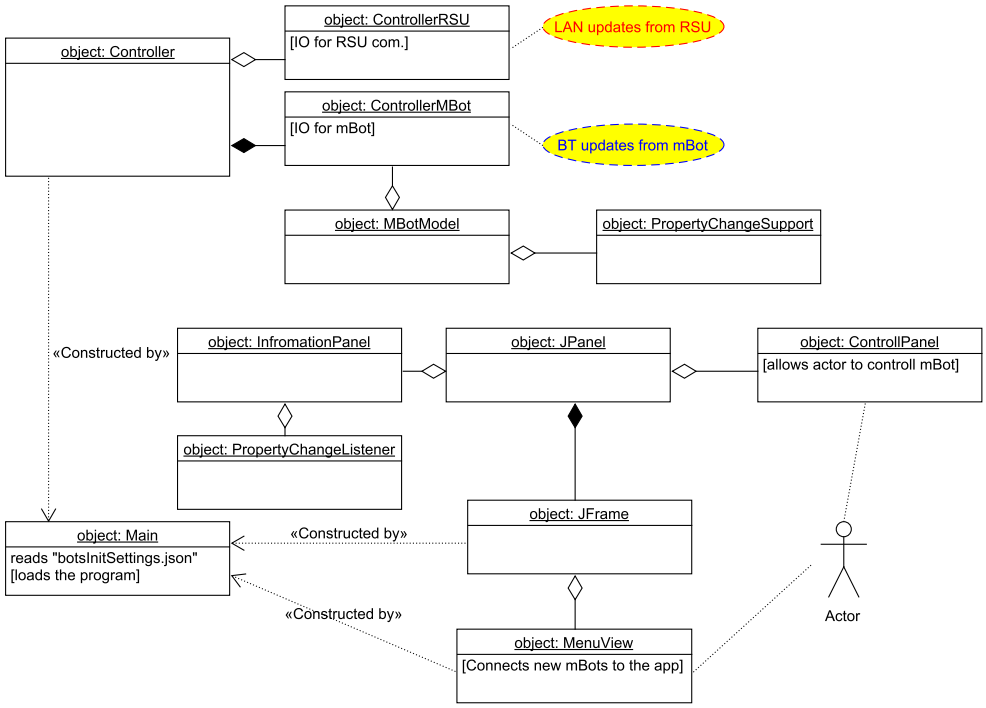

Figure 4.4: UML class diagram of the debugging software

A more detail overview of the classes ControllerRSU and ControllerMBot are provided in Figure 4.1 and respective Figure 4.2. The software follows the MVC model that has been extended to deal with Bluetooth and LAN communication. On initialization the software reads a setting file with information about the mBots that are to be used. From this information instances of ControllerMBot are constructed by the controller. A RSU server is run locally on the system and each instance of ControllerMBot communicates using local host.

The controller consists of one main controller that in turn handles two different types of controllers, these are ControllerRSU and ControllerMBot. Both of these sub-controllers are identical to the software that are running on the mBots and the RSU. The controller has support for one RSU and multiple mBots. On creation of a mBot it is connected to the RSU via localhost and the predefined Bluetooth address. Both controllerRSU and ControllerMBot are identical to the classes used in the RSU software and the RPi software. It's possible for the controller to handle multiple mBots, but is only able to handle one RSU. The controller handles input from the user via the GUI, LAN messages from the RSU and Bluetooth messages from the mBot. The controllerMBot uses a extended version of the MBotModel, that has a propertyChangeSupport that the GUI can listen to.

The GUI was constructed using standard Java Swing components. In Figure 4.5 a screenshot of the GUI running is displayed. The GUI shows information about the mBots, the intersection queues and the the traffic light states. It also allows the actor to send messages directly via Bluetooth to the mBots. An actor can connect to a mBot with the debugging software by pressing the connect button. Using the GUI the actor can set mBot standard speed, ping the Bluetooth connection and

set if the mBot is driving in clockwise or anticlockwise direction. The information displayed in the GUI is updated when changes are fired by the MBotModel.

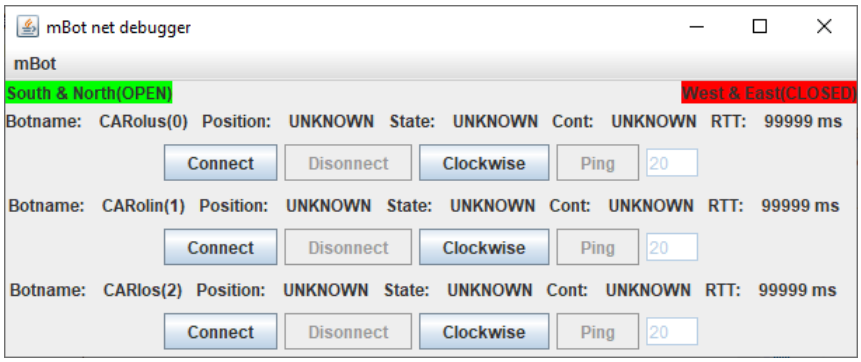

Figure 4.5: The GUI of the debugging software

4.5 JSON messages

All communication over LAN and Bluetooth is sent via messages written in JSON. For each message that is to be sent a JSON object is created that consists of a varying number of data types. A receiver of a messages checks each variable name individually and performs corresponding action with the given value. If a message isn't recognized by the receiver no action is taken. The communication uses only simple JSON objects that consist of Numbers, Strings or Booleans. An object or an array in the base JSON object would simply be ignored. In the Ardunio software no objects are used, instead the JSON messages are constructed using strings.

4.6 Track

The track proportions are 105.0 times 194.0 centimeters and it is formed after two intersecting eights, where a four-way intersection is created in the center of the eights. The dashed lines are six by one centimeter with one centimeter white space separators, see Figure 3.1. These dashed line are strategical places around the track in order to position the mBots. The solid path line has a width of three centimeters.

The section of dashed lines in the top and bottom of the track is there to help the mBot position itself after it has started. The sections of dashed lines in the middle is used by the mBot to calculate its position inside the intersection. An algorithm counts continuously lines during a specific time, this allows the mBot to precisely know where on the track it is located. This algorithm is dependant on the mBot knowing what direction its driving in on the track, clockwise or anticlockwise. A larger view of the track can be found in Appendix A.3.

Positioning the cars on the track was made by dividing the track in to different sections. The entrances to the intersection are named after the four cardinal directions. This created two travel directions through the intersection, south-north and east-west. Exit sections from the intersection entrances was created at the top and the bottom of the track. A vehicles position state is dependant on it traveling in

a clockwise or anticlockwise direction, see Figure 3.3. A vehicles traveling on the clockwise track would go from exit north, to enter west, to exit west, to enter north, and then return to exit north. A vehicles traveling on the anticlockwise track would go from exit east, to enter south, to exit south, to enter east, and then return to exit east.

5

Results

To answer the question regarding what impact the implementation of ICWS have on the efficiency of intersections, three questions have been defined to explain how the efficiency of an intersection is to be measured. These are:

- How long did the vehicle wait in front of the intersection described in percentage of total driving time?
- What throughput can the algorithm give described in vehicles per minute passing through the intersection?
- How does different number of vehicles on the track affects the throughput and percentage?

5.1 Tests

The tests were performed with the two collision avoidance algorithms described in the Architecture chapter. Three different mBots were used in a total of eighteen tests. The three tests shown in Table 3.1 was run for ten minutes and repeated three times. The first test with one mBot, the second with two mBots and the final with three mBots. The tests were run once for each algorithm. The details regarding the collected data can be found in Figure B.1 in Appendix 2.

5.1.1 Traffic Light Algorithm

The tests start with the all directions closed and then switches to open the south-north direction. The south-north direction is then open for seven seconds and after that all direction are closed for three and a half second. Thereupon the east to west direction is open for seven seconds time, thereafter it all directions closes three and half second. Then the process repeats.

Table 5.1 shows that AWT did not change much between the tests except for a slight decrease in AWT in the test with three vehicles. After the second test with three vehicles, vehicle two had not registered any waiting time and this could be an anomaly. The tests also show that by increasing the number of vehicles on the track the throughput decreases.

Algorithm	# Vehicles	Avg. waiting time [%]	Throughput [vehicle/min]
TrafficLight	1	21	3.17
TrafficLight	2	21	3.03
TrafficLight	3	20	2.81

Table 5.1: The test result of the Traffic Light algorithm

5.1.2 Queuing Algorithm

The queuing algorithm directly gives priority to a vehicle entering the intersection when no other vehicle is present. It also gives priority if another vehicle already is passing through the desired direction. The tests with Queuing Algorithm seen in Table 5.2, shows that with an increases number of vehicle that AWT increases while the throughput decreases. When only a single vehicle drove on the track no AWT-time was measured. These test were run with no anomalies detected.

Algorithm	# Vehicles	Avg. waiting time [%]	Throughput [vehicle/min]
Queue	1	0	4.07
Queue	2	1	3.90
Queue	3	4	3.64

Table 5.2: The test result of the Queuing algorithm

6
Discussion

This chapter discusses real world applications of V2X and its impact on society. It also discusses the result of the tests that was performed and the development of the software. The chapter ends with a conclusion of the project. The following sections provides a discussion on how V2X communication impacts traffic if it is implemented in a real scenario. Intelligent transportation systems are discussed from the aspects of driving experience, efficiency, safety, environment and ethics.

6.1 Test anomalies

Only one anomaly was detected in our test results and this anomaly is shown in Table 6.1. It shows the results of vehicle 2 in the tests with the traffic light algorithm and three vehicles. As described in the result, the second test result shows a waiting time of zero, a result that stand out compared to the other two tests. We recognize this is due to the vehicles moving in different speeds and one vehicle getting stuck behind the other. This vehicle congestion result in that the vehicle behind does not register a waiting time, due to it stopping for the vehicle in front of it and not the ICWS. Since this only happened once and the result is still relevant and its valid enough for conclusions to be based on it.

Algorithm	**Test nr**	**Wait time[s]**	**Intersections pass**
TrafficLight	1	116.07	36.00
TrafficLight	2	0.00	28.00
TrafficLight	3	183.29	28.00

Table 6.1: Vehicle twos result from the tests with three vehicles, using the Traffic Light algorithm.

6.2 Traffic efficiency

The result in Table 5.1 and Table 5.2 shows that V2I technology can be applied in intersections and improve the traffic efficiency in a small scale modeled environment. When the queuing algorithm is effective and what would happen if we continued to add vehicles to the track is to be discussed.

Queue algorithms could be used to speed up an intersection that uses traffic lights, especially in low traffic areas. As the results show queuing algorithms could reduce the waiting time by up to 16% and increase throughput by up to 30%. This is

something that an autonomous vehicle could handle, however it is worth questioning how a human driver would handle the quick signal changes required in the algorithm. It is also worth to mention that different communication delays may have a impact on the efficiency of the algorithm. Other research show that V2X-communication has the potential to reduce travel times by 6-27% [4].

The result shows that both the values of AWT and throughput in the two algorithms approaches each other as the number of vehicles on the track increases. It is reasonable to assume the as more vehicles were added to the track, the results from the different algorithms would keep getting closer to one another and finally even out to the same values. This is because it should be possible for both algorithms reach close to the physical max throughput of the intersection. The Traffic Light algorithm would most likely still perform worse however, since it has a state where all directions are closed. The only hard delay that doesn't allow the queuing algorithm to reach the physical max throughput is the time it takes for the RSU to inform a vehicle that the intersections is free.

6.3 Driving experience

When implementing an ICWS-system it's important to take the people traveling in the vehicle in account. Whether the system is implemented for a autonomous vehicle or giving directions to the driver, many aspects of driving experience needs to be considered. For example, the vehicles need to ease in when it brakes, and it also needs to provide the driver with the feeling that the system is a helpful tool. No such driving experience tests were performed during our testing process, mostly due to the simplicity of the modeled environment. However, adding a simulated breaking distance to the vehicles could have helped with providing a more meaningful result considering driving experience.

6.4 Traffic safety

The goal of "Nollvisionen" is that none will be seriously harmed or killed in traffic in Sweden [3]. For this goal to be reached we think that innovative V2X-technological solutions are the future for traffic safety. For example, other research that implements an ICWS audio warning system for drivers results in a decrease of collisions by 18-26% [8]. While another test shows a potential of reducing accidents by 30-50% in intersections with bicycle crossings [9]. Our project has shown the potential of limiting accidents and increase the safety in intersections by adding an extra layer of security that could stop a vehicle that is inbound for a collision. This could help unobservant drivers and facilitate the development of security functions in autonomous vehicle. Security features from V2X-technology would not be limited to ICWS-systems but could also create safer pedestrians or bicycles crossings and road safety in general.

6.5 Environmental aspects

ICWS-systems and V2X in general show potential for lessening the pollution from vehicles. Implementing V2X algorithms will increase traffic flow in intersections, resulting in less stops and shorter waiting times. This has the potential to reduce emissions as a vehicle would have to spend less time idling and reduce the number of stop-start sequences that has heavy impact on fuel economy. This is confirmed by our result that show decrease in AWT. This is further backed with the earlier mentioned research that show how V2X communication could reduce CO_2 emissions and fuel consumption with up to 11% [4]. It could have been a good idea to measure the number of stops in our tests and in that way get another measurement on environmental effect. Since these types of technologies aims to increase driving experience and efficiencies it is natural to assume that people would use these types of services more often. Even thought these vehicles would be more environmental than today's vehicles an increased usage may have a grander total impact on the destruction of the environment.

6.6 Ethics

Our tests never considered how a collision would be handled, since it was assumed that all collisions would be avoided. Therefore, no ethical choices in course of a collision needed to be considered. However, ethic is an important subject in traffic and if testing were to be performed on a larger scale this is something that should be taken in consideration.

Research points out that not all crashes could be avoided with an ICWS-system, an ethical choice that the ICWS system could face is what instructions it should give when a crash is inevitable. For example, it could have to decide what or who a vehicle should crash in to. One viewpoint is that if the vehicle is non autonomous the driver should handle all ethical choices he or she may face in traffic. V2X could be an important puzzle piece for developing autonomous vehicles and this could in turn lead to increased unemployment among professional drivers.

Ethically we choose a utilitarian approach to evaluate the ethics behind V2X implementation. It is our firm believe that the benefits from having safer traffic and lower environmental impact out weight the disadvantages of increased unemployment. History has plenty of examples of rises in unemployment due to the implementation of new technology, still no one seems to want to go back.

6.7 Critical Discussion

If we were to redo the project with all the knowledge, we have today there are a few things that we would have done differently.

- Even though we achieved success with the mBot, other technologies for modeling vehicles exists. Developing the mBots took a lot of time and it is reasonable

to assume that the same result could have been achieved with a toy railed car track.
- We would realize the importance of having a modular message structure earlier and thereby reducing allot of the problems we had in the first part of the project. It would also have been helpful to do better research on Bluetooth and realize its limits and potential.
- The C software could have been given more planning and structure. It only uses simple techniques and more advanced programming methods could have been used to make the code more effective and readable.
- The Java applications requires more computing power than what seems reasonable. Testing the different modules of the Java program better would most likely have resulted in a more resource effective program.

Viewing our tests and system from a critical standpoint there are a few issues that may have affected the results. These are important to take in consideration when considering at the results.

- The modeled vehicles have a probability of syncing with the traffic lights, meaning that when they approaches the traffic light it transmits the same signal as last time it approached the intersection.
- The queuing algorithm could result in a race condition, were one queue gets blocked indefinitely. This could happen with the track design used in these test since the vehicles never returns directly to the same queue. However with a different track design this would become a problem.
- The modeled vehicles have different max speeds that depended on their motors, this was not adjusted for. This resulted in that if two vehicles are in the same direction and one is faster than the other the faster one gets stuck behind the slower one. This could result in that the calculated intersection stop time is low even though it waited in front of the intersection in the same manner as the vehicle in front of it.

6.8 Future development

If we had more time to continue developing our modeled environment these are the points that we see as the next step in developing the system:

- The light switching time used on the traffic light algorithm was arbitrarily chosen and was only adjusted slightly during the development process. A more scientific approach for deciding the time could have been more advantages, for example machine learning could have been used in order to find a light switch time for optimal vehicle flow.
- Implement a adjustment for vehicle speed, that would make each vehicle adapt motor power to match a set speed as this might have had an impact on our result.
- The vehicle computer was never mounted to the mBot during the tests, a future implementation could be to construct a mounting section for the RPi

and power supply on the bot itself. This would provide a more realistic test rig and allow testing the RSU communication with moving targets.
- Changing the LAN communication to use UDP instead of TCP, this would probably give more realistic test results as the communication in vehicular environments does not use TCP. The next step after UDP would be to test it with WAVE, as it is the standard set for vehicular communication and would provide a better picture of how the system would perform in realistic environments.

6.9 Conclusion

Implementing V2X technology is something that could improve traffic in general and according to us this is the future of infrastructure. We firmly believe that a distinct improvement of safety and efficiency in traffic would be associated with the implementation of V2X. The environment would also benefit of V2X.

Our tests show that the traffic light algorithm may be the slowest, but we think that it is the most reliable. If the project where to be scaled up and more complexity added we are unsure if the queuing algorithm would still work, but we are feeling confident that the traffic light algorithm would do its job. This is to say that if breaking distance, reaction time and weather conditions where to be factored in an optimized version of the traffic light algorithm would be our recommended implementation. Since this study almost exclusively approached the software and communication aspects of ICWS-systems further research on how intersections are handled today is required, this include what requirements that exists and what that means for effective algorithms.

Bibliography

[1] G. Dimitrakopoulos and G. Bravos, *Current Technologies in Vehicular Communication.* Springer, 2017.

[2] S. F. Hasan, N. Siddique, and S. Chakraborty, *Intelligent transportation systems: 802.11-based Vehicular Communications.* Springer, 2017.

[3] Trafikverket, "Det här är nollvisionen," Jan 2020, accessed 23 May. 2017. [Online]. Available: https://www.trafikverket.se/resa-och-trafik/Trafiksakerhet/det-har-ar-nollvisionen/

[4] J. Chang, G. Hatcher, D. Hicks, J. Schneeberger, B. Staples, S. Sundarajan, M. Vasudevan, P. Wang, K. Wunderlich *et al.*, "Estimated benefits of connected vehicle applications: dynamic mobility applications, aeris, v2i safety, and road weather management applications." United States. Department of Transportation. Intelligent Transportation ..., Tech. Rep., 2015, accessed 19 April. 2020. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/3569

[5] F. Outay, F. Kamoun, F. Kaisser, D. Alterri, and A. Yasar, "V2v and v2i communications for traffic safety and co2 emission reduction: A performance evaluation," *Procedia Computer Science*, vol. 151, pp. 353–360, 2019, accessed 19 April. 2020.

[6] A. Frost, "Volvo introduces v2v warning systems on new models across europe," Apr 2019, accessed 17 April. 2020. [Online]. Available: https://www.traffictechnologytoday.com/news/safety/volvo-introduces-v2v-warning-systems-on-new-models-across-europe.html

[7] Volkswagen, "Technical milestone in road safety: experts praise volkswagen's car2x technology," Mar 2020, accessed 17 April. 2020. [Online]. Available: https://www.volkswagen-newsroom.com/en/press-releases/technical-milestone-in-road-safety-experts-praise-volkswagens-car2x-technology-5914

[8] S.-H. Chang, C.-Y. Lin, C.-C. Hsu, C.-P. Fung, and J.-R. Hwang, "The effect of a collision warning system on the driving performance of young drivers at intersections," *Transportation research part F: traffic psychology and behaviour*, vol. 12, no. 5, pp. 371–380, 2009, accessed 19 April. 2020.

[9] C. Sommer and F. Dressler, *Vehicular networking.* Cambridge University Press, 2015.

[10] N. Sequeira, "What 5g means for the future of internet of things," Jan 2019, accessed 5 May. 2020. [Online]. Available: https://www.5gtechnologyworld.com/what-5g-means-for-the-future-of-internet-of-things/

[11] T. L. Ericsson, "5g and v2x ecosystem," Sep 2019, accessed 5 May. 2020. [Online]. Available: https://www.ericsson.com/en/news/2019/9/5g-and-v2x

[12] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Addison Wesley, 2005.
[13] U. Dalal, *Wireless Communication and Networks*. Oxford University Press, Inc., 2015.
[14] D. Thakur, "Bluetooth," 2017, accessed 4 May. 2020. [Online]. Available: https://piembsystech.com/bluetooth/
[15] Sony, "What is bluetooth pairing?" 2019, accessed 25 May. 2020. [Online]. Available: https://www.sony.co.uk/electronics/support/wireless-headphones-bluetooth-headphones/wf-1000xm3/articles/00196698
[16] R. Pi, "Teach, learn, and make with raspberry pi – raspberry pi," 2020, accessed 15 May. 2020. [Online]. Available: https://www.raspberrypi.org/
[17] Makeblock, "mbot," 2020, accessed 5 May. 2020. [Online]. Available: https://www.makeblock.com/mbot
[18] Arduino, "Arduino - introduction," 2020, accessed 4 May. 2020. [Online]. Available: https://www.arduino.cc/en/guide/introduction
[19] Makeblock, "mcore - steam kid robotics projects | makeblock," 2016, accessed 4 May. 2020. [Online]. Available: https://www.makeblock.com/project/mcore
[20] J. Skansholm, *C från början*. Studentlitteratur AB, 2016.
[21] A. Binstock, "Java's 20 years of innovation," 2015, accessed 5 May. 2020. [Online]. Available: https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation/#2607927e11d7
[22] D. Crockford, *How JavaScript Works*. Virgule-Solidus, 2018, accessed 5 May. 2020.

A

Appendix 1

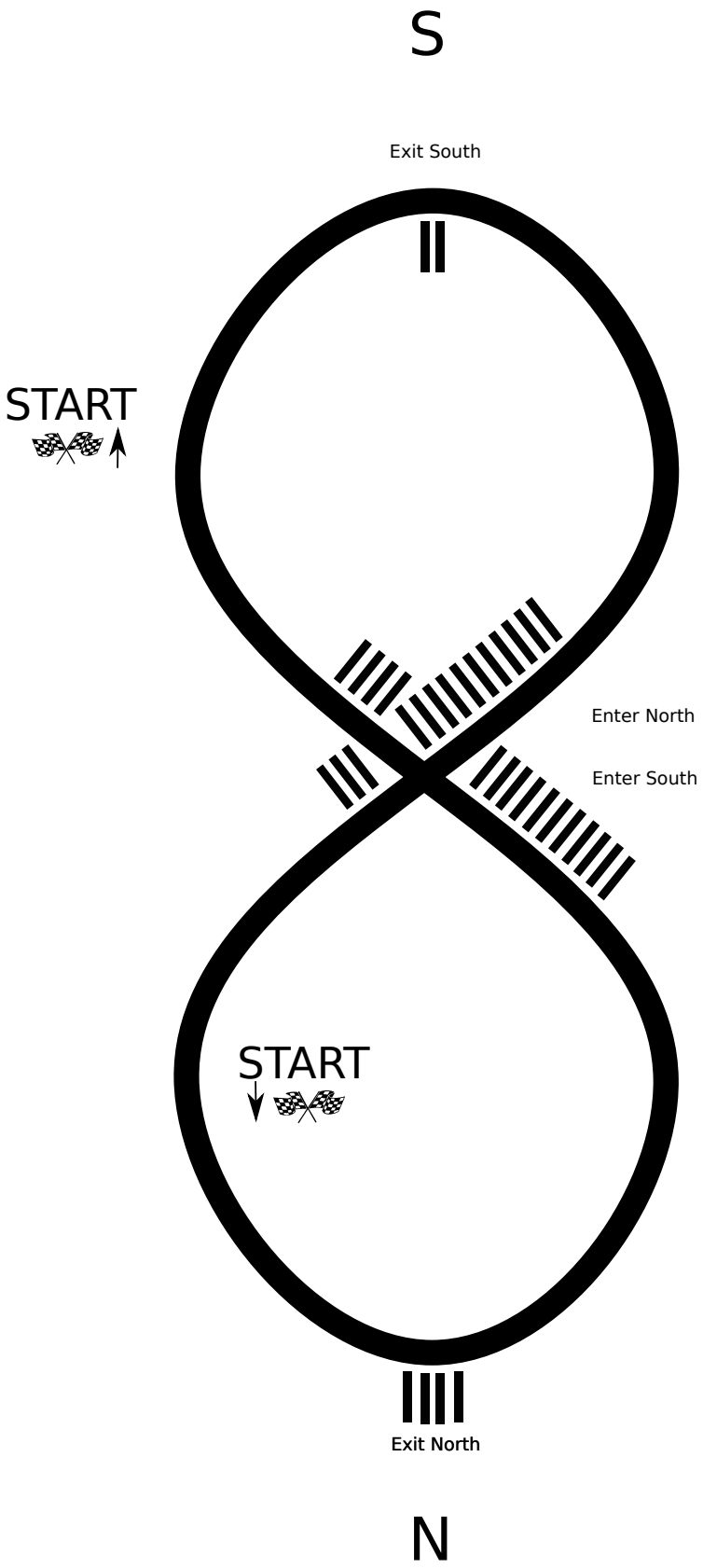

Figure A.1: The first track design

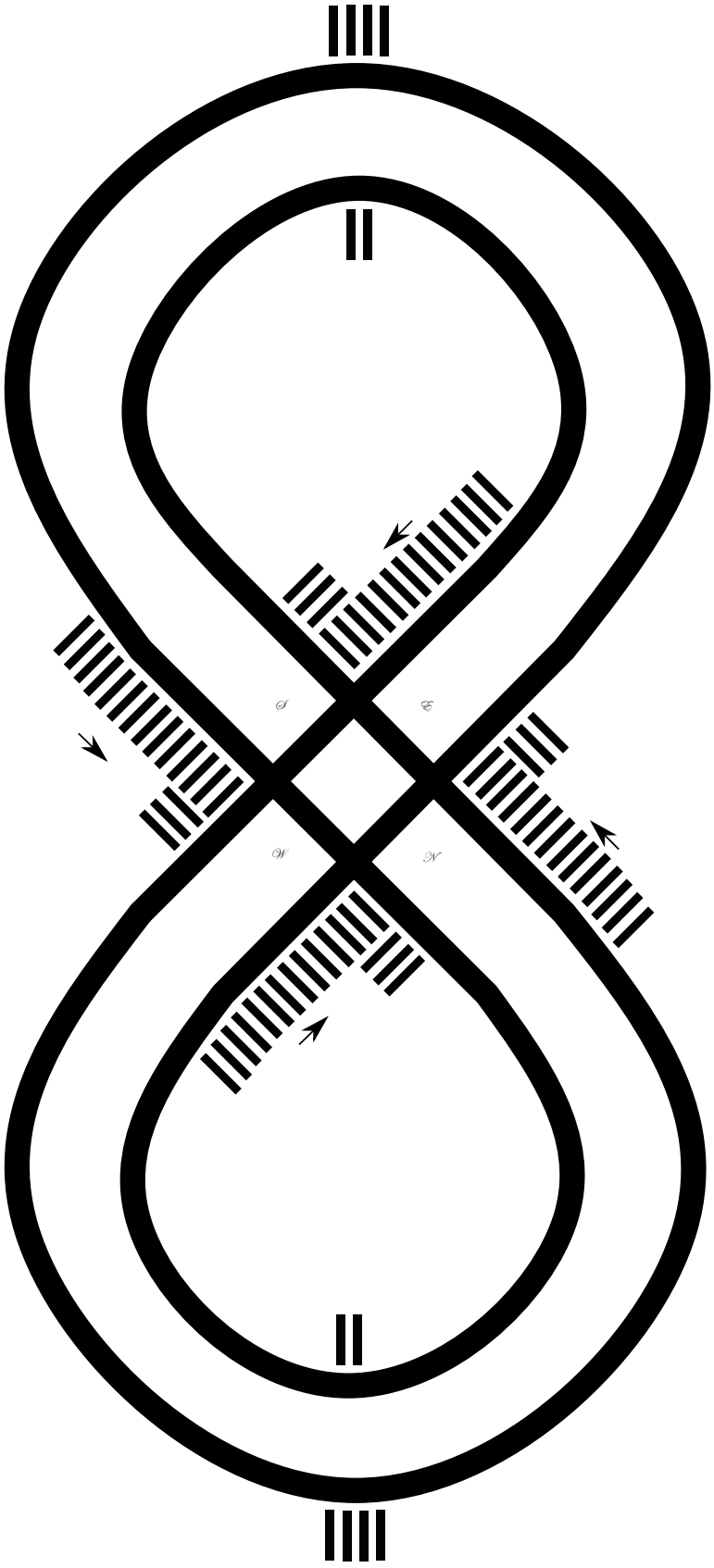

Figure A.2: The second track design

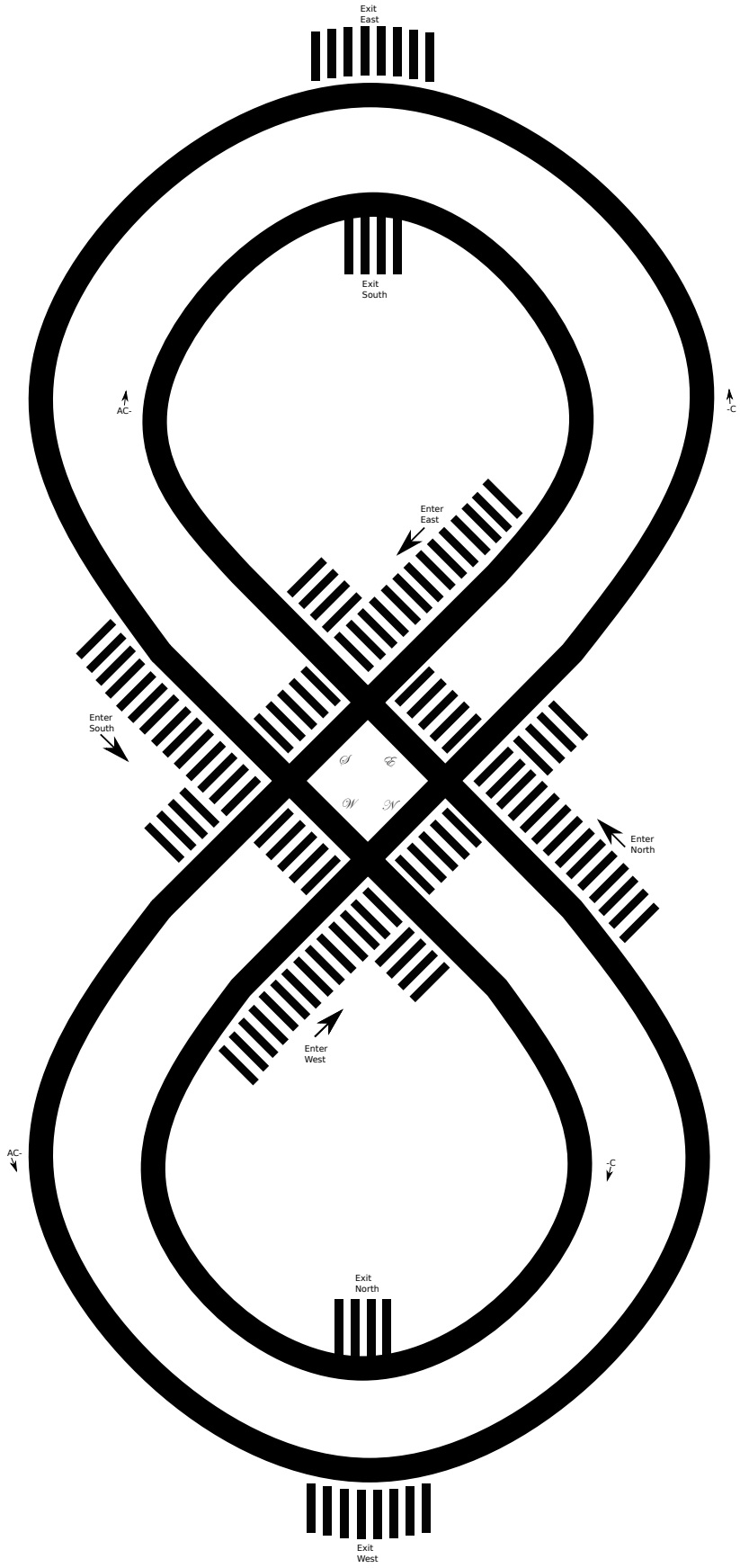

Figure A.3: The final track design

B

Appendix 2

Alg. Type	Test	Nr of Cars	CW	ACW	Wait time (0) [s]	Wait time (1) [s]	Wait time (2) [s]	Avg. wait time [s]	Proportional wait time [%]	Intersections pass (0)	Intersections pass (1)	Intersections pass (2)	Cars per min though the intersection
TrafficLightAlg	1	1	1	-	-	105,19	-	**105,19**	**17,53%**	-	38,00	-	**3,80**
TrafficLightAlg	2	1	0	-	101,88	-	-	**101,88**	**16,98%**	28,00	-	-	**2,80**
TrafficLightAlg	3	1	2	-	-	-	170,17	**170,17**	**28,36%**	-	-	29,00	**2,90**
TrafficLightAlg	1	2	1	2	-	119,13	133,96	**126,54**	**21,09%**	-	36,00	37,00	**3,65**
TrafficLightAlg	2	2	0	1	100,55	133,91	-	**117,23**	**19,54%**	27,00	27,00	-	**2,70**
TrafficLightAlg	3	2	0	2	87,83	-	169,45	**128,64**	**21,44%**	27,00	-	28,00	**2,75**
TrafficLightAlg	1	3	0,1	2	155,10	171,48	116,07	**147,56**	**24,59%**	25,00	28,00	36,00	**2,97**
TrafficLightAlg	2	3	1,2	0	82,91	137,54	0,00	**73,48**	**12,25%**	28,00	27,00	28,00	**2,77**
TrafficLightAlg	3	3	0,2	1	103,47	127,97	183,29	**138,25**	**23,04%**	26,00	27,00	28,00	**2,70**
TrafficLightAlg	**AVG**	**1**	-	-	101,88	105,19	170,17	**125,74**	**20,96%**	28,00	38,00	29,00	**3,17**
TrafficLightAlg	**AVG**	**2**	-	-	94,19	126,52	169,81	**124,14**	**20,69%**	27,00	31,50	32,50	**3,03**
TrafficLightAlg	**AVG**	**3**	-	-	113,83	145,66	99,79	**119,76**	**19,96%**	26,33	27,33	30,67	**2,81**
QueueAlg	1	1	1	-	-	0,00	-	**0,00**	**0,00%**	-	41,00	-	**4,10**
QueueAlg	2	1	0	-	0,00	-	-	**0,00**	**0,00%**	38,00	-	-	**3,80**
QueueAlg	3	1	2	-	-	-	0,00	**0,00**	**0,00%**	-	-	43,00	**4,30**
QueueAlg	1	2	1	2	-	16,81	3,66	**6,58**	**1,10%**	-	37,00	44,00	**4,05**
QueueAlg	2	2	0	1	0,00	6,25	-	**6,25**	**1,04%**	39,00	38,00	-	**3,85**
QueueAlg	3	2	0	2	14,49	-	6,90	**10,70**	**1,78%**	35,00	-	41,00	**3,80**
QueueAlg	1	3	0,1	2	0,00	3,29	51,72	**18,34**	**3,06%**	38,00	37,00	39,00	**3,80**
QueueAlg	2	3	1,2	0	25,73	4,13	1,46	**10,44**	**1,74%**	37,00	35,00	37,00	**3,63**
QueueAlg	3	3	0,2	1	1,75	24,16	77,97	**34,63**	**5,77%**	35,00	35,00	35,00	**3,50**
QueueAlg	**AVG**	**1**	-	-	0,00	0,00	0,00	**0,00**	**0,00%**	38,00	41,00	43,00	**4,07**
QueueAlg	**AVG**	**2**	-	-	7,25	11,53	5,28	**7,84**	**1,31%**	37,00	37,50	42,50	**3,90**
QueueAlg	**AVG**	**3**	-	-	9,16	10,53	43,71	**21,13**	**3,52%**	36,67	35,67	37,00	**3,64**

Figure B.1: All tests performed