DEGREE PROJECT IN COMMUNICATION SYSTEMS, FIRST LEVEL
STOCKHOLM, SWEDEN 2014

Inter-Vehicle Communication with Platooning

JOAKIM SANDBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Inter-Vehicle Communication with Platooning

Joakim Sandberg

2014-07-08

Bachelor’s Thesis

Examiner & Academic adviser
prof. Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

Today's way of driving works very well, but there can be substantial improvements made in the road systems and in the vehicles themselves. Many of the disadvantages of current road systems and vehicles can be removed in the future by using appropriate information and communication technology.

A disadvantage that has been considered to be a major problem for many years is the fossil fuel-consumption of vehicles. Hybrid-cars and all-electric cars are being developed to reduce the use of fossil-based fuels. Since it could take a long time for these new types of vehicles to replace vehicles currently using internal combustion engines, development must also seek to improve current vehicles. Fuel-savings and safety are two major aspects that researchers and vehicle manufacturers are trying to address.

One approach that provides fuel-savings is driving in a convoy. Both Scania and Volvo are currently developing this approach. They aim to achieve the same goal, but in two different ways - since they do not build upon the exact same concepts. Scania is a major manufacturer of trucks and buses, while Volvo is a major manufacturer of trucks, buses, and cars. Both are seeking to improve the fuel-savings for trucks and busses, but Volvo is also seeking to improve fuel-savings for cars.

Unfortunately, with every solution are new problems. Convoy driving brings advantages, but appropriate communication between the vehicles of the convoy and those seeking to join a convoy is necessary for this approach to work well. This is particularly challenging as these vehicles are in moving while communicating. For this reason, the communication needs to utilize wireless links.

This thesis shows in more detail how the inter-vehicle communication works using Wi-Fi and why this is a good media to use when driving a convoy. The testing of Wi-Fi between two driving vehicles and in implementation of two model vehicles shows another perspective of Wi-Fi than today's use of it.

Keywords

IT-system, fuel savings, convoy, Scania, Volvo, wireless communication, Wi-Fi, model vehicles.

Sammanfattning

Dagens sätt att köra i samhället fungerar väldigt bra men det finns naturligtvis massor av nackdelar med olika vägsystem och fordonen själva. Dessa nackdelar kan i framtiden försvinna med utvecklingen av IT-systemen.

En stor nackdel som setts som ett problem sen flera år tillbaka är bränsleförbrukningen hos fordonen. Det finns hybridbilar och t.o.m. elbilar vilka utvecklas i syfte att spara på jordens bränsle resurser. Men eftersom det antagligen kommer ta flera tiotals år innan dessa fordon kommer ersätta dagens fordon med bränslemotorer så måste utvecklingen också gå i två vägar, nämligen att förbättra dagens bränsledrivna fordon. Bränsleförbrukning och säkerhet är de två främsta aspekterna vid denna typ av utveckling.

Ett system som faktiskt förbättrar bränslebesparing är att köra på led som en konvoj. Detta körsystem utvecklas just nu av två större företag, Scania och Volvo. De siktar mot samma mål men har två olika tillvägagångssätt då de inte är i grunden exakt likadana företag. Scania bygger lastbilar och bussar medan Volvo förutom dessa fordon även bygger bilar. Detta ger Volvo en chans att även förbättra bilkörandet.

Men med varje lösning kommer det nya problem. Detta sätt att köra ger givetvis fördelar men man oroar sig ändå för kommunikationen som behövs för detta system. Detta är inte enheter som står stilla på exempelvis ett kontor eller flygplats, utan det är enheter som rör sig ständigt, vilket betyder att kommunikationen måste vara trådlös.

Denna rapport går in mer i detalj hur den externa kommunikationen mellan fordon fungerar med Wi-Fi och varför det är ett bra protokoll att använda i konvojer. Testerna med Wi-Fi körandes i två bilar och även i två små modellbilar ger Wi-Fi ett annat perspektiv än dagens användning.

Nyckelord

IT-system, Minskad bränsleförbrukning, Förbättrad bränslebesparing, Konvoj, Scania, Volvo, Trådlös kommunikation, Wi-Fi, modellbilar.

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

ACC	adaptive cruise control
AEBS	Advanced Emergency Braking System
AIS	automatic identification systems
BSW	Blind Spot Warning
CAN	Control Area Network
DASR	Digital Airport Surveillance Radar
DNPW	Do Not Pass Warning
DSRC	Dedicated Short Range Communication
EEBL	Emergency Electronic Brake Lights
FCW	Front Collision Warning
GM	General Motors
GNNS	Global Navigation Satellite System
GPS	Global Positioning System
IMA	Intersection Moving Assist
ITS	Intelligent Transportation System
LCW	Lane Change Warning
LDWS	Lane Departure Warning System
LED	Light Emitting Diode
LIDAR	LIght Detection And Ranging
LIN	Local Interconnect Network
PoE	Power over Ethernet
P2P	Peer To Peer
SARTRE	SAfe Road TRains for the Environment
STDMA	Self-Organised Time Division Multiple Access
TTL	Time To Live
U.S. DOT	United States Department Of Transportation
USDR	UltraSound Distance Radar
V2I	Vehicle To Infrastructure
V2V	Vehicle To Vehicle
VDL	VHF Digital Link
WAVE	Wireless Access in Vehicular Environments
WIFI	Wireless Fidelity (i.e. Wireless Networks)
WLAN	wireless local area network

1 Introduction

This chapter describes the specific problem that this thesis addresses, the context of the problem, the goals of this thesis project, and outlines the structure of the thesis.

1.1 General introduction to the area

This thesis will introduce readers to the general area of convoy driving and then conduct a deeper examination of inter-vehicle communication (i.e., communication between the vehicles in a convoy).

As one can notice when driving today, the distances between vehicles depends on the road surface, weather conditions, type of roadway, other vehicles, and local speed limit. For example, on a normal concrete road with a speed limit 30 km/h the distance between vehicles may be only a few meters, while on a highway during winter the distance should be at least 10 meters. Today this inter-vehicle separation is normal behaviour, as drivers do *not* want to crash into other vehicles.

Today drivers are eager to achieve fuel savings due to the high cost of fuel and their desire to live in a more sustainable manner. There are many ways to save fuel while driving. One factor that increases the amount of fuel that must be consumed to move a vehicle forward is air resistance. Decreasing air resistance can save fuel. One method is to drive closer to the vehicle in front of you. In this case, the airflows above the vehicle in front of you can will go direct above your vehicle rather than first traveling down between the vehicles and then go over your vehicle. Figure 1-1 to Figure 1-3 show these two different patterns of airflow.

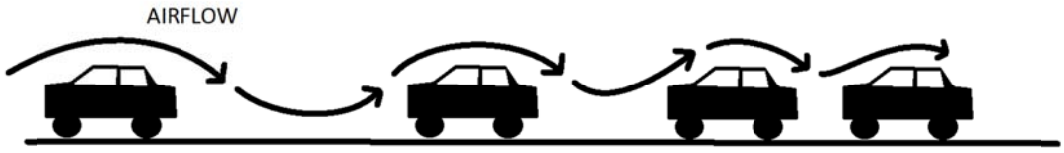


Figure 1-1: Airflow with large and small inter-vehicle separation

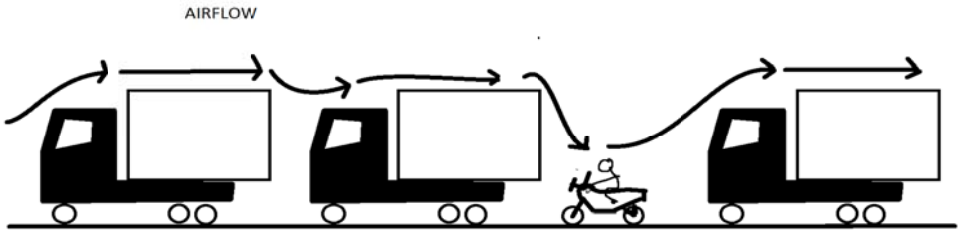


Figure 1-2: Airflow along trucks

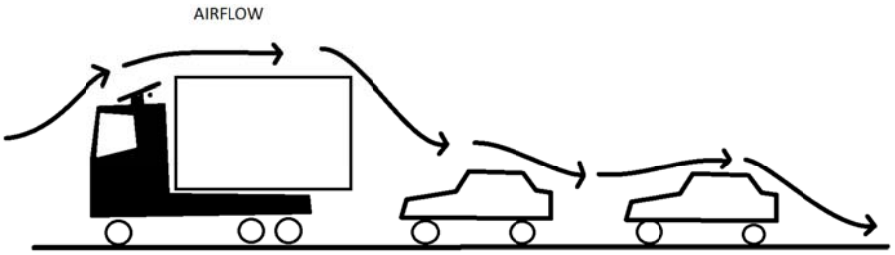


Figure 1-3: Airflow along one truck and several cars

Unfortunately, at high speeds it is impossible for human drivers to drive closely following the vehicle in front of them due to the high risk of crashing into this vehicle if the vehicle in front brakes rapidly. The solution is to introduce a system that prevents such crashes, while enabling close inter-vehicle spacing. In order to do this the vehicles must communicate with each other, leading to inter-vehicle communication. Not only must there be inter-vehicle communication, but also this communication is safety critical – so we must ensure that this communication is both *secure* and *reliable*.

There are three types of communications that are candidates for implementation in systems today: short range wireless communications systems, such as wireless local area network (WLAN) (e.g. Wi-Fi*) and Dedicated Short Range Communication (DSRC), RADAR, and Global Navigation Satellite System (GNSS) – such as Global Positioning System (GPS). Each of them has their own advantages and disadvantages. Our expectation is that one or more of these alternatives can be used to introduce a secure and reliable system that enables vehicle to closely follow the vehicle in front of them, thus building a convoy.

1.2 Problem definition

We define the problem, and then describe the requirements for reliability. Next, several of the different types of communication technologies that could potentially be used in inter-vehicle communication are considered†. Finally, we consider the current environment and what requirements a solution using wireless communication has to fill.

1.2.1 Problem

Although the problems of realizing a convoy might seem more related to traffic engineering, design, and perhaps the construction of vehicles, this project takes an information technology (IT) perspective – specifically focusing on the use of inter-vehicle communications to realize and facilitate vehicle convoys. The problem is to select an appropriate communication technology (including protocol stack) for inter-vehicle communication in order to realize and facilitate vehicle convoys.

* Wi-Fi is a trade name for devices that utilize the IEEE 802.11 standards.

† For a detailed discussion of wired communication *within* vehicles see the candidate thesis by Rasmus Ekman [1].

1.2.2 Reliability

In general, there are two ways of realizing inter-vehicle communication: wired and wireless communication links. While wired communications in many settings offers high reliability it is infeasible in some settings. For example, it is possible to used wired communication between a truck and its trailer (or trailers). In this setting a wired communications link works well and this solution has been in use for many decades. Although wireless replacements for this link have been investigated (see for example [2]).

In the setting that is the focus of this thesis project, wireless communication will be used for inter-vehicle communication. This means that each vehicle needs to be equipped with one or more antennas and transmitters & receivers. Depending on the specifics of the wireless link the transmitters and receivers will use different types of antennas (in the case of radio links) or emitters/detectors (in the case of optical links).

The reliability of inter-vehicle communication is important to prevent incorrect information from being used. Sources of error include environmental noise, weather, and intentional attacks. For example, a malicious attacker could generate fake packets; conduct signal gaming against wireless receivers; attempt to introduce viruses to cause problems with the electronic systems connected to the receiver(s), etc. A secure and reliable system has to consider both accidental and purposeful impairments.

Different types of wireless links offer different levels of reliability and safety. For example, WLAN links can enable reliable and very secure communication [3]. WLANs are very common today and nearly all smartphones, laptops, etc. have one or more WLAN interfaces built into them. WLAN links work well when the end-points are close to each other; however, these links can be subject to interference due to other transmitters and the interaction of the radio waves with the environment. DSRC links are similar to WLAN, but they have been allocated a dedicated portion of the wireless spectrum. For example, the US Federal Communications Commission (FCC) has allocated 75 MHz of spectrum at 5.9 GHz for vehicle safety applications[4]. RADAR systems emit a signal and listen for reflections of this signal. A scanned RADAR system can be used to scan a volume of space in front of a vehicle (to identify obstacles such as vehicles, persons, animals, structures, etc.). Additionally, the RADAR system can exploit the Dopper shift information to tell if the surface causing the reflection is moving toward or away from the RADAR emitter. A RADAR system's reliability depends upon how sensitive the receiver is, how rapidly the system scans the volume of space, how the received signal is processed, etc. GNSS systems, such as GPS, need to be able to get signals from three or more satellites in order to compute their position in three dimensions and from four or more satellites in order to compute their position and time. Buildings, vehicles, tunnels, etc. may prevent the GNSS receiver from being able to receive signals from a sufficient number of satellites, thus reducing the accuracy of the positioning (and time) information. In severe cases, the device may not be able to compute its position. Many GNSS receivers use supplementary sources of information (such as accelerometers) to improve their reliability. In some cases, a combination of GNSS receiver and information from fixed base stations is used to provide improved accuracy and reliability.

1.2.3 Inter-Vehicle Communication

As noted earlier inter-vehicle communication requires high reliability and security as this communication is being used in a safety critical application. This is particularly true when the vehicles are moving at high speed with only very limited separations. In this setting data loss, incorrect information, and unreliability may result in severe damage or even death.

The primary use of GNSS systems in vehicles today is for navigation to a destination and to get directions to that destination. Trucks have utilized a GPS receiver to help the truck achieve a better fuel economy, for example by using information from the GPS receiver to select the most appropriate gear and when to use brakes during a descent. Since trucks have much greater fuel consumption than

normal cars (0.4-1.2 L/10 km for a car versus 3.5-4.5 L/10 km for a big truck with a heavy load[5]) the fuel savings by electing the most appropriate speed, gear, braking profile, etc. can be considerable for trucks. Additional information concerning the use of GPS can be found in [6]. Later in this thesis we will examine if GNSS can be helpful in inter-vehicle communication, for example by considering the use of these systems as proposed in Håkan Lans's Self-Organised Time Division Multiple Access (STDMA)[7] data link to an create automatic identification systems (AIS) (for maritime use) and VHF Digital Link (VDL) Mode 4 for air traffic control.

1.2.4 Environmental issues

Today there are many different communication systems. Many of these systems are wireless. The communication devices have some requirements in order for the communication to work properly. First, they have to reach other devices – this means distance is an important factor. The material of buildings and objects that the radio waves must pass or reflect from is a factor. Other requirements are for example that communication should be available despite the weather or other traffic passing between devices. The later means that wireless communication should be possible even when other devices are using same protocol, same source and destination address, and even same frequency. Therefore, the different types of wireless communication systems should *not* disturb the existing systems.

Additionally, it is important that these systems should not constitute a health problem[8]. Today there are many different types of wireless communication systems, such as cellular phones (including mobile 3G and 4G systems), emergency services, Wi-Fi, and many more. All of these devices emit radio waves. With more wireless communication, the cumulative amount of radio wave energy is increasing. This is an important aspect since radiation has a negative effect on human body[9].

Another factor to take into account is the dead zones that occur either due to long distance from the source or because different signals destructively interfere with each other [10]. In summary, when creating wireless networks, careful considerations of environmental issues are as important as the functionality of the system.

1.3 Goals

The main goal of this thesis project is to examine how to utilize inter-vehicle communications to realize and facilitate vehicle convoys. This examination must consider whether this communication can be sufficiently reliable and secure to be used to safely realize a convoy. This analysis will consider the effect of convoys on traffic flows and whether convoys will create new problems.

1.4 Problem context

The examination carried out in this thesis project must consider environmental, technical, and safety aspects of an inter-vehicle communication system. Safety is obviously a critical aspect since we formation of a convoy will reduce the spacing between vehicles and should collisions occur the risk of damage or even death is high. This is especially true when the vehicles are moving at high speeds as the system will need to operate largely automatically since there is not enough time for the slow reactions of humans to prevent a collision (or in the worst case a chain reaction of collisions).

1.5 Research Methodology

To find relevant information about earlier work and information that is trustworthy I have chosen to search popular websites for the different subject areas. I will not search forums and online encyclopaedias (such as Wikipedia) since they are *not* reliable sources. School libraries, websites of public authorities and newsrooms are considered reliable sources of information. Later in this thesis, I

will analyse information using common sense and the scientific and engineering methods that I have learned in my studies, and if necessary, I may even need to consider some less reliable sources.

1.6 Structure of this thesis

This thesis is divided into chapters. This first chapter introduced the thesis area, the problem, and the goals of this thesis project. Chapter 2 provides relevant background information for the readers of this thesis. Chapter 3 describes the method used to solve the problem and achieve the stated goals. Chapter 4 gives the results of the analysis of the proposed solution(s). The thesis concludes in Chapter 5 with a summary of conclusions, suggestions for future work, and some reflections on the economic, social, sustainability, and ethical aspects of this thesis project.

2 Background

Convoy driving has not been introduced globally, but it has already gained some interest. The primary focus thus far has been on larger vehicles (such trucks) with heavy loads, but this approach has also been tried with cars. This chapter will describe some of the early projects to introduce convoy driving.

The chapter begins with a description of a RADAR sensor, as this is an important sensor that has been applied in some approaches to convoys. This is followed by a description of how Wi-Fi is implemented in vehicles and how some vehicle manufacturers envision the use Wi-Fi. This is followed by a description of the DSRC system. Finally, the chapter ends with a summary of related work.

2.1 RADAR

Radar has been used in many applications ranging from distance measurement to scanning for obstacles, clouds, vehicles, etc. In air traffic control airplanes communicate with air traffic controllers and airport based guidance systems in order to navigate safely from one location to another while avoiding other aircraft, storms, mountains, etc. The same idea is used with boats and ships at sea. Additionally, traffic control use radar for detecting vehicles that are going faster than the posted speed limit. Road/highway departments use radar to detect voids under the road surface (in order to plan road repairs).

2.1.1 DASR

Traditional air traffic control radar has enabled airport controllers to keep track of planes in a three dimensional space with high precision and to monitor weather conditions. Today these high precision radars for areas near air terminals are being replaced the Digital Airport Surveillance Radar (DASR) which are more efficient[11].

2.1.2 Bosch

Bosch has developed high precision radars for usage in vehicles. Radar is used in vehicles to assist other systems with the aim of increasing driving safety[12]. To understand how this is realized we need to know a little bit about several other technologies used in modern vehicles.

A controller area network (CAN) is a communication system *within* the vehicle. It is typically realized as a bus that interconnects different types of microcomputers so that these separate controllers can communicate with other systems. For more information about CAN see the thesis of Rasmus Ekman[1].

A related concept is a Local Interconnect Network (LIN). LIN (like CAN) is an internal communication network, but it allows smaller and more lightweight systems to connect with each other. To use CAN the microcomputers need to be relatively advanced, more complex, and typically more expensive; while LIN allows simpler and lower cost interconnections.

Adaptive Cruise Control or Automatic Cruise Control (ACC) is implemented in most new vehicles as an upgrade from the standard cruise control. ACC is similar to standard cruise control, but is adaptive so that if the vehicle in front is braking, then this car will apply its brakes. ACC enables a vehicle to maintain a minimum fixed distance from the vehicle ahead of it.

According to EU regulations[13], Advanced Emergency Braking System (AEBS) needs to be implement in different classes of vehicles (categorized in terms of the total number of wheels, number of seats, or freight weight and the total weight), specifically vehicles with the following properties[14]:

- M2 = Vehicles with mass under 5 tonnes*, at least four wheels, and maximum of eight seats.
- M3 = Vehicles with mass above 5 tonnes, at least four wheels, and more than eight seats.
- N2 = Vehicles for goods carriage between 3.5 tonnes and 12 tonnes.
- N3 = Vehicles for goods carriage above 12 tonnes.

A Lane Departure Warning System (LDW) is a system that warns the driver if the vehicle is beginning to move outside of its lane. The system takes into account whether the turn indicator has been activated. This technology can also be used to assist the driver to keep the vehicle centred in the lane.

ACC, AEBS, and LDW systems are used to assist the driver in order to increase safety. ACC, AEBS, and some LDW systems use radar as input. Bosch is still developing their radar systems. The first version used a radar system operating in the 24 GHz frequency band to scan the area in front of the vehicle. However, this frequency is used by other applications[15], hence a 77 GHz version was subsequently introduced[16]. As the frequency increases the results are greater precision and the area that the radar can cover increases, but the required power can be decreased if the components are fully integrated[17: pp2746-2756].

2.1.3 Functions

The radar in vehicles can achieve a lot even though it is a quite simple component. If the radar transceiver is placed at the vehicle's front grill, it can scan an area from half a meter ahead up to 300 metres. Additionally, it may be able to detect vehicles driving in adjacent lanes when this other vehicle is a couple of metres ahead of the transceiver. That means when driving on a single lane road in the countryside, when the road goes to the left or the right, the radar still has knowledge of the vehicles ahead.

2.1.4 Audi

Audi has introduced a new development to increase the safety. They implement radar to scan ahead and to scan *behind* the car. Since the driver of vehicle A (the lead vehicle) cannot apply the brake of vehicle B (unless they are communicating), all vehicle A can do is to prepare for a possible crash with vehicle B coming from behind. In order to prepare, vehicle A scans the area behind it and if vehicle B is approaching too fast, it tightens the seatbelts, folds up seats, and lowers the windows to about 90%, leaving just a small gap for air. If a crash does not occur, then the system restores the previous settings and carries on.[18]

2.1.5 Scania

Scania is manufacturing trucks and buses. Today's trucks and busses consist of many IT-systems. Some of these IT systems directly control and operate engines and gearboxes. For example, it is possible to optimize the engine speed and choice of gear *without* any driver input. Additionally, the driver can control these subsystems.

Although most of these systems can be switched on or off by the driver, most drivers prefer to keep these systems engaged because they contribute substantially to efficient and safe operation of the vehicle. Several years ago Scania developed and implemented[19] standard cruise control in many cars. This innovation has the ability to maintain a fixed vehicle velocity, even when going up and down small hills. Some cars and Scania trucks have extended ACC to make use of RADAR system that scan in front of the vehicle in order to detect upcoming obstacles. If the vehicle is closing on an obstacle, then the ACC slows the vehicle down to a speed that will not result in a collision. This is of course most

* Here tonnes refers to metric tonnes.

efficient at high speed on highways as opposed to driving within cities where starting- and stopping occurs more frequently.

Scania is in process of introducing what they call “platooning”. Their first step is to use ACC to maintain a specific separation distance from the vehicle in front of the vehicle, thus establishing a convoy. In this first phase they simply programmed the ACC to maintain an optimal distance to vehicle in front in order to gain efficiency, hence minimizing fuel consumption[20]. They are now implementing a combination of Wi-Fi, GPS, and RADAR to support platooning.[21]

2.1.6 Volvo

Volvo has a very ambitious platooning effort in their project called SAfe Road TRains for the Environment (SARTRE). This project builds upon a threefold problem statement: environment, safety, and congestion.

Volvo has taken a different approach than Scania. Since Volvo also manufactures cars, they have focused in their project on convoys consisting of *both* large trucks and cars. They have already implemented this system to realize a convoy consisting of a leader truck (which acts as a master) and following cars (which act as slaves to this master). What is different with their platooning from Scania’s systems is that the following cars are not only using ACC but also use automatic steering. In other words, the cars are driving themselves without any input from the driver. These cars simply follow the leader truck’s speed and direction.[22]

The radar system in Volvo vehicles has been used mainly to prevent collisions in a city (where speeds are slow). Their radar system uses a Light Detection and Ranging (LIDAR) sensor which detects obstacles ahead and can stop the car when operating at speeds lower than 30km/h. As with many of the automotive radar systems which implement automatic braking, the computer in the car prepares the brakes when the radar senses that a crash is highly likely, enabling the brakes to be applied more quickly.[23]

2.2 Wi-Fi

Wi-Fi is widely used to realize WLANs. When one hears the term “Wi-Fi”, many people immediately associate it with home networks, office networks, school networks, public networks, and perhaps even networks on airplanes (as some airlines have installed WLANs on their aircraft).

What if Wi-Fi was used in another application area, such as for vehicular traffic? Today the Internet is used for traffic control, for example the traffic cameras and red light cameras of some cities uses either Power over Ethernet (PoE)[24] or 3G or 4G cellular networks[25] to communication with a central surveillance control center. Many vehicle manufacturers have already introduced Wi-Fi in vehicles in order to improve their vehicles in different ways. Of course, the main function has been safety improvements, but this technology has also been applied to improve traffic flow.

2.2.1 GM

General Motors (GM) has introduced Wi-Fi as a safety improvement of their vehicles. Their idea is that vehicles can communicate which each other, i.e., vehicle-to-vehicle (V2V), and even communicating with pedestrians, bicyclists, and other infrastructures, i.e., vehicle-to-infrastructure (V2I). Their hypothesis is that peer-to-peer (P2P) communication can decrease communication latency, since latency is a very important factor in preventing traffic accidents. If one did not use P2P communication, then all communication would to go through access points which will introduce a bit more delay. As a result not only vehicles need to be equipped with access points or routers, but other devices, such as each pedestrian’s smartphone or tablet must have applications that support this P2P communication when the pedestrian is traveling near vehicular traffic. GM’s system utilizes “Wi-Fi

Direct®" [26] (a P2P standard for Wi-Fi) to prevent many different types of accidents once it is widely deployed in cities.[27]

2.2.2 Ford

The United States of America's Department of Transportation (DOT) is working on V2V together with many vehicle manufacturers. However, the US DOT is *not* in charge of what kind of V2V system is implemented in vehicles. Ford is one of many vehicle manufacturers that have chosen to work with Wi-Fi systems. Their system has every vehicle (with this system implemented) broadcast its position, heading, and speed to nearby cars. This results in every car with this system being able to calculate if another car could potentially crash into this vehicle. This Wi-Fi operates on a secure channel so that only V2V cars speak to each other.[28]

2.3 DSRC

Similar to Wi-Fi, DSRC is a wireless protocol developed for V2V by the U.S. DOT and several vehicle manufacturers. DSRC is an element of an intelligent transportation system (ITS). Although DSRC does not support fully automatically driving vehicles, it is a major step toward a more secure and autonomous world of vehicles and traffic.

DSRC can be divided into many subcategories of systems. All have one common goal: safety. Since DSRC includes a lot of safety systems which works in parallel, a lot of information must be sent between and received by the DSRC devices. This information is carried via different types of messages. This data includes:

- GPS position,
- Speed,
- Acceleration,
- Heading,
- Transmission state,
- Brake status,
- Steering wheel angle,
- Path history, and
- Path prediction.

All but the last two are already common data in most vehicles, where this data may be used for existing safety systems. As a result there are already computers inside vehicles collecting data from various sensors, but in DSRC path history and path prediction have been added [29]. The reasons for adding these two additional types of data are described in the following two subsections.

2.3.1 Path history

This system combines data from the different types of DSRC messages to calculate the historic route for the last couple of hundreds of meters. The system dynamically creates a series of waypoints at different distances between them (depending on the road). As the vehicle moves, it deletes older data points when they are no longer necessary. For a long straight path (no sudden curves), the path history is very simple, therefore the distance between data points can be quite far. However, when driving around a curve, the direction is always changing so the data points are closer to each other in order to give more precise data about the historic path of the vehicle.

2.3.2 Path Prediction

Path prediction is similar to path history, but instead of storing waypoints for the path that has already been driven a path is calculated based upon the planned route to a destination, i.e., the potential route the vehicle will follow in the future. For this to work in the short term – without a specific destination, the system combines GPS data with the steering wheel angle, brake setting, and acceleration, to predict a possible route. DSRC uses path prediction and path history to predict and prevent future crashes.

2.3.3 Emergency Electronic Brake Lights

When driving several vehicles in a row, as in a convoy, it is sometimes difficult for drivers in the middle or further back in the convoy to see what is actually happening far in front of them, as most of the time they can only see the vehicle in front of them. Therefore, if an accident or a sudden stop occurs by the first driver of the convoy, the last driver may not know of this until all of the drivers ahead of them have noticed the sudden stop, thus a crash is highly probable.

Emergency Electronic Brake Lights (EEBL) can be used to prevent this problem by communicating to all of the trailing vehicles when a vehicle ahead is making a sudden stop. In such an event EEBL activates a light on the dashboard or windscreen increasing the probably that the driver will be able to stop before crashing.

2.3.4 Blind Spot Warning

DSRC also seeks to improve safe lane changing while driving. However, the blind spot is a problem that exists when driving, but it is possible to assist drivers with a Blind Spot Warning (BSW) system. When a vehicle is detected within the blind spot area and the driver applies their turning indicator a light on the rear view mirrors will flash.

2.3.5 Lane Change Warning

Lane Change Warning (LCW) is similar to BSW, but by using V2V DSRC can know that a vehicle is approaching the blind spot *before* a lane change is attempted, preventing a stressful situation for both the driver attempting to change lanes and the driver approaching in that lane. If the driver heeds the warning this may prevent an accident. The system uses the same flashing lights that BSW uses.

2.3.6 Front Collision Warning

The Front Collision Warning (FCW) system uses the EEBL as a warning system, whether driving in a convoy or approaching a stationary vehicle on the road ahead when no reaction is taken by the driver.

2.3.7 Do Not Pass Warning DNPW

The Do Not Pass Warning (DNPW) is warning system that may prevent many common stressful situations. Imagine that a truck is climbing slowly up a hill and you want to pass it to maintain your tempo. However, since you are going up a hill, you might not be able to see an oncoming vehicle until it is very close to you. DNPW uses DSRC to communicate with approaching vehicles (also using DSRC), thus as soon as you attempt to pass (applying the turning indicator and switching lane) the DNPW will warn you if another vehicle is approaching in opposite direction.

2.3.8 Intersection Moving Assist

There are many intersections in the road network today. Not all of them are safe, due to limited visibility (with buildings or trees blocking your light of sight or due to a vehicle standing on the side of

the road) or due to a vehicle stopped in the intersection. Intersection Moving Assist (IMA) is a DSRC system that via V2V calculates if another vehicle is approaching the intersection. If so, the system warns the driver to be more careful via flashing lights on the dashboard. If a vehicle has stopped at an intersection and another vehicle is approaching the intersection at high speed the driver of the stationary vehicle can be warned before entering the intersection.

All these various subsystems of the DSRC provide safely improvements. Although DSRC offers many possibilities to an individual vehicle, it works best with all other vehicles are equipped with the same DSRC system. Therefore, there is a potentially long adoption curve and a substantial aggregate investment must be made to achieve a fully functioning system. Additionally, the system will need to be improved and optimized in the future, while the devices that must be installed in each vehicle will need to have a low price. U.S. DOT believes that widespread adoption is still many years in the future.

2.4 Related work

As described above, there are some companies working with developing inter-vehicle communication in different perspectives, such as V2I, V2V, and the vehicle's own technology in traffic such as brake assist and warnings.

Fortunately, there are also a lot of students that are working with the area as well, even though they do not have the same resources available. As long as they have the interest, ideas and the knowledge, they can contribute to the development as well.

There are many theses in this area with all different or quite similar perspective of the area. Here is mentioned three theses done by students of KTH and Linköping University.

There is a thesis done by Simon Eiderbrant [30] which is a more analytic model of the convoys with a very deep mathematic perspective. It is described very deeply with formulas how forces works on the convoy and how different driving environments affect the driving.

Another thesis is done by Joakim Kjellberg [31] and describes also the convoy driving but with a more analytic perspective of algorithms. This is also an interesting perspective since the algorithm is the key of a platoon to be working correctly.

The last mentioned related thesis is done by Mani Amoozadeh [32] and is also a more of algorithm perspective but is more deeply dug into messages that are sent in the convoy. The Certificate Revocation List (CRL) is the key in the thesis.

All of these theses are very interesting because they all are within the same area of work, but each perspective is very different from another.

3 Method

This chapter describes what I am going to do to solve the problem stated in Section 1.2, how I will do it, and how I aim to accomplish the goals stated in Section 1.3. First, I will describe my method of work. Why I have chosen this method and why I reject other methods. This is followed by a description of the goals and how I will attempt to accomplish them. This will be followed by a description of the implementation of a prototype and the planned tests, modelling, and analysis.

The overall work is described as a list of tasks. Each of these tasks is described along with how it should be done. These tasks include setting up a test-bed and what software to be used.

The sections of this chapter will describe the method and guide both physical and theoretical progress in this thesis project.

3.1 Choice of method

I have chosen to organize my work here with the help of Scrum method. This is because I have experience from an earlier course in using this method for a project. From that course (IT-project, built a robot); I learned that using Scrum to organize all of the tasks from the beginning to the end of the project will result in more qualitative progress and maybe also achieve a better result.

Scrum was originally introduced within software develop since software development is more agile than the hardware develop department and Scrum is an agile-method [33].

In my earlier IT-project course, I was introduced to Scrum and learned then that even though we worked with a hardware project we could organize our work with Scrum. Since my work in this thesis includes both hardware development and software development, I have chosen to organize the complete effort with Scrum. Unlike the earlier course and the original applications of Scrum, I will work alone rather than in a team. However, this thesis project is being done in parallel with two other related thesis projects – hence there is some degree of interaction with these other two thesis projects.

I began by collecting every idea I could find in a directory and later sorted through this material. I began by collecting information about previous work in this area. This helped provide me with a good understanding of this area. Next, I began with the actual modelling and testing of network signals (as described in the next chapter). These efforts lead to the realization that I needed to organize by effort. My own method of collecting, storing, and working with information would not have worked with all of the tasks that I planned for this thesis project. One of the first steps was to organize tasks by writing notes and organizing these notes as shown in Figure 3-1.

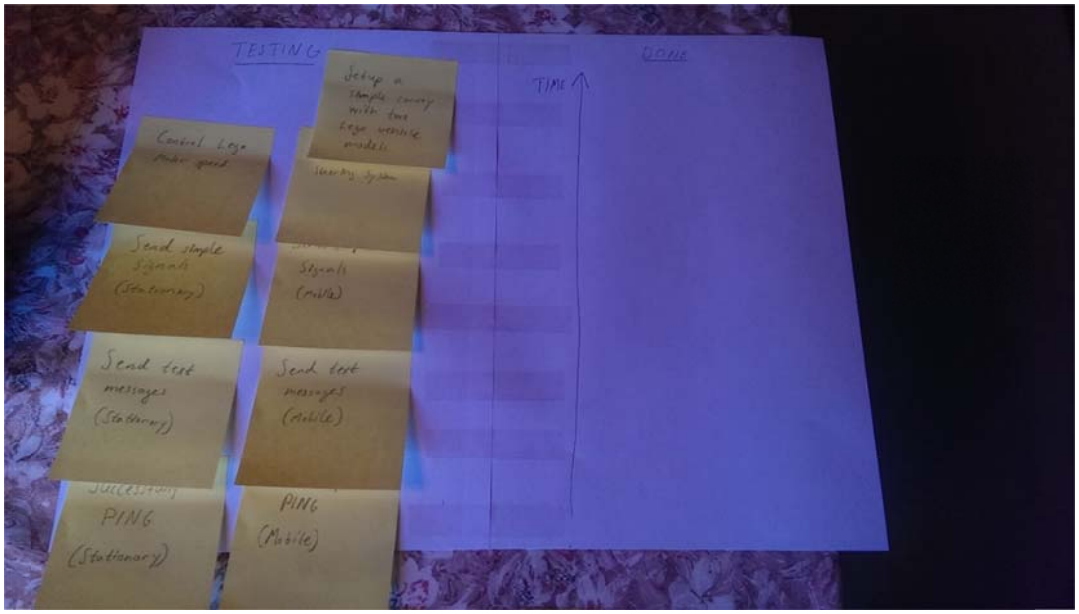


Figure 3-1: Post-IT notes for organizing the work progress

3.2 Goal

My goal was to realize a simple convoy using model vehicles. These models could be used to examine the principle of a convoy and to understand how inter-vehicle communication could be used. This is an ambitious project, but I believed that I could build upon well-established solutions to reach the project's goal.

I was so eager to build the models that I was later going to use that I purchased the parts I did not already have. Unfortunately, this is not a good idea at the very beginning of a project, because tasks may change due to changes in the demands, available resources, or even the whole idea behind the project. Although I built the models initially, I knew that there was a limit to what I could achieve using models. Without further research about inter-vehicle communication or my own testing of different communication systems, it was impossible to create a suitable working system for inter-vehicle communication.

3.3 Progress

My progress was divided into a number of different stages. To achieve a working convoy with the models and gain competence in the area, I had to do many tests in order to understand the different problems that needed to be solved.

The first stage was to collect a lot of information about the area. The next stage was to come up with ideas about how the communication needed to work. These stages established the theoretical foundation for the thesis. The next stage was to test my ideas and analyze the results of these tests in order to change components or adjust the direction of my progress.

When deciding what components I should work with, there were many different options to choose between. Subsequently I needed to make even more decisions about what components to choose. In the following subsections, I will describe some of the initial test stages and their components. The first major decision was to use a router and a WLAN access point for communication between the vehicles. As I had a couple of Arduinos laying around together with a couple of Ethernet Shields for them, rather than purchasing expensive Wi-Fi shields for the Arduino (at ~US$100 each) I decided to use a cheap router as a host and an WLAN access point for a client. Later, this host was placed in the first vehicle of the convoy and one client was placed in each of the following vehicles.

3.3.1 Components

The initial set of component are listed in Table 3-1. Most of these components were chosen because they were available, as opposed to being specifically selected for this project.

Table 3-1: Initial components

Component	Description	Weight (grams)
Computer #1	Dell XPS13 i7 CPU, WLAN Interface, USB- Gigabit Ethernet Interface, Windows 8.1	
Computer #2	HP Pavillion DV6 AMD Turion X2 CPU, WLAN Interface, Ethernet Interface, Windows 7	
Router	D-Link GO-RT-N300 powered with 12 VDC, 0.5 A	200
Access point	NETGEAR WNCE2001 powered with 5 VDC,1 A	50
two Arduinos	MEGA 2560 http://arduino.cc/en/Main/arduinoBoardMega2560	35 each
Arduino shield	Ethernet shield based on Wiznet W5100 ethernet chip. http://arduino.cc/en/Main/ArduinoEthernetShield	25
Arduino shield (for use in models)	Motor shield based on the dual full bridge L298 chip. Motor max current 2A. http://arduino.cc/en/Main/ArduinoMotorShieldR3	25
UltraSound Distance Radar (USDR) sensor	ElecfreaksHC-SR04 ranging module, measurement range ~2cm to ~400cm, measurement angle 15 degrees, measurement accuracy 3mm.	14
Battery (for use in models)	Make model “HQ Sealed Rechargeable battery” 12 V and 1.3 Ah lead acid battery	600

3.3.2 Implementation

Three different configurations were tested. The first test configuration was setup is on a “test-bench” where everything is on a desk and mainly powered by mains power (using a 230 VAC to 12 VDC adapter for the router and a 230 VAC to 5 VDC adapter for the access point). The laptops were powered either by mains power adapters or by their internal batteries.

The second test configuration involved placing the devices in one or two vehicles. Some of the system tests required only one vehicle, but other tests required two vehicles. The components were powered either by internal/external batteries and/or the cigarette lighter socket.

The final test configuration was a LEGO® convoy model with each of the components powered by external batteries.

3.4 Tasks

This section enumerates the tasks that I needed to complete. These tasks were expected to generate a large amount of data that could subsequently be analyzed and documented. As mentioned above there are three different configurations. As shown in the list of components I use a router and an access point for wireless communication. The access point is used as a bridge[34] as shown in Figure 3-2. Therefore, the router will only use *ad hoc* mode to communicate with the access point, i.e., the WLAN will not accept association requests from other Wi-Fi devices trying to connect to router. As computer #1 did not have an Ethernet port, a USB-Ethernet interface was used to connect it to the router or access point.

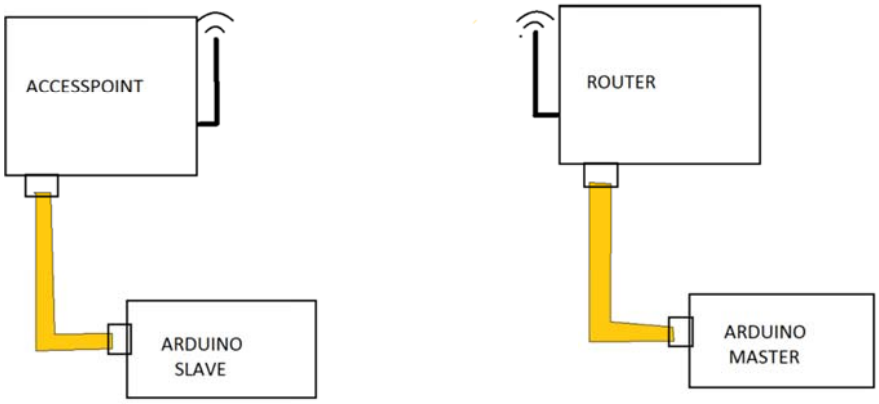


Figure 3-2: Simple setup of an access point, router, and a couple of Arduinos

3.4.1 Successfully PING (Stationary)

This first task was split into several smaller sub-tasks. This task was a simplest test, but it was a key test since this it showed that connectivity could be established. Additionally, this task help acquainted me with the tools that I would use in subsequent testing and I learned how to carry out the testing in a more optimal way.

3.4.1.1 Wired PING between computer and router

The initial test was done to check the connection between a computer and a router using the built in "ping" command via the Microsoft Windows' command prompt. The computer was connected to the router via a CAT5e Ethernet cable.

3.4.1.2 Wired PING between two computers

The second task was to ping one computer from another. Both computers were connected to the same router, hence they should be able to ping each other.

3.4.1.3 Wireless PING between two computers

The task is the same as the above where two computers should be able to ping each other, but in this case one computer is connected to the router and the other computer is connected to the access point.

3.4.1.4 Wired PING between Arduino and router

In this test one Arduino card with an Ethernet shield is connected to the router using a CAT5e Ethernet cable. The test simply pings the router from the Arduino. I modified the code written by Blake Foster as shown in code listing 1.1 in the Appendix [35]. This code sends several pings in a row, similar to what the Windows ping program does. This code sends 10 pings, and either receives them all or prints a time-out error. It also prints the time it took, time-to-live (TTL) fields, and the size of each packet. This code is used in subsequent tests using the Arduino to ping different devices. The only part of the code that I changed was the destination IP address and perhaps the Arduino's IP address if it matched another device's IP address.

3.4.1.5 Wired PING between Arduino and computer

After trying to ping the router, I tried to ping a computer attached to same router. This is a two way ping test. First I performed a ping from the Arduino to the computer and then from the computer to the Arduino.

3.4.1.6 Wired PING between two Arduinos

In this test two Arduinos were connected via cables to same router and they tried to ping each other. The purpose of this test was to check that each Arduino is able to ping the other Arduino, i.e., to establish that they have bi-directional communication.

3.4.1.7 Wireless PING between two Arduinos

In this test one Arduino is connected via a cable to the router. This router is also connected via cable to a computer that is monitoring the network using Wireshark. The second Arduino is connected to the access point via an Ethernet cable. This second Arduino can also be connected via a USB cable to a computer for power and to monitor and control this Arduino via the Arduino software's built-in terminal window*, a.k.a. Serial Monitor. A number of tests were conducted after placing the devices at different places in my home (with varying distances between the devices). These tests were conducted to see how the distance between the devices affected the delay and throughput.

3.4.2 Successfully PING (Mobile)

These tests repeated the same tasks as above, but now the devices were placed in cars and measurements were made in a number of different environments. In these tasks, I sat next to a driver in a car where I have placed the router, a computer running Wireshark, and an Arduino. Another car followed behind us with a driver. In this second car an access point was attached to either a computer or an Arduino. Measurements were made in different environments (specifically city driving, driving through a tunnel, and countryside driving). The distance between the cars was logged while collecting these measurements. The router was powered from either the cigarette lighter socket (using the adapter shown in Figure 3-3) or an external 12 VDC battery (as shown in Figure 3-4). An initial set of measurements was made to ensure that the cigarette lighter socket provide a voltage of 12 V or less in order to avoid any potentially damage to the router and to ensure that sufficient voltage and power were provided to power the router.

Figure 3-3: Cigarette lighter socket adapter, direct voltage, no conversion

* The Hypertem program was used for this monitoring.

Figure 3-4 **Cigarette lighter socket adapter, conversion to 5V 1A**

3.4.2.1 Wireless PING between two computers

In this test a router was placed in the front car with one of the computers with Wireshark installed, while the following car had an access point connected with a computer. I this test I will perform a simple ping command in different driving environments. The configuration of the equipment used for this test is shown in Figure 3-5.

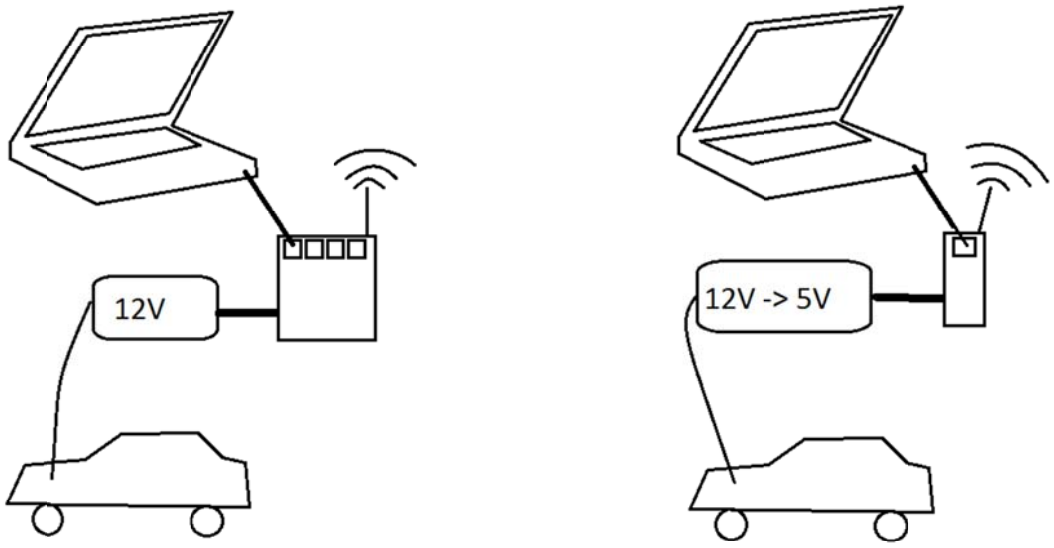


Figure 3-5: **Setup of computers in two cars**

3.4.2.2 Wireless PING between two Arduinos

This configuration is similar to the setup above with two computers. However, in the following car (i.e., the car with the access point) an Arduino was connected to the access point instead of a computer being connected to this access point. In the car with the router, an Arduino was connected to another of the router's ports. The computer used Wireshark to monitor the traffic throughout the entire test.

3.4.3 Send text messages (Stationary)

In this test a number of different types of data packets were sent between the Arduinos. The first question was whether text messages would be received in the expected order and what the transmission delay was. This task was only performed once.

3.4.3.1 Wired text sending between two Arduinos

In this test text messages containing simple sentences or block of words were sent to check how they would be received by the other device.

3.4.3.2 Wireless text sending between two Arduinos

In this test text messages were send to check if every letter in a sentence was received as expected, i.e., if there were any errors in the received messages.

3.4.4 Send simple signals (Stationary)

This set of test concerns signals from future sensors and systems that would need to be sent between the Arduino cards. In the tests with the LEGO® models, I use only one type of sensor (a UltraSound Distance Radar (USDR) sensor) that was placed on the second vehicle in the convoy. All of the signals sent between Arduinos are encrypted/decrypted by the router and Accesspoint, as the Arduino's Ethernet shield does not provide encryption. When setting up the connection, the only requirements for the code was to specify the MAC-addresses, IP-addresses, and port-numbers of the end points of the communication. The access point supports WPA2-PSK [AES] encryption, while the router supports WPA/WPA2 Mixed encryptions.

3.4.4.1 Wired connection between two Arduinos

The USDR sensor, a motor, and a servo motor are connected to the Arduinos and both of the Arduinos are connect by Ethernet cables to the router. The signals sent included output from the USDR and input to the servo motor and a regular motor.

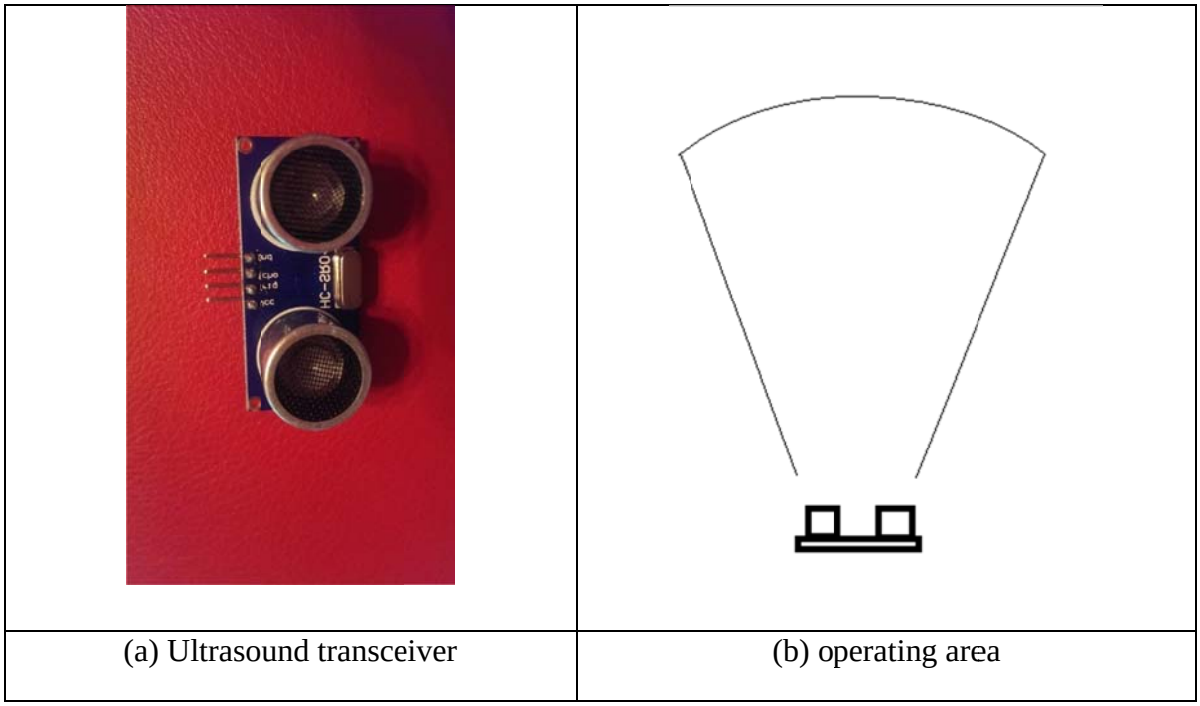
3.4.4.2 Wireless signal sending between two Arduinos

This task is similar to the above, but one of the Arduinos is connect it to the access point and the distance between the access point and router is varied. In this task I try to send the request signal and leave signal from the "client" side of the convoy (the client is connected to the access point). This organization was chosen because the connection to the convoy leader should be established by the drivers of the model. The client model asks to join the convoy by sending a "join convoy" request to the master driver. The master driver will either accept or decline this request. If a Wi-Fi connection is established, then the client should send a "leave convoy" request when client wish to take over driving their vehicle.

3.4.5 Send simple signals (Mobile)

Here we describe the setup of the Arduinos with each LEGO® motor, USDR sensor, and the servo motor. The servo motor is used to control steering of the models. This setup was first done for each part individually and then more test components were added if the previous test worked as expected. This divide-and-conquer technique saved time during the test phase, since if the results are not as expected then troubleshooting could be done. Figure 3-6 shows the USDR sensor and the measuring area. The angle is quite narrow and the range is between 3 – 300 cm. Figure 3-7 and Figure 3-8 show the setup of the two models and how they are powered when testing simple signals. Figure 3-9 shows the two models, each with an Arduino. The front model is the "master" and carries the router which requires the Arduino to have an Ethernet shield attached. The rear model is the "slave" and it carries a NETGEAR® Accesspoint and the LEGO® motor which requires the Arduino to have both an Ethernet shield and a motor shield attached to it. Both models have the servo motor attached to the Arduino to control the steering of the front wheels. Additionally, the rear model has an USDR sensor attached in front of the steering wheels.

As mentioned above, the Arduino itself does not provide any encryption when using the Ethernet shield. Although the router and access point use WPA2 encryption, the programs that runs in both Arduinos use special characters to make sure the correct data is sent. The code uses characters to indicate the purpose of the value, for example 'A' to accept convoy, 'R' to request convoy, 'M' for motor speed data, 'S' for servo degree data, and 'E' for end character of a data value. This encoding was used because the data packets that are sent from the Ethernet shield are sent as a one byte packet. They're sent this way due to the implementation of the Ethernet send methods, which uses the Arduino's main sending class and that class is implemented by sending byte by byte.



(a) Ultrasound transceiver

(b) operating area

Figure 3-6: **Ultrasound distance radar sensor which can measure about 2-400 cm in a 15 degree wide operating area. [36]**

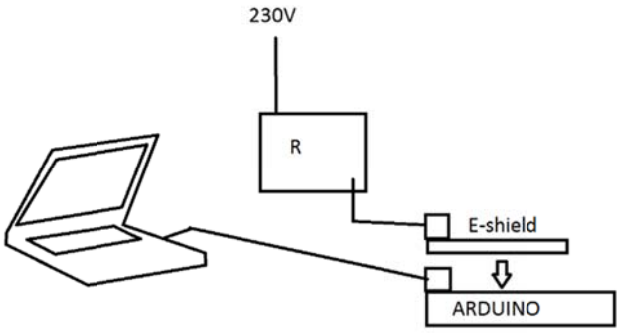


Figure 3-7: **Setup of front model with router R. Arduino is powered by computer via the USB interface.**

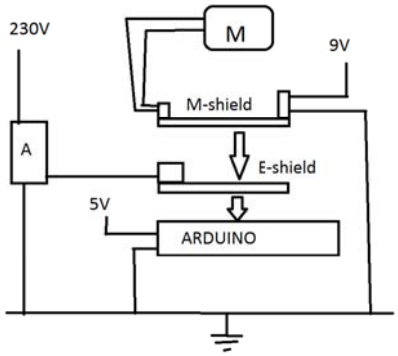


Figure 3-8: **Setup of rear model with components (such as access point A and motor M). The components are powered by three different power sources.**

Figure 3-9: **Setup of simple signals test. Front model with router and servo. Rear model with access point, servo motor, USDR sensor, and motor. Figure 3-7 and Figure 3-8 shows the blueprints for each of these.**

3.4.5.1 USDR signal

The USDR sensor will be attached to the front of the models to estimate the distance to an obstacle in front of it. This signal will be sent to the host from the client, and if the host has a similar sensor it will send its data to the client. In the experimental setup the front model did not have a USDR attached to it. In this experiment the lead vehicle did not have such a sensor. Only the rear vehicle sends information to the lead vehicle to inform it of what is happening. As the rear vehicle is equipped with a USDR sensor it can act if a sudden braking action occurs by the first vehicle.

3.4.5.2 Motor speed signal

The motor speed signal is used to accelerate or brake the model vehicle. Choosing the appropriate speed for this motor is the most important aspect of maintaining a convoy. This requires that the inter-vehicle signals be sent with a high priority in order to maintain the desired spacing between the vehicles, despite changes in motion of the lead vehicle.

3.4.5.3 Steering signal

An optional part of the model convoy was to make the following model car turn to follow the lead vehicle, thus it will simply follow the lead vehicle wherever it goes.

3.4.5.4 Request to Join/Leave convoy

Here I will try to implement the same test as was done in the stationary test, where a vehicle requests to join or leave the convoy. Hopefully some experience was gained from other tests with the models where I sent signals or text between Arduinos, because this task requires sending signals that will affect the final testing with the models.

Join and leave requests will utilize a handshaking protocol to ensure a safe and complete join or leave of the convoy. The enter-convoy request begins with the rear model (acting as a “slave”) sending a request to the master (i.e., the lead vehicle). The master can accept this convoy-request by sending a signal to the slave to confirmation that the request has been granted in which case the convoy is formed. A leave request is similar. The slave can send a leave request which the master accepts and replies to indicating that the convoy is terminated. However, the master can also disable the formation of a convoy by skipping the first handshake of a request from the slave. In this case, a convoy is not formed.

3.4.6 Handling lost connections

In reality V2V communications must be very reliable and lost signals and connections must be taken into account. Lost signals can be re-sent, but this requires a protocol to ensure that the signals are re-sent. Lost connections are worse since no signals can be sent and the communication protocol must start over.

For the tests in the two cars where I make pings I did not implement any handling of lost connections, but I simply repeated the tests over and over to see where and when connectivity was lost.

It is important to properly handle connection in the tests with the models. Because I used a wireless router with N-technology, the range of the router is quite large and it transmits quite strong signals. The results was that it was very difficult to have a connection loss since both models had to be very far from each other. A simulated connection loss would be sufficient to test what will happen to the “slave” model vehicle and how it should handle such a lost connection.

4 Analysis

This chapter summarizes the results of the different tasks described in the previous chapter. It will also analyze these results and draw parallels to currently developing systems.

The first section covers the basic tests of the components and power measurements. These results and their consequences are important for the next section's task to work properly and produce successful results. The second section concerns the results of the basic ping tests.

The third section gives results about the tests conducted with the LEGO® model cars and the implementation of the inter-vehicle communication system in these models.

4.1 Voltage measurements

As mentioned earlier, the components needed for this work were two Arduinos (each equipped with an Ethernet shield), two computers, one router, and one access point.

The initial tests were conducted at a desk or in nearby room, but mainly were conducted within the same building. The router, the access point, and the computers were connected to mains power. The Arduinos were powered from their attached computers via a USB-cable which supplies 5V. These Arduinos can either be powered via the USB host's 5 VDC or an external power ranging from 7-12 VDC (The onboard regulator is limited to an input voltage of 6-20 VDC).

For testing the device in real car, power was supplied via the car's cigarette lighter socket. This socket will supply ~12 VDC. The router's power input is labelled 12 VDC at 500 mA. I initially believed that it would not be a problem for the car to supply this power, but to ensure this I measured the voltage to be sure of the exact voltage when the car's engine is off, when the ignition is on, when starting the engine, when the engine is running, and when turning off the engine. The results of these measurements are shown in Figure 4-1. As can be seen in this figure the voltage, when the engine is on, is a little above 14 VDC which I was a little concerned about since the router expected to be supplied with 12 VDC. To address these concerns I bought an adapter which could be set to different output voltages including 12 VDC, when the input voltage is 12-24VDC. Measurements of the adapter's output voltage during same engine states is shown in Figure 4-2. In both figures the dip in the voltage occurs when the engines starts. To be more precise, the dip occurs exactly when the starting motor requires a large amount of current to force the cylinders to move inside the petrol engine. This does not occur when first turning the ignition key, but occurs shortly after this. This high demand for power is provided by drawing power from the battery, however this causes the voltage to drop for just a second because the alternator of the petrol engine then starts charging the battery by supplying it with about 14V. Since the cigarette lighter socket is directly connected to the battery's poles, this output is also 14V. When I connected the router to the 12V output adapter and ran the car through the same engine states as earlier, I noted that the voltage drop during engine start did not affect the router, thus the router stayed on during all of the different engine states.

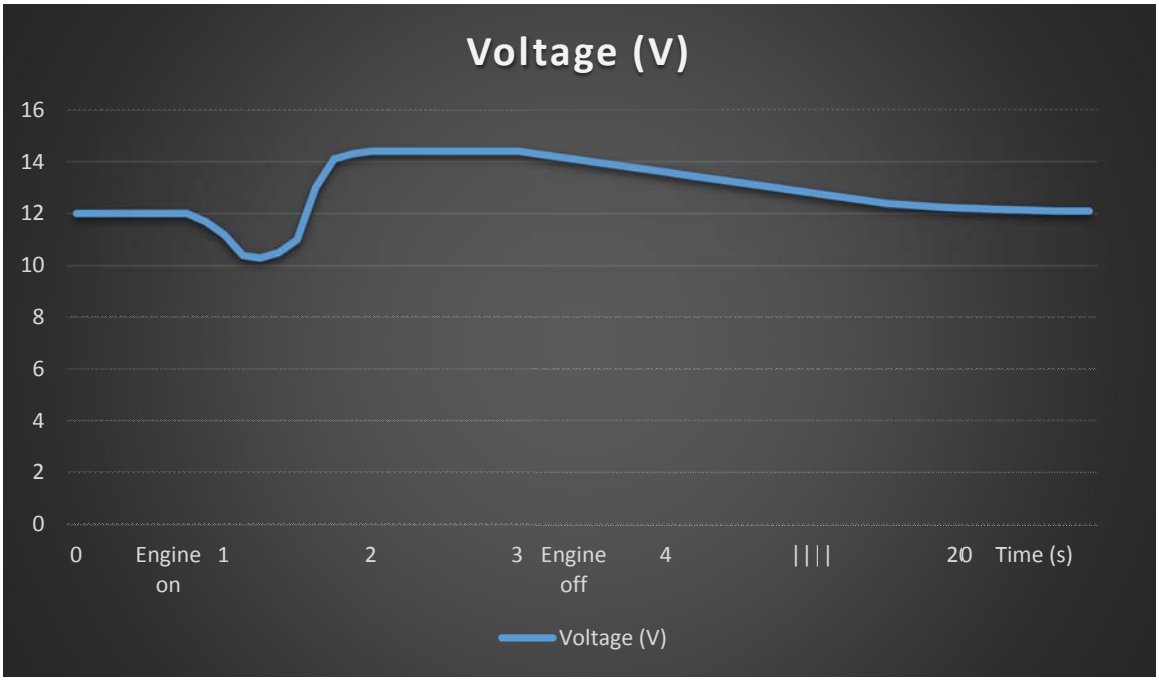


Figure 4-1: **Voltage measurements of the car's cigarette lighter socket during a short time period.**

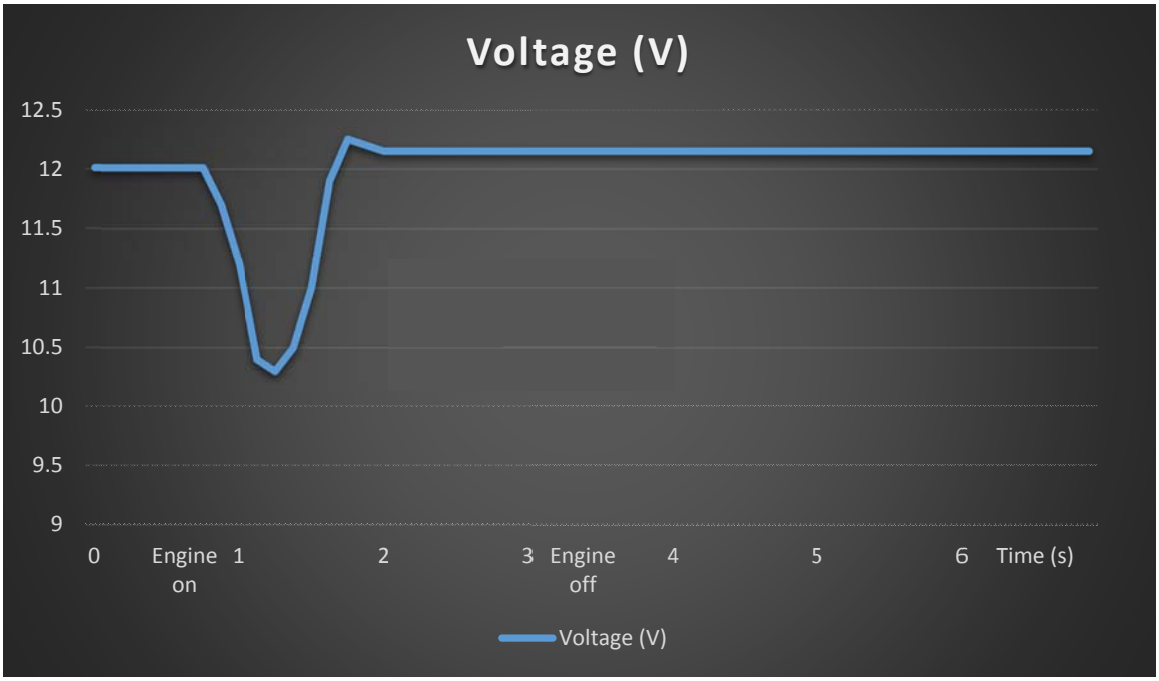


Figure 4-2: **Voltage measurements of same socket but this time with a converter. It gives a steady 12V even when the engine is on**

The access point requires 1 A at 5 VDC. Therefore, I used an adapter that could take 12-24 VDC and output 5 VDC.

4.2 PING tasks

This section covers the PING tasks that were performed as described above in Sections 3.2.1-3.2.2. The results are shown as Wireshark screenshots and the PING request time and PING reply time are analyzed. These lines of the screen capture where PING requests and PING replies occur utilize the ICMP protocol. The protocol of each packet is indicated in the fifth column.

4.2.1 Results from PING tasks

Pinging from the computer using the built-in Microsoft Windows ping tool via the command prompt generated four pings. Figure 4-3 shows these ping requests and replies along with other relevant information. Figure 4-7 shows the results when 10 pings requests are send by the Arduino (along with the same relevant information). Wireshark shows the same information for both tests, along with a lot of extra information. For these tests, the ping request times and replies times (shown in units of seconds in the second column of both figures) are the information needed to compute the delay between a request and a response. The statistics of the wired ping from computer to router are shown in Table 4-1. These statistics were computed over 4 pings.

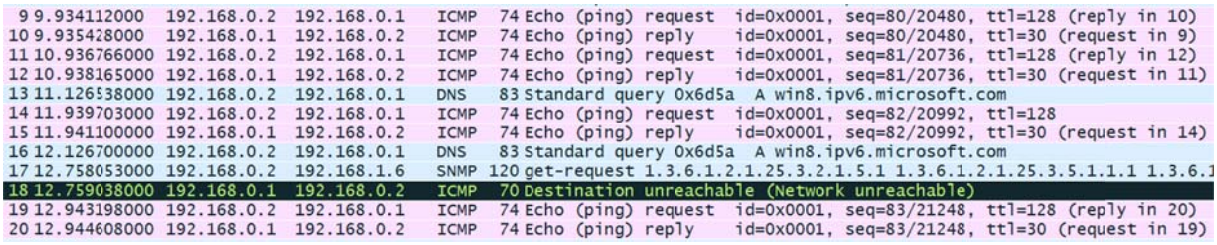


Figure 4-3: Ping requests and replies between computer and router

Table 4-1: Statistics of the wired ping from computer to router

Average time	0.00138s
Minimum time	0.00132s
Maximum time	0.00141s

Figure 4-4 shows the case for a ping request being sent by one computer and the reply sent to the requesting computing. The resulting delay statistics are shown in Table 4-2. These statistics were computed over 4 pings.

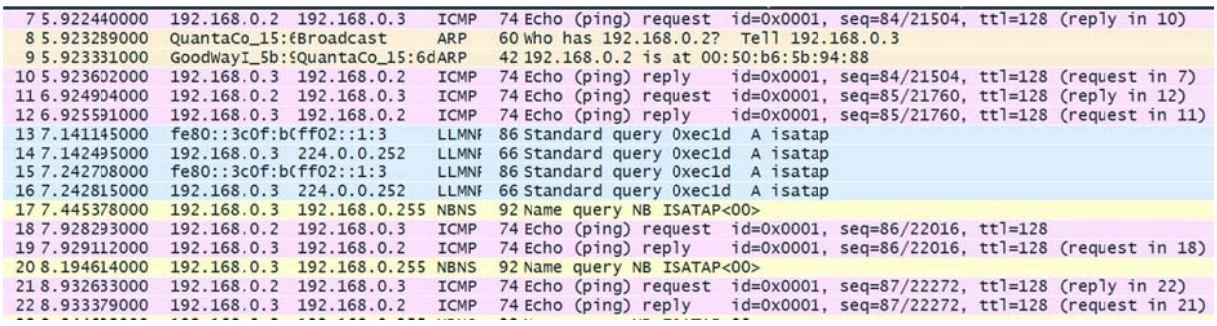
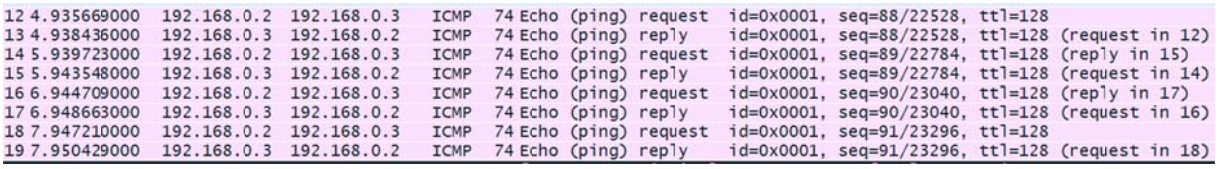


Figure 4-4: Ping requests and replies between two computers via router

Table 4-2: Statistics of the wired ping from computer to computer

Average time	0.00085s
Minimum time:	0.00069s
Maximum time:	0.00116s

Figure 4-5 shows the pings between a computer connected to the router and a computer connected to the access point (wireless PING between two computers 1st case). The statistics of these pings are shown in Table 4-3. These statistics were computed over 4 pings.



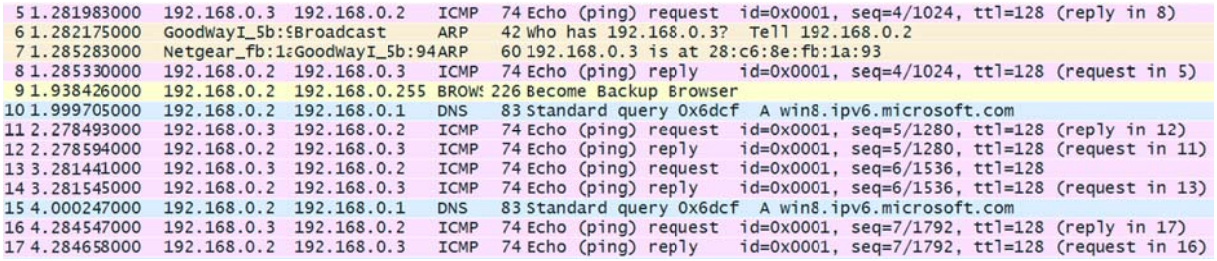
12	4.935669000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=88/22528, ttl=128
13	4.938436000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=88/22528, ttl=128 (request in 12)
14	5.939723000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=89/22784, ttl=128 (reply in 15)
15	5.943548000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=89/22784, ttl=128 (request in 14)
16	6.944709000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=90/23040, ttl=128 (reply in 17)
17	6.948663000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=90/23040, ttl=128 (request in 16)
18	7.947210000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=91/23296, ttl=128
19	7.950429000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=91/23296, ttl=128 (request in 18)

Figure 4-5: **Computer connected to router sends ping requests to computer on access point**

Table 4-3: **Wireless PING between two computers 1st case**

Average time	0.00344s
Minimum time:	0.00277s
Maximum time:	0.00395s

Figure 4-6 shows the pings between a computer connected to the router and a computer connected to the access point (wireless PING between two computers 2nd case). The statistics of these pings are shown in Table 4-4. These statistics were computed over 4 pings.



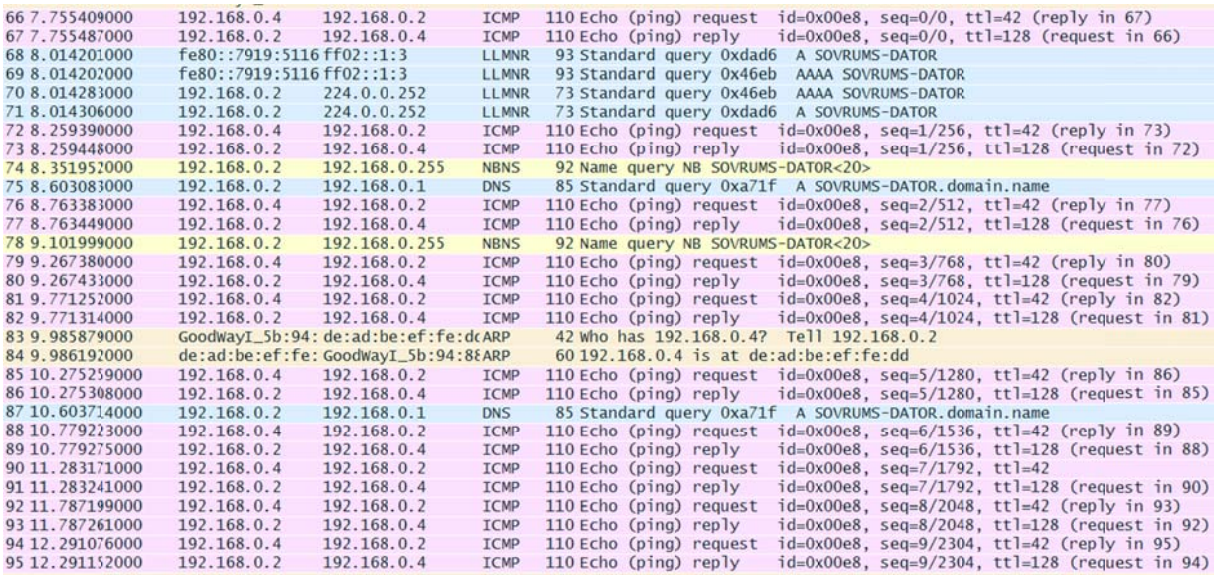
5	1.281983000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=4/1024, ttl=128 (reply in 8)
6	1.282175000	GoodWayI_5b:9	Broadcast	ARP	42	Who has 192.168.0.3? Tell 192.168.0.2
7	1.285283000	Netgear_fb:1a	GoodWayI_5b:94	ARP	60	192.168.0.3 is at 28:c6:8e:fb:1a:93
8	1.285330000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) reply id=0x0001, seq=4/1024, ttl=128 (request in 5)
9	1.938426000	192.168.0.2	192.168.0.255	BROWS	226	Become Backup Browser
10	1.999705000	192.168.0.2	192.168.0.1	DNS	83	Standard query 0x6dcf A win8.ipv6.microsoft.com
11	2.278493000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=128 (reply in 12)
12	2.278594000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128 (request in 11)
13	3.281441000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=6/1536, ttl=128
14	3.281545000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) reply id=0x0001, seq=6/1536, ttl=128 (request in 13)
15	4.000247000	192.168.0.2	192.168.0.1	DNS	83	Standard query 0x6dcf A win8.ipv6.microsoft.com
16	4.284547000	192.168.0.3	192.168.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=7/1792, ttl=128 (reply in 17)
17	4.284658000	192.168.0.2	192.168.0.3	ICMP	74	Echo (ping) reply id=0x0001, seq=7/1792, ttl=128 (request in 16)

Figure 4-6: **Computer connected to access point sends ping requests to computer on router**

Table 4-4: **Wireless PING between two computers 2nd case**

Average time	0.00092s
Minimum time:	0.0001s
Maximum time:	0.00335s

Arduino make the ping request, computer responds with a ping reply. Figure 4-7 shows the pings Arduino sends ping requests to computer via router. The statistics of these pings are shown in Table 4-5. These statistics were computed over 10 pings. (Very fast pings, possible due to Arduinos simple OS, few routines to handle by the microcontroller?)



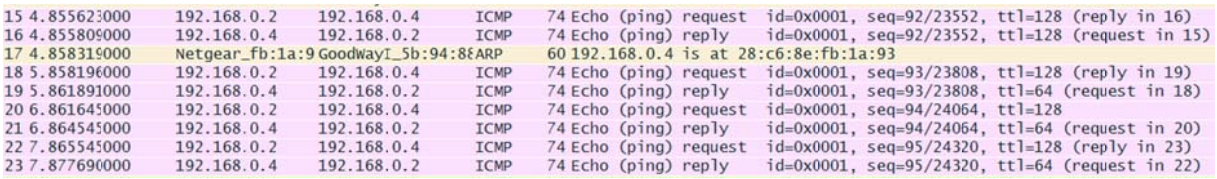
66	7.755409000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=0/0, ttl=42 (reply in 67)
67	7.755487000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=0/0, ttl=128 (request in 66)
68	8.014201000	fe80::7919:5116	ff02::1:3	LLMNR	93	Standard query 0xdad6 A SOVRUMS-DATOR
69	8.014202000	fe80::7919:5116	ff02::1:3	LLMNR	93	Standard query 0x46eb AAAA SOVRUMS-DATOR
70	8.014283000	192.168.0.2	224.0.0.252	LLMNR	73	Standard query 0x46eb AAAA SOVRUMS-DATOR
71	8.014306000	192.168.0.2	224.0.0.252	LLMNR	73	Standard query 0xdad6 A SOVRUMS-DATOR
72	8.259390000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=1/256, ttl=42 (reply in 73)
73	8.259448000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=1/256, ttl=128 (request in 72)
74	8.351952000	192.168.0.2	192.168.0.255	NBNS	92	Name query NB SOVRUMS-DATOR<20>
75	8.603083000	192.168.0.2	192.168.0.1	DNS	85	Standard query 0xa71f A SOVRUMS-DATOR.domain.name
76	8.763383000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=2/512, ttl=42 (reply in 77)
77	8.763449000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=2/512, ttl=128 (request in 76)
78	9.101999000	192.168.0.2	192.168.0.255	NBNS	92	Name query NB SOVRUMS-DATOR<20>
79	9.267380000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=3/768, ttl=42 (reply in 80)
80	9.267433000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=3/768, ttl=128 (request in 79)
81	9.771252000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=4/1024, ttl=42 (reply in 82)
82	9.771314000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=4/1024, ttl=128 (request in 81)
83	9.985879000	GoodWayI_5b:94:	de:ad:be:ef:fe:dc	ARP	42	Who has 192.168.0.4? Tell 192.168.0.2
84	9.986192000	de:ad:be:ef:fe:	GoodWayI_5b:94:8E	ARP	60	192.168.0.4 is at de:ad:be:ef:fe:dd
85	10.275259000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=5/1280, ttl=42 (reply in 86)
86	10.275308000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=5/1280, ttl=128 (request in 85)
87	10.603714000	192.168.0.2	192.168.0.1	DNS	85	Standard query 0xa71f A SOVRUMS-DATOR.domain.name
88	10.779223000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=6/1536, ttl=42 (reply in 89)
89	10.779275000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=6/1536, ttl=128 (request in 88)
90	11.283171000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=7/1792, ttl=42
91	11.283241000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=7/1792, ttl=128 (request in 90)
92	11.787199000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=8/2048, ttl=42 (reply in 93)
93	11.787261000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=8/2048, ttl=128 (request in 92)
94	12.291076000	192.168.0.4	192.168.0.2	ICMP	110	Echo (ping) request id=0x00e8, seq=9/2304, ttl=42 (reply in 95)
95	12.291152000	192.168.0.2	192.168.0.4	ICMP	110	Echo (ping) reply id=0x00e8, seq=9/2304, ttl=128 (request in 94)

Figure 4-7: **Arduino sends ping requests to computer via router**

Table 4-5: **Statistics of Arduino sending ping requests to computer via router**

Average time	0.00006s
Minimum time	0.00005s
Maximum time	0.00008s

Computer makes the ping request, Arduino replies. Figure 4-8 shows the pings when a computer sends ping requests to Arduino via router. The statistics of these pings are shown in Table 4-6. These statistics were computed over 4 pings. (Very slow pings compared with previous statistics, computer handles more processes)



15	4.855623000	192.168.0.2	192.168.0.4	ICMP	74	Echo (ping) request id=0x0001, seq=92/23552, ttl=128 (reply in 16)
16	4.855809000	192.168.0.4	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=92/23552, ttl=128 (request in 15)
17	4.858319000	Netgear_fb:1a:9	GoodWayI_5b:94:8E	ARP	60	192.168.0.4 is at 28:c6:8e:fb:1a:93
18	5.858196000	192.168.0.2	192.168.0.4	ICMP	74	Echo (ping) request id=0x0001, seq=93/23808, ttl=128 (reply in 19)
19	5.861891000	192.168.0.4	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=93/23808, ttl=64 (request in 18)
20	6.861645000	192.168.0.2	192.168.0.4	ICMP	74	Echo (ping) request id=0x0001, seq=94/24064, ttl=128
21	6.864545000	192.168.0.4	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=94/24064, ttl=64 (request in 20)
22	7.865545000	192.168.0.2	192.168.0.4	ICMP	74	Echo (ping) request id=0x0001, seq=95/24320, ttl=128 (reply in 23)
23	7.877690000	192.168.0.4	192.168.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=95/24320, ttl=64 (request in 22)

Figure 4-8: **Computer sends ping requests to Arduino via router**

Table 4-6: **Wireless PING between Arduino and computer 2nd**

Average time	0.00473s
Minimum time	0.00019s
Maximum time	0.01215s

Table 4-7 shows the case of a wireless PING between two Arduinos (Stationary case). The Wireshark screen capture for this case is shown in Figure 4-9 and the statistics in Table 4-8.

Table 4-7: PING times (in milliseconds) from varying distances. The tests with 10m and 20m have also obstacles such as walls in between

Ping #	Distance 1m (ms)	Distance 10m (ms)	Distance 20m (ms)
1	609	617	620
2	2	48	9
3	2	39	7
4	2	31	10
5	2	16	3053
6	2	56	8
7	2	13	2062
8	3	10	1024
9	3	63	7
10	3	2033	1058

Wireless PING between computer and access point (mobile)

IP setup:

192.168.0.1 = Router (front car)

192.168.0.2 = Accesspoint (rear car)

192.168.0.3 = Computer (front car)

192.168.0.4 = Arduino#1 (front car)

192.168.0.5 = Arduino#2 (rear car)

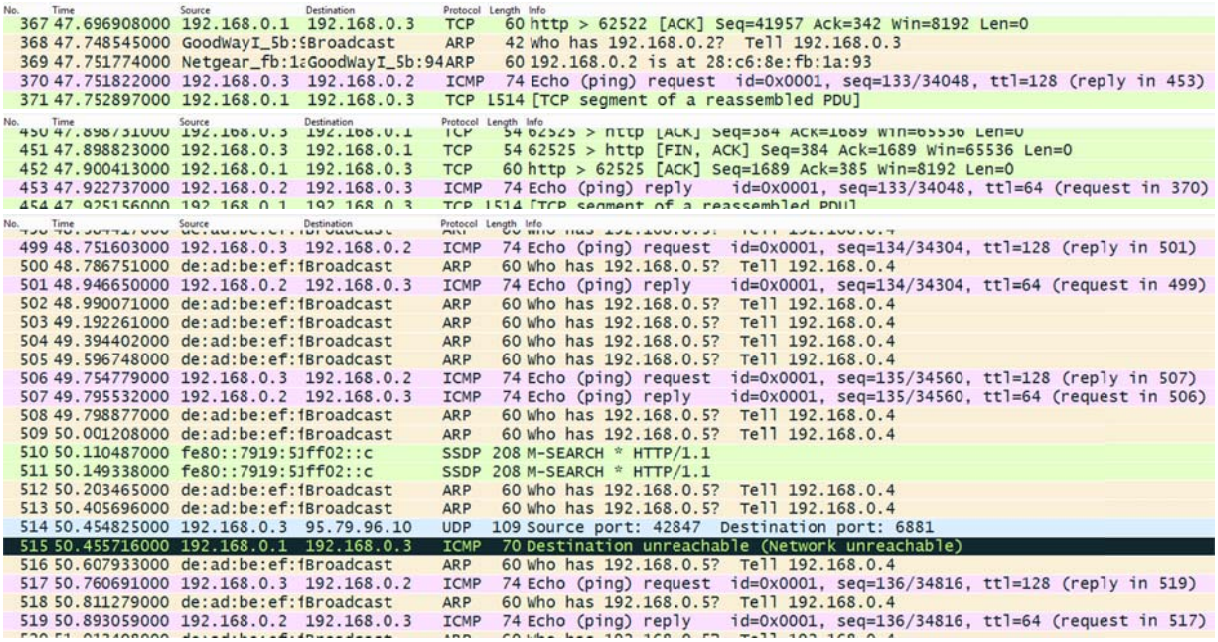


Figure 4-9: Wireshark capture of a ping from a computer in the front car to the access point in the rear car

Table 4-8: Statistics of a ping from a computer in the front car to the access point in the rear car

Average time	0.13477s
Minimum time	0.04075s
Maximum time	0.19505s

Table 4-9 shows the ping times when driving about 30-40 km/h on normal road with a distance about 10-15 meters between cars, while Table 4-10 shows the ping times when driving about 50-60 km/h on normal road with distance about 20-30 meters between cars. In the third test, the front car accelerated very fast from the rear car which resulted in the fourth ping being lost and at time the distance was about 50 m.

Table 4-9: Ping times when driving about 30-40 km/h on normal road with distance about 10-15 meters between cars

Test number	1	2	3	4	5
Average time (s)	0.00965	0.00687	0.00301	0.01535	0.02436
Minimum time (s)	0.00374	0.00216	0.00172	0.00313	0.00562
Maximum time (s)	0.01720	0.02028	0.00502	0.03596	0.04876

Table 4-10: Ping times when driving about 50-60 km/h on normal road with distance about 20-30 meters between cars. In the third test the front car accelerated very fast from the rear car which resulted in the fourth ping got lost and at time the distance was about 50 m.

Test number	1	2	3	4
Average time (s)	0.11430	0.28424	0.01505	0.00287
Minimum time (s)	0.00374	0.06933	0.00590	0.00211
Maximum time (s)	0.21676	0.48417	0.03442	0.00369

4.2.2 Analysis of PING results

In the mobile implementation of the ping tasks, the setup in the two cars had only one goal initially. It was to enable two Arduinos be able to ping each other under different conditions while driving. However, when I setup everything in the cars I decided to make them ping a first time before starting driving, just to ensure that they were able to ping each other. This is where I encountered a problem. I could not communicate between the two Arduinos even when the cars standing still near each other. I could see on the access point that it had associated with the router by a green Light Emitting Diode (LED). However, I could not understand why the Arduino could not successfully ping the Arduino connected to access point. I asked the drivers to begin driving even though I had no connectivity between Arduinos. To avoid wasting time I changed my goal to simply be able to ping the access point in the other car, thus I had some sort of V2V communication – although it was not the initially desired communication. This alternate goal was achieved, and the ping response time results shown above prove that distance and perhaps speed affect the pinging times.

A few hours after this task was done, I tried to troubleshoot the Arduinos and realized that the Arduino in the rear car attached to the access point did not run its program, therefore it did not have an IP-address - hence this Arduino was not visible on the network.

One driving test environment I wanted to try was to drive through a tunnel while pinging. Unfortunately, there was no nearby long tunnel. A future tests should try to ping between Arduinos when in a tunnel to check whether the tunnel makes a difference.

4.3 V2V implementation in model

This section covers the last testing phase. In this test I implemented a simple V2V or “Inter-Model communication” using Wi-Fi to communicate and used a USDR sensor to estimate the range between the vehicles and obstacles. I began by testing the Wi-Fi communication for simple functions such as steering and setting a specific motor speed. Then I used the USDR sensor to measure distances to

obstacles. When those two tests were completed I combined these functions to integrate both USDR sensor readings and Wi-Fi communication. This enabled me to steer both models in same direction or running each motor at same speed.

As mentioned in section 3.4.5 the Ethernet shield uses one byte data packet to send bytes to another Ethernet shield. This became a problem when I wanted to send values longer than one digit (as ASCII charcters). While I could use letters as commands in the protocol, I recognized that data values for both servo and motor could have values greater than one digit – I need to support two or three digit values. Details of how this was done are described in the following subsections.

4.3.1 Wi-Fi communication testing

This first step was to test Wi-Fi communication between two Arduinos, one connected to the router via Ethernet shield and the other connected to the access point via an Ethernet shield. For these tests I made the Arduinos PING each other, connect and disconnect to/from each other, and connected every component I had (motor, servo motor, and USDR sensor). When initially connecting all of the components I had not yet written code to support each of them, but I wanted to see if there was any disturbance while these components were connected. Fortunately, this initial testing was success and there were no errors when the Arduino connected to each other, even when programs begun running. The first problem I encountered regarding Wi-Fi was that the behavior of the models when the Arduinos connected to each other as this behavior seemed to differ from time to time when these device were started. On the controllers (described further in Section 4.3.6), there are two LEDs indicating different states during the testing process. These LEDs indicated when the Arduinos in the models were connected to each other and indicated the “handshakes” while joining/leaving a convoy (handshakes = requesting to join a convoy & the reply indicating that the vehicle was accepted to join the convoy).

4.3.2 USDR sensor testing

In this testing, I ran through several tests to gain knowledge how the USDR sensor worked. The first tests were mainly to see how sensitive the sensor was depending on the measured angle and how far away the obstacle was. One result from the first test was that the floor had a negative effect on the measurements. Because of that, I increased the horizontal angle of the sensor so that it was pitched up a bit, about 20 degrees. After this adjustment, the sensor measured ranges perfectly when testing it, although these test were component tests using only the sensor and some simply code (specifically code to realize a serial connection with the computer). Using this code, I could monitor the output of USDR sensor. One observation was that the farther away from the sensor the obstacle was within the measured area, the longer it took for the pulse to return. This was as expected. However, the signal was more unreliable as the distance to the obstacle increased. This was not good result because the signal values could vary between 0-300, and if there was no obstacle within 300 cm, it could cause a very long delay, since the sensor was waiting for the returning pulse echo.

4.3.3 Signals

Combining Wi-Fi communication, USDR sensor, and the motor was very straightforward and everything worked as expected. I wrote code for one task or combined two or three tasks. If any component acted unexpectedly, it was not hard to locate the error in the code. The main tasks where to ensure that the rear model would not crash into the front model, to have both models steer in same direction, and ensure that the rear model would properly apply its emergency brake if for some reason the model in front stopped. All these tasks were part of the main goal of this thesis project. When testing the individual components (USDR sensor, servo motor, motor), they ran well when sending data or receiving commands over Wi-Fi. However, when they were connected to each other different problems began appear.

4.3.3.1 Keeping distance to front vehicle

The first goal of the test with the two models was to see if the models could maintain a given separation (i.e., for the following vehicle to remain within a fixed range of the lead vehicle).The goal was not completely accomplish. Once every component on the models were electrically connected I began testing, but I noticed a 30 second delay before the front model sent its data to the rear model. It was very strange that this delay appeared because the requesting and accepting of convoy signals were not delayed (as confirmed by the LEDs). However, once the "accepting convoy" signal was sent the delay appeared. It was not easy to locate where in the code this delay originated, since everything was sent wirelessly between the models. I began by searching the web for delay issues concerning the Arduino Ethernet shield. However, this seemed to be an uncommon problem, although the issue seemed to be related to Ethernet shield rev.2 but I used rev.3. Since I did not believe that the Ethernet shield itself was the source of the delay I experimented with the code. After a while, I found the cause of the delay and it indeed had nothing to do with Ethernet shield. It turned out that the delay was caused by the USDR sensor. This sensor is used to measure the distance from the following model to model in front of it.

Figure 4-10 shows a code snippet of the USDR sensor's calculation. This code caused the 30 second delay. However, this was very strange since the code only has two intentional microsecond long delays plus the measured delay saved in variable "dur". During a test where I wanted to confirm that this code snippet was actually causing the delay I clocked the system time, the function millis() which returns the system time after startup in milliseconds, before the code snippet and right after. Then to subtract the times I got the time it took to compute the code snippet. If the sensor was working correctly and since the code variable "dur" could vary between 116 – 23200 (speed of sound 340m/s is equal to 29 µs/cm and measureable distances is 2-400 cm, gives total distance of 4-800 cm which is 116-23200 µs), total time is maximum 23200 + 2 + 10 µs. This is equal to about 0,023 seconds which is nowhere near the 30 seconds error delay. My conclusion is that the delay must be due to the physical setup of the USDR sensor. The USDR sensor is mounted on a small breadboard. This board is connected via wires to the Arduino, a power source, and the system ground. Perhaps the observed delay is caused by poorly connected wires leading to some of the ranging echoes not being detected or perhaps it could be because of the total delays in loop is not proper for what the sensor is required. Apparently the sensor requires a minimum pulse length of 10µs according to [37] but minimum pulse length of 5µs according to [38] and a minimum delay of 60 ms between two pulses. I wrote my code according to Arduinos website which used the 5µs but I did not use a minimum 60ms delay between two pulses. These settings may be the reason why USDR sensor is behaving incorrectly.

```
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long dur = pulseIn(echoPin, HIGH);
long distance = microsecondsToCentimeters(dur);
```

Figure 4-10: Code snippet of calculating distance to obstacle.

This problem caused me to be unable to use the USDR sensor together with wireless data transfer. Since the USDR sensor was supposed to help the system initiate emergency braking if necessary, I did not see a major problem since the models were not going at high speed (as their speed was limited to 0.54 km/h, for details see Section 4.3.6). Given these low speeds, if the rear model crash into the front model, it would not be such a great impact. On the first try, I also noticed that the rear model was travelling a bit faster than the front model, even though each motor had been set to the same speed.

This was probably because the front model had a different weight and balance. Note that most of the weight was due to the router, which was placed right above the rear wheels, and the heavy battery, which was placed near the rear wheels. Due to these placements, the model was very heavy in its rear end, which could be seen on the suspensions (see Figure 4-10). Therefore, the motor of the front model might have had more resistance than the rear model, where the weight was more balanced in the middle of the model (see Figure 4-11). Figure 4-12 and figure 4-13 show the rear axle in a closer view. The difference in weight balance cause the rear axle to bend more on the front model, shown in figure 4-12 than the rear model shown in figure 4-13.

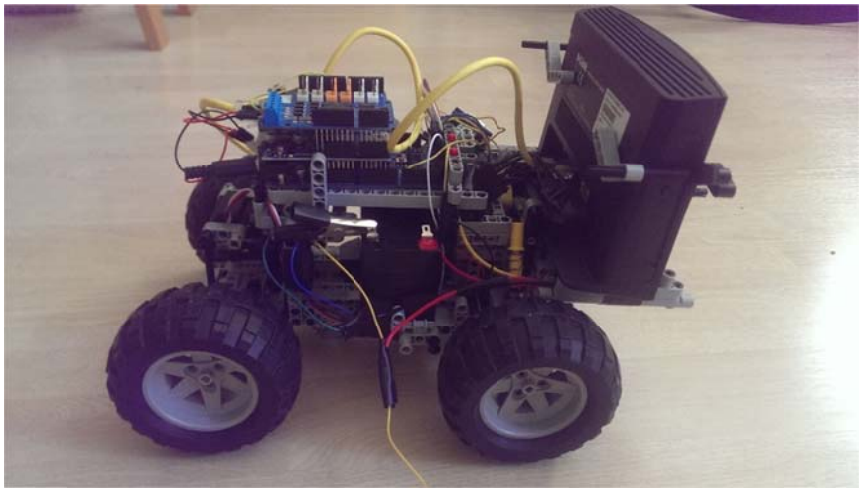


Figure 4-10: **Front model from the side. Leaning back because of the weight, the yellow suspensions on rear wheels are almost fully retracted.**

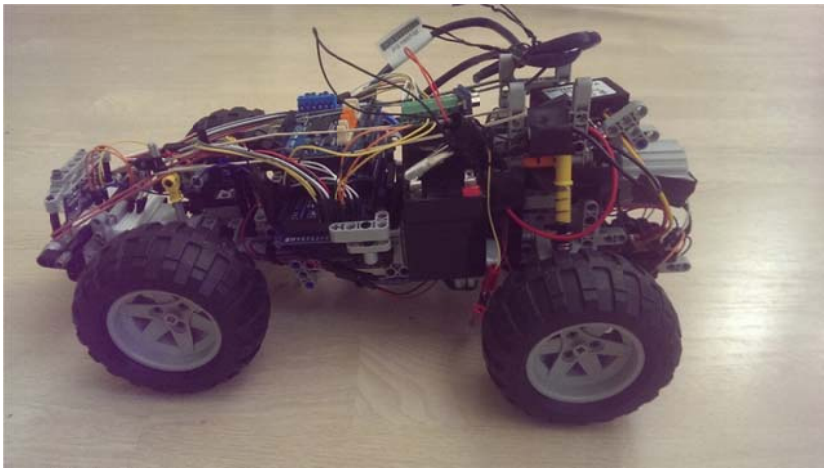


Figure 4-11: **Rear model from the side. Leaning a bit forward due to better weight balance so the suspensions are more extended than the front model.**

Figure 4-12: **The rear axle is horizontal due to the retracted suspensions, because of the weight.**

Figure 4-13: **The rear axle is not horizontal due to not fully retracted suspension, because of better balance of the weight.**

4.3.3.2 Keeping same direction as front vehicle

I programmed the Arduinos to run to be able to steer according to the input from the user of computer #1. Code listing 2.3* is the master's code and it gets the servo angle from the user of computer #1. The master then sends this same servo angle value to the rear model. As a result, both models turn to the same orientation, i.e., they steering in same direction. Code listing 2.4 was running on the rear model when this test was done.

* The code snippets are included in the appendix.

There is a big problem with this code when looking at the steering. With this code algorithm the steering on the following vehicle is applied directly with no intentional delay which causes the vehicles to choose different paths, as shown in figure 4-14.

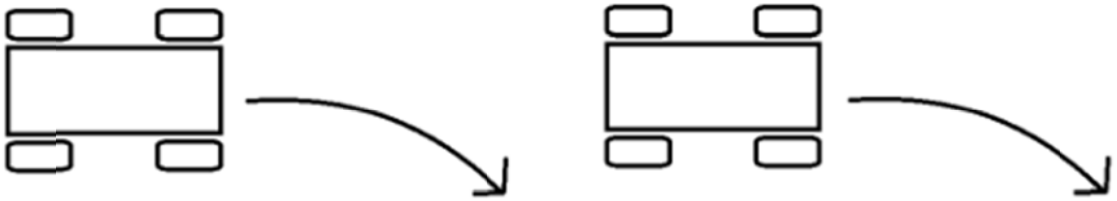


Figure 4-14: Steering algorithm problem

I examined the code for the following vehicle to see if there could be a solution to this problem. I did not have time to develop a solution in the code, but the purpose was make the following vehicle drive forward a bit before applying the turning as showing in figure 4-15. To make the vehicle turn to the right path, it had to first drive forward a bit and then make the turn. However, since that did not depended on time, because the vehicles could drive different speed at different times, the algorithm had to depend on distance travelled. One possible solution is to measure the distance the rear wheels travelled, calculate the distance between the vehicles, and then apply the steering at the proper moment. The problem with this solution is that either the distance between vehicles should be as a static input which also means the distance must be same during all driving, hence each vehicle speed must be exactly same, or the USDR sensor must work properly and the distance can be dynamically varied depending on each vehicles' speed.

I tried to apply this solution by thinking of code algorithms. One was to have a buffer where every steering degree value and speed value should be inputted, but that would require a big buffer since the reading of the buffer would be large per second. If the distance between vehicles was large enough to make the delay in time say 3 seconds before the following vehicle should apply steering, that would require a lot of readings of the buffer during those 3 seconds.

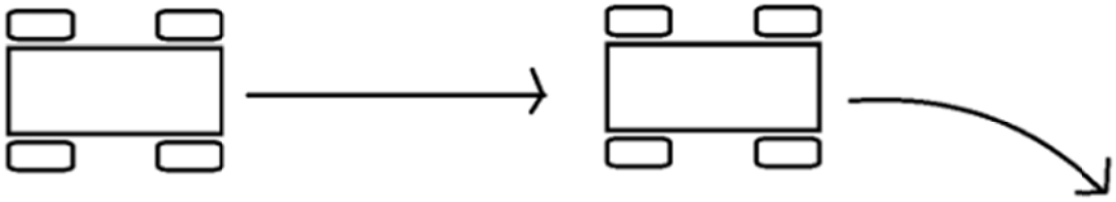


Figure 4-15: Steering algorithm as it should work

Unfortunately, the models are not exactly the same, which caused the tires of the front model to bump into the chassis in some cases. If the model were moving when this happens, the tire can touch the chassis causing the model to stop because of the tire's friction against chassis. Because the servo angle has a range of 0-180 degree, with about 90 degrees orienting the wheels straight forward, I had to limit the angles values to be between 40 and 140 degrees, in order to avoid the tires touching the chassis.

4.3.3.3 Emergency braking

The emergency braking system was a very important part of what I wanted to accomplish. However, because the USDR sensor failed to work properly (as described in Section 4.3.2) I chose to deactivate the sensor and therefore I could not complete the emergency braking system. The idea was that if the rear model vehicle continued moving toward the front model or was not slowing down sufficiently based upon wireless commands, the USDR sensor was to override the system and apply the brakes, thus preventing an impact with front model. Fortunately the speed, as I mentioned above, is very low thus the result of an impact would be insignificant and the rear vehicle would simply shove the front vehicle forward a little. As part of future work this functionality should be enabled and tested.

The following pseudo-code should prevent the rear model from crashing into the front model. The code first checks if convoy mode is active, as otherwise a user is directly controlling the model. In convoy mode, the distance between the vehicles is estimated using the USDR sensor and the speed of following model's motor is adjusted to maintain the desired separation. If the distance is 5 cm or less, then the motor is immediately stopped, otherwise the motor's speed is changed depending on how close the following vehicle is to the lead vehicle. If closer than 10 cm, then the speed is slowed down more rapidly, while if the separation is between 10 and 20 cm, then the speed is slowed down less rapidly. When the separation is greater than 20 cm, it is unnecessary to perform an emergency braking maneuver. The motor speed range is integer values between 0-255, however when increasing the values from 0 and up to 255, there is first a noise coming from the motor when the values are between 0 and about 30 and the motor is not moving. It is caused by the pwm which is currently changing the pulse width. Why the motor is not running at those values is also because it requires a minimum current flow for the axle to start moving, which is too low at low integer values, but at some value the current reaches the minimum edge and the axle starts to move. This edge is around integer value 30 in the range 0-255. This value is of course higher than if the motor were not connected to anything, now it has to start pulling the 1.8 kg vehicle, which resulting the value to be higher. I assume the motor axle starts move at perhaps integer value 20-25 when it is disconnected from everything.

```
if(convoy == active){
   if(USDR.measuredDistance < 5cm){
            speed = 0;
   }
   else if(USDR.measuredDistance < 10cm){
            speed-=5;
   }
   else if(USDR.measuredDistance < 20cm){
            speed--;
   }
}
```

4.3.4 Joining and leaving the convoy

This section was planned to describe the testing of joining and leaving a convoy. While not all of the desired functionality of the earlier stages of the project was achieved, I put a lot of time into the process of joining a convoy. I succeeded in making a second vehicle join a convoy. The procedure is quite simple; a wireless connection must be established to the lead vehicle. The user of the rear model presses button "B" on their keypad, a couple of LED light up, and the user of front model is ask to press button "A" (to accept the vehicle joining the convoy), and finally the convoy should be active. In convoy mode the front model sends it speed data and servo motor degree data to the rear model that should apply these values to its own motor and servo motor). This was a success and work almost every time. However, sometimes it did not work and I was unable to solve the problem other than a cold reboot. In

the last days of the project, I found a temporary solution was to press the reset button on each Arduino causing each system to reboot. Most of the time the connectionless state would not re-appear, although when it appears it was generally because the access point did not get any power or the Ethernet shield did not have time to initialize.

When things worked as they should, the access point and router were able to connect when the Arduinos were initialized. My first thought was that a connection between the access point and the router should be established when the user controlling the rear model pressed “B”, but that lead to errors (More about that in next section).

4.3.5 Lost connection

Before I built the models, I thought that the users of the models would determine when the connection between the vehicles was to happen. This would mean that the users of the vehicles would drive their models for a couple of seconds and then the rear model driver could press the request button “B” to to initiate the Ethernet connection and tell the rear model to send the request convoy message. Unfortunately, this did not working since the Ethernet shield behavior was a bit different each time the system started.

To solve this problem each Arduino waits for a specified time (30 seconds for the front model and 50 seconds for the rear model) to give the router and access point time to boot before attempting to connect to their first hop device (i.e., router or access point). This was necessary because otherwise the Arduino-to-Arduino connection was not always established. By waiting for the first hop device to start the wireless connection between router and access point is always established before there is an attempt at data transfer.

During testing there was never a lost connection between router and access point since the models were never far enough from each other that the wireless signal was too weak for communications between them.

However, I was made a routine to handle a lost connection just in case it should occur. This code checks if the Ethernet is connected and if convoy mode is active. The following pseudo code illustrates the idea:

```
if(Ethernet.connected(){
   if(convoy == active){
            speed = Ethernet.readSpeed();
            direction = Ethernet.readDirection();
            //apply motor speed and servo direction
   }
   else{
            speed = joystick_1.readY();
            direction = joystick_2.readX();
            //apply motor speed and servo direction
   }
}

else{
    speed = joystick_1.readY();
    direction = joystick_2.readX();
    //apply motor speed and servo direction
}
```

4.3.6 Power consumption and other model specifications

Both models used the same type of power source, a 12 V 1.3 Ah lead acid battery. I chose this type of battery because I thought it was the most suitable for my components. For example, the front model has a router which requires 12 V and 0.5 A. The Arduino can be supplied with either 5 V via USB or 7-12 V via the secondary power input. The motor shield on that Arduino should also be supplied with 12 V for the motor. Both models use the same sort of motor, a LEGO® Power Function XL motor which even though is made for 9 V supply can be supplied with 12V. According to Philippe Hurbain [39], who tested the motor with 12 V, found that it required 0.56 Amps and developed 14.5 N.cm at 214 rpm. Since both models used a 1:7 gearbox to get a smaller ratio and more power, the power and speed of the wheels were about 101.5 N.cm (= 1.05 Nm) and at 30.5 rpm drives the 297 mm circumference wheels about 9080 mm/min or ~9m/min. That is equal to 0.15 m/s or 0.54 km/h. The reason for using a gearbox to get a smaller gear ratio was that the models were quite heavy. They both weigh about 1.8 kg, which I thought was too much for the motor to be connected directly to the wheels.

With every component setup on the front model, it was clear that everything could be powered with 12 V that the 12 V 1.3 Ah lead acid battery could supply. The battery itself weighs 0.6 kg. On the rear model since the Arduino could supply 5V and ground, it could power the different components that needed 5V, such as the servo motor and USDR sensor.

In order for users to drive these models, I designed and implemented a controller. I chose to build two identical controllers to drive the two models. The final controller is shown in Figure 4-14 and Figure 4-15.

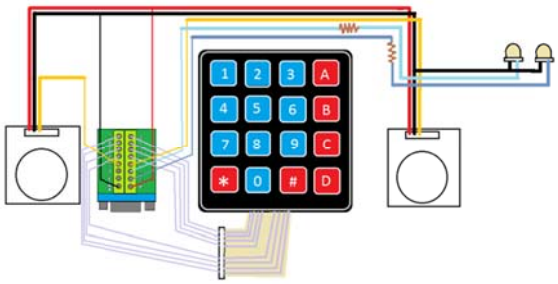


Figure 4-14: I/O connections of the controller

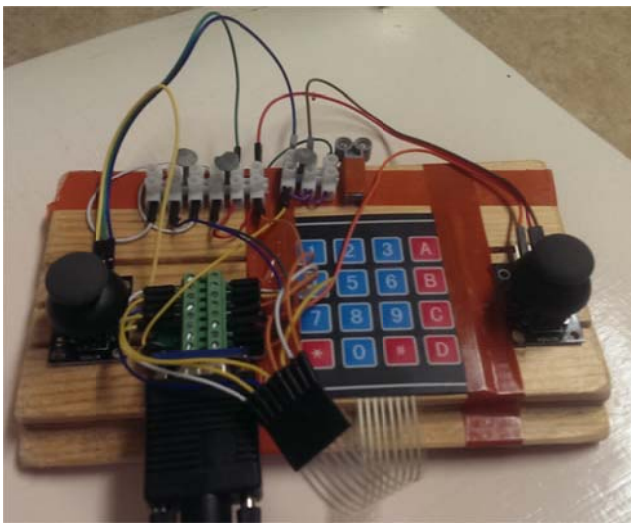


Figure 4-15: Final model controller

These controllers are connected to the models via a 16pin VGA cable. On the controller boards there are two joysticks, a 4x4 keypad, and two LEDs. The keypad connects directly to the Arduino. However, the joysticks require 5V. The LEDs are each connected via a 220 Ohm resistor to the Arduino, while the other side of the diode is connected to a common ground. I chose to use a VGA cable since it had an appropriate number of pins.

After these controllers were completed, I tested them. During this testing I noticed some issues when driving, such as unexpected delays and servos behaving weirdly. As I had powered every 5V component on the rear model using the Arduino's 5 V output I suspected that this did not provide enough current to power everything. To solve this, I redesigned the power system for the real model. Now every component that needs 5 V uses a 12V → 5V converter. This converter has a current limit of 3 Amps, which should be enough.

As the front model did not initially have a 12V → 5V converter I had to do install such as converter in order to use the controller.

Once everything worked after initial testing with the controller. I measured the current that was required by the whole system (the steering servo motor, the driving motor, the keypad, the LEDs turned on, the traffic between router and access point and of course the Arduino cards themselves) by connecting a multimeter between the 12 V battery and the input connectors to power system. The current varied between 0.3 – 0.6 Amps depending how much I used drive motor and steering servo motor. If we assume an average of 0.5 Amps at 12 V the battery would be sufficient to operate each vehicle for ~2.6 hours.

4.4 Met goals

The goal of this thesis project was to examine inter-vehicle communication and what the benefits and problems might occur when using this in convoys. During my work I defined some smaller sub-goals. These sub-goals made it easier to draw conclusions and then see if the initial goal was met.

These sub-goals were all related to inter-vehicle communication. First, ping was used to examine the connectivity between the router and access point when they were placed in two different cars. This enabled me to see how speed, distance, and environment could affect the WLAN radio-signals passing between two vehicles. Even though I ran into a problem when the one Arduino was not running properly, I could still use the access point, which this Arduino was attached to send and receive PING frames and examined the results of capturing this traffic. This sub-goal was met and then I could continue testing model vehicles by building up my own convoy.

When testing the model vehicles, there were several additional sub-goals defined. Several tests had good results, but there were a couple tests that did not succeed. A important result of this testing was I realized that there were only two important sub-goals with respect to successful convoy driving. First, the speed of the vehicle should be the same and secondly the direction that each vehicle should turn at each point along a path should be same, with only a small error allowed. After I made the decision to deactivate the USDR sensor, I was unable to test the emergency braking procedure. In reality, this functionality would be essential for a real convoy. The steering did not work as well as desired either because the servo motor was not sufficiently powerful or the attachment of the servo motor to the steering mechanism should be improved.

Using the two models I was able to establish that the working principle of using inter-vehicle communication via Wi-Fi to form a convoy would work. However, the combination of small delays, inaccurate motor speed, and errors in servo positions made the system's behavior less that I would have desired. Therefore, while I met the goal of a working convoy using inter-vehicle communication a lot improvements are required in future work before even this model system will behave as I desire.

The main goal was met from a certain perspective. Although I did not manage to create a complete working inter-vehicle communication system, I was able to solve many of the problems that came up and was able to identify some of the issues that need to be taken into account and solved in the future.

As a result of this project I gained a lot of knowledge about inter-vehicle communication and see why it could greatly benefit convoy driving. Given my experience, I know there are many problems that can occur and that they need to addressed with care since safety is a critically important factor of convoy driving.

Figure 4-16 to Figure 4-18 shows the final system during one trial. It can be easily seen that the models were not follow exactly the same path when driving (the front model driver was driving and the rear model was simply following this lead vehicle according to the Wi-Fi data it was receiving from front model).

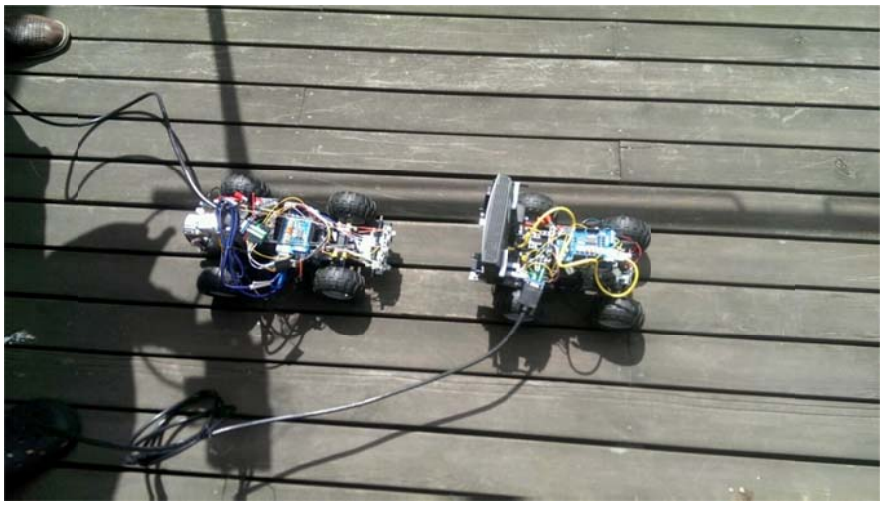


Figure 4-16: Began driving

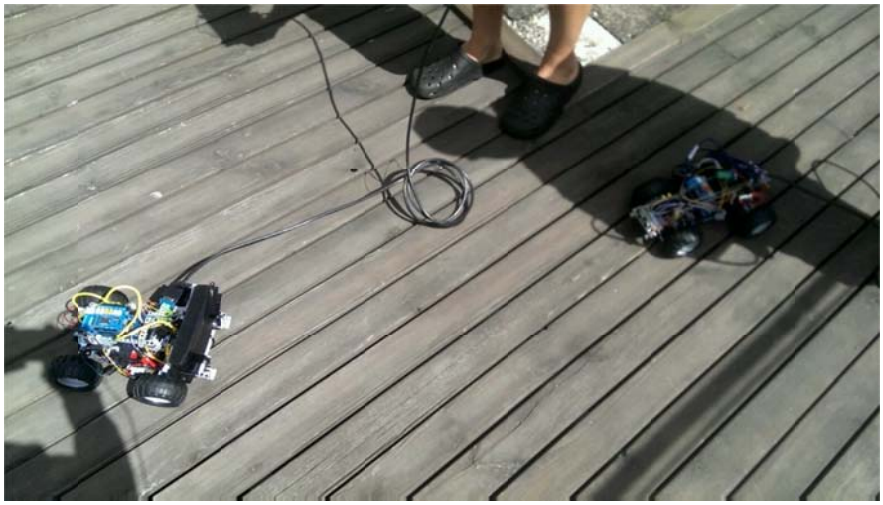


Figure 4-17: After 10 seconds of driving

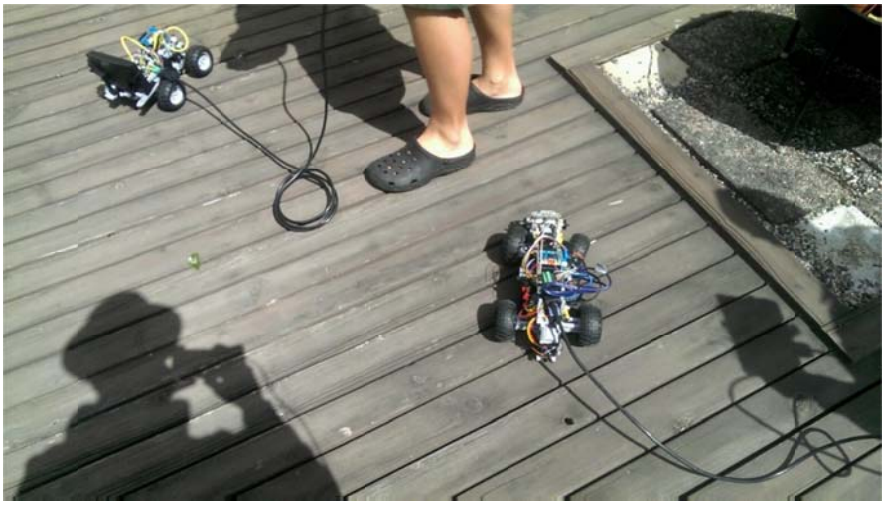


Figure 4-18: **After 20-25 seconds of driving. The front model at the upper left corner and the rear model in the middle which apparently chose its own path.**

5 Conclusions and Future work

This chapter involves the conclusion of the thesis and the future work of the thesis concept. First is my conclusion, what I have learned, whether I meet my goals, and if I made some mistakes and could pass some useful knowledge on to others working in the same area. Secondly is what future work I suggest. Specifically this second section describes things that I intentionally left undone or problems that I was unable to solve during my project. The last section of this chapter gives some of my reflections on this work regarding economy, environment, and other aspects.

5.1 Conclusions

I knew that inter-vehicle communication would be an important part of the future of driving and traffic management when I stated this thesis project 4 months ago. When I researched what had been done and what a number of large companies were working on I gained more understanding of how the simple components of different systems (such as measuring distance, communicating between several vehicle, maintaining a specific speed or direction) could altogether support a new means of safe and economical driving. By testing these ideas myself with a simple inter-vehicle communication between two cars implemented using model vehicles, I learned that these concepts could definitely improve driving. Of course, there are many problems remaining to be solved and ensure a reliable system that could perhaps replace human driving in the future, there need to be a lot more testing and analysis.

Because I worked alone in this thesis project I feel that I gained a lot of knowledge by trying to make a working system. Even though I did not create optimal choices regarding the speed, weight, and cabling, I managed to make an inter-vehicle communication system that worked. I am sure that inter-vehicle communication will someday work in real vehicles. My use a of a home router and access point increased the weight of the vehicles, thus I had to use fairly powerful motors. The weight of the vehicles required me to gear down the ratio of the motor speed about 1:7, otherwise the motor would have problems propelling the heavy models. An additional source of weight was the lead acid battery, even though they were the smallest capacity (12 V at 1.3 Ah). This small capacity and the malfunction the battery charger meant that I had to charge the batteries since the models required about 0.4 Amps. While each of my choices might not have been the best choice, they were sufficient to make models that worked sufficient to get some results. This made me realise that when inter-vehicle communication is transferred to bigger models or real vehicles, everything will need to be adapted to avoid unnecessary weight.

The goal was to examine whether inter-vehicle communication is a reliable communication mechanism for using in a convoy and to see if it benefits the convoy or creates new problems. To see if this goal was met, I analysed my test process and made a conclusion regarding Wi-Fi communication, since that was the only inter-vehicle communication technology that I used.

The test process was divided into two parts. The PING testing phase where I setup the system in two real vehicles and analysed the wireless communication while travelling through different traffic environments. The second phase was to implement the wireless communication in models where the goal was to make the rear model follow the front model regarding speed and direction.

During the first phase I had to ensure that the Wi-Fi devices, computers, and Arduino card could communicate while being inside a car. While I did encounter a problem where I could not make a connection between the computer in first car and the Arduino in the second car, I was able to communicate with the access point in the second car, which the Arduino was attached to. This enabled me to both establish that the Wi-Fi communication functioned and that this communication was independent of the computers attached to the Wi-Fi devices (the router and access point). In the analysis part of this phase I analysing the PINGs between the computer in the first car and access point in the second car. This analysis was shown in section 4.2. It is clear that the distance and the speed may have affected the PING replies leading to longer delays with higher speeds and longer distances. Some of the PINGs were lost because the distance between the vehicles was too far. From this, I drew

the conclusion that even though I met the goal of establishing communication between real vehicles, the environment through which these devices were moving and the driving conditions affected the communication more than I had thought. Unfortunately, I did not have time to test this communication in additional driving environments, such as tunnels or high density traffic in cities.

In the second phase I implemented inter-vehicle communication between two models vehicles. I encountered more problems than I expected, such as no connection between vehicles, data transfer was different from sending versus receiving, motor was running even though I did not ask for acceleration via the joystick, servo motor flipped out when trying to turn the front wheels, and the hand controller was not correctly configured.

While I managed to solve many of the problems that occurred, I was forced to resort to temporary solutions for some other problems. The majority of my problems were code related, but some of them were actually hardware based, such as a lost power in the cables or perhaps a hidden shortcut between a couple of wires. The first small test in which I tested the Arduino-to-Arduino wireless communication I succeeded almost right away, but the biggest problem was not to establish a connection it was to handle the data that was being sent between these two processors. It was very difficult to understand exactly how the Ethernet shield and its Arduino library operated when sending and receiving data. To better understand what was being done involved sending different types of values, such as sending as an array, a single byte, an integer, a character, and more. Since I used the keypad on the controller to request joining a convoy and an accept such a request, I had to read these input and then sending them via the Ethernet shield as characters (i.e., as bytes). The problem occurred when receiving them on the other model, as the ASCII values from the keypad were in the range of values for the motor and servo signals. This was caused by my not understanding that I needed to encode my message appropriately in order to send and receive them. I solved the problem by sending and receiving an array and processing the bytes by reading two bytes as a “word” with the library function “word(highByte, lowByte)”[40].

This thesis has given me a lot of experience even though it was only a 4 month long project. First, it has taught me how to work alone with a big project and what the limits are when working alone. Secondly, it showed me how many problems can encounter during a longer period of time. Working with a project such as this shows the demands for quality is and making decisions about what is significant and what is not.

I believe it is easy to work in this specific area if you have some interest in vehicles, traffic, or internet communication in general. It is of course fun to begin a project where you have to research outside of those areas you normally would work in. For me it has been very interesting to combine internet technology with vehicles and see what benefit the combination might provide.

If I were to do this project all over again, I would have focused more on only one test of the model vehicles, and perhaps focused on achieving excellent results for this one test, rather working with many processes concerning the test models as I tried to accomplish everything myself. Because I worked alone it is easy to start a variety of efforts and I was eager to succeed in everything. However, for quality’s sake, which is very important in this area, I should have focused on one component or one small sub-system of the complete problem of inter-vehicle communication in a convoy. I am not disappointed with my work and my results, but I know that the quality of work is not the best, with many errors, and poorly chosen solutions. I will take this experience with me in my career and especially in my future work.

5.2 Future work

When I started this thesis project I made a survey of what had been done and companies are currently working on. That information provided the background for my work with the models. In order to rapidly realize a prototype using these model vehicles I used an existing Wi-Fi router and access point as the wireless communication system between each of the two models. I also tried to use a USDR to estimate the distance between the vehicles.

Actual vehicles that use inter-vehicle communication also use GPS (a standard component already in vehicles). Additionally, cameras have been developed for use in vehicles to implementation ACC or city emergency brake systems. These components plus the wireless communication system will be the main components of future V2V systems. All these components were a bit too expensive for my experiment. Also it seemed they would be very difficult to work with in the small models that I utilized. Fortunately, by using a low cost router designed for home-use together with some chips to control two motors and perform the necessary I/O a low fidelity model system could be constructed.

In the future GPS and road-cameras that scan the road ahead could be added to the system to develop it further. Extensive research is already being conducted for using GPS and road-cameras, so we expect a lot of progress with these two sources of additional inputs. In comparison to actual V2V system my models left out these and many other systems. Some future technologies that could be explored include long distance radar and V2I systems. Radar could be used by the rear model to measure a much larger area and could be effective even in models. As described in Section 2.1.2, companies are already working on this.

I did not explore the use of V2I with my model vehicles, as they only use V2V. A V2I system could interoperate with Wi-Fi devices in public Wi-Fi networks. In the case of model vehicles they could communicate with Wi-Fi equipped cellphones, intersection cameras, or other devices that could provide additional inputs to the system. For example, a smartphone with a Wi-Fi interface could have been used instead of building a special purpose controller.

While there is quite a lot of future work related to this thesis, there are areas that could be seen as a complement to it. Although I prototype an inter-vehicle communication implementation with two models, there are many problems remaining before these systems will be implemented in real world, especially as they need to work safely and be approved by the various regulators that will be involved.

It is not sufficient to carry out tests with only two models. In order to study traffic behavior, there need to be many vehicles in order to extract patterns of driving in different environments. In the future it will be increasing important to carry out tests with real vehicles, instead of small. However, such testing will required greatly increased resources and perhaps authorization from different authorities to carry out such tests. In order to avoid these problems I chose to work with model vehicles. Another advantage of model vehicles is of course that any errors would not cause great damage or endanger human life.

To follow up this work, I suggest focusing on a specific subarea. My work focused on inter-vehicle communication but attempted to address many related problems in order to realize a prototype. For the future, it will be important to understand how in vehicle Ethernets can exploited by vehicles and what messages should be sent and how this communication can meet the requirements for safety, reliability, and functionality. Clearly, safety is the most important factor when driving in a convoy and it will be important to understand what requirements this will place on inter-vehicle communication within the convoy.

5.3 Required reflections

There are many aspects to mention related to this thesis concerning economic, environmental, ethical, and political issues. For example, V2V, V2I, intra-vehicle communication, and vehicle communication will need to address safety. As this thesis has emphasized safety is a critically important issue for convoy driving.

From a health perspective, wireless communication increases the radio energy that humans are exposed to. Already today, users are exposed to radio waves from their smartphones, surf pads, Bluetooth devices, Wi-Fi equipped devices, and more. So there is a question of what problems could this convoy communication cause and what are the benefits of this communication? Do the benefits outweigh the risks? What can be done to design the vehicles to minimize this exposure while gaining the maximum benefits?

The economic aspect of this area is very important and can have a very big effect. This new technology can further leverage the internet to provide many new services to vehicles in traffic. As noted in the beginning of this thesis convoy driving can directly reduce vehicles' fuel consumption. When combined with the self-driving vehicles that are now being tested the future seems to be eliminating the driver which could have a significant economic impact on freight transport and delivery and would also potentially have a major impact on traffic injuries and deaths, while increasing the effective capacity of existing roadways.

From a social perspective, I believe it will be difficult for society to adapt to a traffic system which relies on electronics, rather than humans. There is already a lot of reliance on electronics in the form of traffic lights at intersections, train crossings, and ferries to name a few applications where traffic movement is governed by electronics. When deploying convoys on highways, self-parking cars, traffic lights that communicate with individual cars and convoys, etc. there must be high reliability as the vehicle velocities will be increasing and the spacing between vehicles will be decreasing. So there is a question of whether considerations of social aspect will slow down the introducing of this technology or speed up the combination of the internet with traffic systems.

I believe there are some deep ethical issues that we will need to address in the near future. For example, is it good to rely on electronics that would replace humans in many situations in the future? What happens when an accident takes place and the car itself was driving? Where would these systems provide the most benefits? How will these systems affect humans in general? How should humans react to the introduction of such systems? All of these questions are hard to answer and I myself can only speculate what the effects will be.

Inter-vehicle communication system will benefit convoy driving and general driving as much as vehicles that will drive themselves in the future. However, this raises issues such as increased unemployment (since there will be a reduction in the need for professional drivers), an increased requirement to trust vehicle and traffic electronics, and the question of where there will be an option for human interaction with the system. For example, when driving fully automatic vehicles which communicate via internet should drivers have the option to take over their vehicle and drive as we do nowadays or will this increase the risks to others too much, increase fuel consumption and pollution to unacceptable levels, and allow too much independence of movement for individual drivers to be acceptable to society and governments?

References

[1] R. Ekman, “XXXX,” Kandidate thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Stockholm, Sweden, (expected) 2014.

[2] M. Gunnarsson, “Truck-Trailer Wireless Connections,” Master’s thesis, KTH, Microelectronics and Information Technology, IMIT, Stockholm, Sweden, 2001.

[3] “A Guide to ensuring Wireless LAN reliability,” *Motorola.com*. [Online]. Available: https://www.google.se/search?q=wifi+reliability&oq=wifi+&gs_l=serp.3.0.35i39j0l9.391088.395202.0.396868.5.5.0.0.0.0.1185.2945.1j1j4-1j7-2.5.0....0...1c.1.39.serp..3.2.1215.G2Lg0DWVigs. [Accessed: 06-Apr-2014].

[4] US Federal Communications Commission, “Amendment of the Comm ission’s Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band): Amendment of Parts 2 and 90 of the Commission’s Rules to Allocate the 5.850- 5.925 GHz Band to the Mobile Service for Dedicated Short Range Communications of Intelligent Transportation Services,” US Federal Communications Commission, Washington, D.C., USA, Report and Order FCC 03-324, Dec. 2003.

[5] “Fuel consumption calculator,” *scania.com*. [Online]. Available: http://www3.scania.com/en/Fuel-Consumption-Calculator/?utm_source=1&utm_medium=2&utm_campaign=6. [Accessed: 06-Apr-2014].

[6] L. Holm, “XXXX,” Kandidate thesis, KTH Royal Institute of Technology, School of Information and Communication Technology, Stockholm, Sweden, (expected) 2014.

[7] K. Amouris, “Space-time division multiple access (STDMA) and coordinated, power-aware MACA for mobile ad hoc networks,” presented at the IEEE Global Telecommunications Conference, 2001. GLOBECOM ’01, San Antonio, TX, USA, vol. 5, pp. 2890–2895.

[8] Strålskyddsstiftelsen, “www.stralskyddsstiftelsen.se/op/sites/default/files/pub/faktablad/13-0005-05.pdf,” *stralskyddsstiftelsen.se*, 2013. [Online]. Available: about:blank. [Accessed: 06-Apr-2014].

[9] L. Hardell, “Case-control study of the association between malignant brain tumours diagnosed between 2007 and 2009 and mobile and cordless phone use,” *International Journal of Oncology*, Sep. 2013.

[10] B. Mitchell, “What Is a Wireless Dead Zone?,” *About.com Wireless / Networking*. [Online]. Available: http://compnetworking.about.com/b/2011/03/09/what-is-a-wireless-dead-zone.htm. [Accessed: 06-Apr-2014].

[11] Raytheon Company, “Raytheon Company: ASR-11 Digital Airport Surveillance Radar (DASR).” [Online]. Available: http://www.raytheon.com/capabilities/products/asr11/. [Accessed: 06-Apr-2014].

[12] CAN-Cia, “CAN in Automation (CiA): Radar for vehicle safety,” *can-cia.org*. [Online]. Available: http://www.can-cia.org/index.php?id=1673. [Accessed: 06-Apr-2014].

[13] N. Bowyer, “InterRegs - Regulations Spotlight,” *InterRegs.com*, Jun-2012. [Online]. Available: http://www.interregs.com/spotlight.php?id=117. [Accessed: 06-Apr-2014].

[14] EUR-Lex, “Commission Regulation (EU) No 347/2012 of 16 April 2012,” *eur-lex.europa.eu*, 2012. [Online]. Available: eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2012:109:0001:0017:EN:PDF. [Accessed: 25-Apr-2014].

[15] ETSI, “ETSI - Automotive Radar,” *ETSI*. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/intelligent-transport/automotive-radar. [Accessed: 06-Apr-2014].

[16] Bosch Engineering, “Train Driver Assistance Radar, same type of LRR3 radar as used in vehicles,” *http://www.bosch-engineering.de/en*. [Online]. Available: http://www.bosch-

engineering.de/en/de/einsatzgebiete/schienenfahrzeuge/sicherheit_8/lokfuehrerassistenz/lokfuehrerassistenz_1.html. [Accessed: 27-Jun-2014].
[17] J. Lee, Y.-A. Li, M.-H. Hung, and S.-J. Huang, "A Fully-Integrated 77-GHz FMCW Radar Transceiver in 65-nm CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, pp. 2746–2756, Dec. 2010.
[18] Audi, "Adaptive cruise control - Driver assistants - Audi A8," *Audi UK*. [Online]. Available: http://www.audi.co.uk/content/audi/new-cars/a8/a8/driver-assistants/adaptive-cruise.html. [Accessed: 06-Apr-2014].
[19] Scania, "Adaptive Cruise Control (ACC) - scania.com." [Online]. Available: http://www.scania.com/products-services/trucks/safety-driver-support/driver-support-systems/acc/. [Accessed: 06-Apr-2014].
[20] Scania, "Scania lines up for platooning trials." [Online]. Available: http://newsroom.scania.com/en-group/2012/04/04/scania-lines-up-for-platooning-trials/. [Accessed: 06-Apr-2014].
[21] Scania, "Innovative Scania: Rolling towards platooning." [Online]. Available: http://newsroom.scania.com/en-group/2013/09/30/innovative-scania-rolling-towards-platooning/. [Accessed: 25-Apr-2014].
[22] Volvo, "The SARTRE project," *sartre-project.eu*. [Online]. Available: http://www.sartre-project.eu/en/Sidor/default.aspx. [Accessed: 06-Apr-2014].
[23] EURO NCAP advanced, "Volvo City Safety | Euro NCAP - For safer cars krocktest säkerhet," *se.euroncamp.com*. [Online]. Available: http://se.euroncap.com/se/rewards/volvo_city_safety.aspx. [Accessed: 06-Apr-2014].
[24] Comtrol, "Power over Ethernet (PoE) For Intersection Monitoring http://www.comtrol.com/applications/transportation/application-stories/power-over-ethernet-poe-for-intersection-monitoring," *COMTROL Corp*, Apr-2014. .
[25] C. Lindenau, "Cameras at the Intersection -- Security Today," 01-Dec-2013. [Online]. Available: http://security-today.com/articles/2013/12/01/cameras-at-the-intersection.aspx. [Accessed: 06-Apr-2014].
[26] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96–104, Jun. 2013.
[27] GM News, "GM Developing Wireless Pedestrian Detection Technology," *media.gm.com*, 26-Jul-2012. [Online]. Available: http://media.gm.com/media/us/en/gm/news.detail.html/content/Pages/news/us/en/2012/Jul/0726_pedestrian.html. [Accessed: 06-Apr-2014].
[28] V. Hennigan, "Vehicle Technology If cars could talk, what would they say? | Ford Tech LaneFord Tech Lane," *Ford Tech Lane*, 21-Aug-2012. [Online]. Available: http://fordtechlane.com/if-cars-could-talk-what-would-they-say-to-each-other/. [Accessed: 03-Jul-2014].
[29] U.S. DOT RITA, "RITA - Intelligent Transportation Systems - DSRC: The Future of Safer Driving Fact Sheet," *its.dot.gov*. [Online]. Available: http://www.its.dot.gov/factsheets/dsrc_factsheet.htm. [Accessed: 06-Apr-2014].
[30] S. Eiderbrant, *Analytical model of a vehicle platoon*. 2013.
[31] J. Kjellberg, *Implementing control algorithms for platooning based on V2V communication*. 2011.
[32] M. Amoozadeh, *Certificate Revocation List Distribution in Vehicular Communication Systems*. 2012.
[33] Scrum Alliance, "What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance," *scrumalliance.org*. [Online]. Available: http://www.scrumalliance.org/why-scrum. [Accessed: 07-Apr-2014].
[34] D-link, "How to extend your network with a Wireless Bridge," *dlink.com*. [Online]. Available: http://www.dlink.com/us/en/resource-centre/how-to-guides/how-to-extend-your-network-with-a-wireless-bridge. [Accessed: 07-Apr-2014].

[35] “Arduino Playground - ICMP Ping Library,” *playground.arduino.cc*, 16-Apr-2014. [Online]. Available: http://playground.arduino.cc/Code/ICMPPing. [Accessed: 16-Apr-2014].
[36] ElecFreaks, “HC-SR04 sensor manual,” *www.micropik.com*. [Online]. Available: http://www.micropik.com/PDF/HCSR04.pdf. [Accessed: 24-Jun-2014].
[37] SwanRobotics, “HC-SR04 Project | SwanRobotics.com,” *www.swanrobotics.com*. [Online]. Available: http://www.swanrobotics.com/HC-SR04_Project. [Accessed: 27-Jun-2014].
[38] D. Mellis and T. Igoe, “Arduino - Ping,” *arduino.cc*. [Online]. Available: http://arduino.cc/en/Tutorial/Ping?from=Tutorial.UltrasoundSensor. [Accessed: 27-Jun-2014].
[39] P. Hurbain, “LEGO 9V Technic Motors compared characteristics,” *www.philohome.com*. [Online]. Available: http://www.philohome.com/motors/motorcomp.htm. [Accessed: 15-Jun-2014].
[40] Arduino, “Arduino - WordCast,” *www.arduino.cc*. [Online]. Available: http://arduino.cc/en/Reference/WordCast. [Accessed: 19-Jun-2014].

Appendix

This appendix contains the code snippets that I used for different tests. Code listing 1.1 realizes a PING program which tries to ping the network gateway.

Code listing 1.1

```
/*
  Ping Example

 This example sends an ICMP pings every 500 milliseconds, sends the human-
           readable result over the serial port.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 created 30 Sep 2010
 by Blake Foster
 */

#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}; // max address for
            Ethernet shield
byte ip[] = {192,168,0,4}; // IP address for Ethernet shield
IPAddress pingAddr(192,168,0,1); // IP address to ping
int ping_count = 10;
SOCKET pingSocket = 0;
char buffer [256];
ICMPPing ping(pingSocket, (uint16_t)random(0, 255));

void setup()
{
  // start Ethernet
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
}

void loop()
{
  if(ping_count > 0)
  {
  ICMPEchoReply echoReply = ping(pingAddr, 4);
  if (echoReply.status == SUCCESS)
```

```
  {
    sprintf(buffer,
            "Reply[%d] from: %d.%d.%d.%d: bytes=%d time=%ldms TTL=%d",
            echoReply.data.seq,
            echoReply.addr[0],
            echoReply.addr[1],
            echoReply.addr[2],
            echoReply.addr[3],
            REQ_DATASIZE,
            millis() - echoReply.data.time,
            echoReply.ttl);
  }
  else
  {
    sprintf(buffer, "Echo request failed; %d", echoReply.status);
  }
  Serial.println(buffer);
  ping_count--;
  delay(20);
  }
}
```

Code listings 2.1 and 2.2 represents the test where I sent simple motor speed signals between two Arduinos connected to router and access point, described in section 3.4.5. Code 2.1 runs on the Arduino which has the router connected to it, and is also connected to computer #1. Code 2.2 then runs on the second Arduino which is connected to the access point.

Code listing 2.1

```
#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>
byte mac[] = {0xDE, 0xAD, 0xBE, 0xFF, 0xFE, 0xED};
byte ip[] = {192,168, 0, 5};
byte master[] = {192,168,0,4};
EthernetClient client;
void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
  delay(1000);
  if(client.connect(master, 40)){
    Serial.println("Connected");
  }
}
void loop()
{
     while(Serial.available() > 0){
      int speed = Serial.parseInt();
      if(client.connected()){
        client.print(speed);
        client.print('e');
        Serial.print("Input speed: ");
        Serial.println(speed);
      }
      else{
        client.connect(master, 40);
        Serial.println("Connected");
      }
    }
}
```

Code listing 2.2

```
#include <SPI.h>
#include <Ethernet.h>
#include <ICMPPing.h>
#include <String.h>
const int PWM_A = 3, DIR_A = 12,BRAKE_A = 9;
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xDD}; //mac address
byte ip[] = {192,168,0,4}; // IP address for Ethernet shield
EthernetServer master = EthernetServer(40);
int sekvens = 1;
String s;
int motorSpeed = 0;
void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
  //pinMode(DIR_A, OUTPUT);
  //pinMode(BRAKE_A, OUTPUT);
}
void loop()
{
  EthernetClient client = master.available();
  if(client == true)
  {
    char c = client.read();
    if(c != 'e')
      s += c;
    else{
      motorSpeed = s.toInt();
      //Serial.println(number);
      s = "";
    }
  }
  delay(10);
  digitalWrite(BRAKE_A, LOW);
  analogWrite(PWM_A, motorSpeed);
}
```

Code listings 2.3 and 2.4 represents the simple steering signals that were sent from the Arduino connected to router, to the Arduino connected to the access point. Code 2.3 runs on the first Arduino and code 2.4 on the second one.

Code listing 2.3

```
#include <SPI.h>
#include <Ethernet.h>
#include <Servo.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xFF, 0xFE, 0xED};
byte ip[] = {192,168, 0, 5};
byte master[] = {192,168,0,4};

EthernetClient client;
Servo servo;

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
  servo.attach(24);
  delay(1000);
  if(client.connect(master, 40)){
    Serial.println("Connected");
  }
}
void loop()
{
    /*
    Servo turning
    */
    int firstChar = 0;
    while(Serial.available() > 0){
      char type;
      if(firstChar == 0){
        type = Serial.read();
        firstChar = -1;
      }
      if(type == 't')
      {
        int degree = Serial.parseInt();
        servo.write(degree);
        if(client.connected()){
          client.print(type);
```

```
          client.print(degree);
          client.print('e');
          Serial.print("Input type = ");
          Serial.print(type);
          Serial.print(" value = ");
          Serial.println(degree);
        }
        else{
          client.connect(master, 40);
          Serial.println("Connected");
        }
      }
      else
      {
        int value = Serial.parseInt();
        if(client.connected()){
          client.print(type);
          client.print(value);
          client.print('e');
          Serial.print("Input type = ");
          Serial.print(type);
          Serial.print(" value = ");
          Serial.println(value);
        }
        else{
          client.connect(master, 40);
          Serial.println("Connected");
        }
      }
    }
}
```

Code listing 2.4

```
#include <SPI.h>
#include <Ethernet.h>
#include <String.h>
#include <Servo.h>

const int
PWM_A = 3,
DIR_A = 12,
BRAKE_A = 9;

Servo servo;


byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xDD}; // mac address for
            ethernet shield
byte ip[] = {192,168,0,4}; // IP address for Ethernet shield

EthernetServer master = EthernetServer(40);
int sekvens = 1;
String motorData;
String servoData;
int motorSpeed = 0;
int servoPosition = 0;
int writeServo = 0;

void setup()
{
  Ethernet.begin(mac, ip);
  Serial.begin(9600);
  servo.attach(24);
}
void loop()
{
  EthernetClient client = master.available();
  if(client == true)
  {
    char c = client.read();
    //if 't' then receive turning data
    if(c == 't')
    {
      sekvens = 2;
```

```
    }
    //else if 'm' then receive motor data (not significant in this test)
    else if(c == 'm')
    {
      sekvens = 3;
    }
    else if(c != 'e' && sekvens == 2)
      servoData += c;
    else if(c != 'e' && sekvens == 3)
      motorData += c;
    else if(c == 'e' && sekvens == 2){
      servoPosition = servoData.toInt();
      servoData = "";
      writeServo = 1;
    }
    else if(c == 'e' && sekvens == 3){
      motorSpeed = motorData.toInt();
      //Serial.println(number);
      motorData = "";
    }
  }
  delay(10);
  if(writeServo == 1){
    servo.write(servoPosition);
    writeServo = 0;
  }
}
```

Final versions of code for the models.

There are two versions of the code on both models: TCP with bad steering algorithm and UDP with not steering algorithm. The UDP versions were tested but the front vehicle was not receiving any data, however the code algorithms in general are supposed to work. Total of four final codes in this appendix section. Code listing 3.1: TCP code on front vehicle. Code listing 3.2: TCP code on rear vehicle. Code listing 3.3: UDP code on front vehicle. Code listing 3.4: UDP code on rear vehicle.

Code listing 3.1

```
#include <SPI.h>
#include <Ethernet.h>
#include <String.h>
#include <Servo.h>
#include <Keypad.h>

const int
PWM_A = 3,   /* pin that controlls pulse width modulation value on motor */
DIR_A = 12,       /* pin that controlls direction of the motor */
BRAKE_A = 9,/* pin that controlls if the brake is on or off on the motor */
Joystick_x_axle_pin = A9,  /* pin that reads the X-value
            of the right joystick */
Joystick_y_axle_pin = A8,  /* pin that reads the Y-value
of the left joystick */
LED_1 = 26,       /* LED 1 that represents the ethernet connection state */
LED_2 = 27;       /* LED 2 that represents the convoy connection state */

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] =
  {{'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}};
byte rowPins[ROWS] = {28,31,32,33};
byte colPins[COLS] = {37,34,35,36};
int count = 0;
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

Servo servo;

/* mac address of ethernet shield */
byte mac[] = {0xDE, 0xAD, 0xBE, 0XEF, 0xFE, 0xDE};
IPAddress ip(192,168,0,10);            /* ip address of ethernet shield */
unsigned int localPort = 40;           /* port to be used in connection */
```

```
boolean active_convoy = false;        /* active convoy variable */
boolean active_request = false;       /* request joining convoy variable */
boolean close_convoy = false;         /* close convoy variable */

/* create server connection on specified port */
EthernetServer server = EthernetServer(localPort);

const int Maximum_command_length = 128;                /* define maximum
            command length of buffer */
char command_buffer_char[Maximum_command_length];       /* buffer to write
            motor and speed data into */
byte command_buffer_byte[Maximum_command_length];       /* buffer to send
            converted motor and speed data */

/* ------------------------------------------------------------ */
void setup(){
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(Joystick_x_axle_pin, INPUT);
  pinMode(Joystick_y_axle_pin, INPUT);
  servo.attach(47);
  Serial.begin(9600);/* allow serial connection if computer is connected */

  int i;
  for(i = 0; i < 30; i++)
    delay(1000);                        /* allow router to be booted up
            before initializing ethernet shield */
  digitalWrite(LED_1, HIGH);    /* indicate that ethernet initializing has
            begun */
  Ethernet.begin(mac, ip);
  delay(1000);
  server.begin();
  delay(1000);
  digitalWrite(LED_1, LOW);         /* turn off when ethernet is ready */
}
/* ------------------------------------------------------------ */
void loop(){
 /*-----READ 4x4 KEYPAD-----------------*/
 char key = keypad.getKey();
 delay(5);
/* if master has got a request from slave and press 'A' on keypad */
 if(active_request == true && key == 'A'){
   server.write('A');           /* send back char 'A' and wait for echo */
```

```
   digitalWrite(LED_1, HIGH);    /* indicate that master has accepted
            convoy request */
 }
/* if convoy is active and master press 'C' on keypad */
 if(active_convoy == true && key == 'C'){
   server.write('C');                          /* write 'C' to slave */
   active_convoy = false;                      /* close convoy */
   digitalWrite(LED_1, LOW);                   /* turn off LED 1 */
 }
/* read right joystick */
 int steering_angle = analogRead(Joystick_x_axle_pin);
 delay(2);
/* read left joystick */
 int motor_speed = analogRead(Joystick_y_axle_pin);
 delay(2);
/*map read values to proper servo angle values */
 steering_angle = map(steering_angle, 1023, 0, 40, 140);




 if(motor_speed < 500)
/* map to proper motor speed values */
   motor_speed = map(motor_speed, 500, 0, 0, 255);
 else
   motor_speed = 0;

 /*apply motor speed and servo degree values */
 digitalWrite(BRAKE_A, LOW);
 analogWrite(PWM_A, motor_speed);
 servo.write(steering_angle);
 delay(2);
 /* --------READ ETHERNET DATA---------------- */
 EthernetClient slave = server.available();
 if(slave == true){
   char data = slave.read();            /* read char from server */
   switch (data){
    case 'A':                           /* if char is 'A' */
     active_convoy = true;              /* activate convoy */
     active_request = false;            /* reset request variable */
     digitalWrite(LED_1, LOW);       /* turn off LED 1 and turn on LED 2 */
     digitalWrite(LED_2, HIGH);
     break;
```

```
    case 'C':                           /* if char is 'C' */
      server.write('C');                /* echo it back */
      active_convoy = false;            /* and close the convoy */
      break;

    case 'R':                           /* if char is 'R' */
      active_request = true;   /* set current state to request is active */
      digitalWrite(LED_1, HIGH);   /* turn LED 1 and 2 ON for one second */
      digitalWrite(LED_2, HIGH);
      delay(1000);
      digitalWrite(LED_1, LOW);
      digitalWrite(LED_2, LOW);
   }
 }
 /* -----WRITE MOTOR AND SERVO DATA -------------- */
 if(active_convoy == true){
   /* write steering angle and motor speed to char buffer */
   sprintf(command_buffer_char, "#steering_angle=%d,motor_speed=%dE\n",
            steering_angle, motor_speed);
   int i;
   for(i = 0; i < 128; i++){
/* convert every char to byte value */
     command_buffer_byte[i] = (byte)command_buffer_char[i];
/* when last char is reached, break */
     if(command_buffer_char[i] == 'E'){
 break;
     }
   }
/* send byte buffer to server */
   server.write(command_buffer_byte, 128);
 }
}
```

Code listing 3.2

```
#include <SPI.h>
#include <Ethernet.h>
#include <String.h>
#include <Servo.h>
#include <Keypad.h>

const int
PWM_A = 3,      /*pin that controlls the pulse width modulation value of
           the motor */
DIR_A = 12,     /*pin that sets the direction of the motor*/
BRAKE_A = 9,    /*pin that controlles whether the brake is one or of on the
           motor*/
trigPin = 48,    /*pin that trig the signal of the USDR sensor*/
echoPin = 49,    /*pin that listen to the echo signal of the USDR sensor*/
Joystick_x_axle_pin = A8,        /*pin that reads the X-axle values on the
           right joystick and applies these values to servo motor*/
Joystick_y_axle_pin = A9;        /*pin that reads the Y-axle values on the
           left joystick and applies these values to motor*/

const byte ROWS = 4;
const byte COLS = 4;
/*matrix that represents the 4x4 keypad*/
char keys[ROWS][COLS] =
  {{'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}};
byte rowPins[ROWS] = {30,31,32,33};    /*match the row pins on the matrix
           with Arduno pins*/
byte colPins[COLS] = {37,34,35,36};    /*match the column pins on the
           matrix with Arduino pins*/
//create a keypad object by the 4x4 matrix*/
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

/*connection states LEDs*/
const int
LED_1 = 40,
LED_2 = 41;

/* ACC variables */
int keypad_input_value = 0;
int default_acc_distance = 80;
```

```
int previous_acc_distance = 80;
int current_acc_distance = 80;
int measured_distance = 0;
boolean set_acc_distance = false;
boolean set_acc_speed = false;
boolean acc = false;
int current_acc_speed = 0;

Servo servo;

/* Ethernet variables */
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xDD};
IPAddress ip(192, 168, 0, 11);
IPAddress master(192,168,0,10);
unsigned int port = 40;
EthernetClient slave;
int connect_attempt;              /*current attempt to connect to master*/
const int Maximum_connect_attempts = 10;  /*maximum times to attempt to
            connectt to master*/
const int Maximum_command_length = 128;    /*maximum length for commands*/
char command_buffer_char[Maximum_command_length];  /*char buffer for
            scaning commands */




/*Convoy variables*/
boolean active_convoy = false;
boolean receive_data = false;
int received_motor_speed = 0;

int wifi_speed = 0;
int wifi_steering = 95;
byte servo_value, motor_value;
/*loop variable for use when parsing the receiving buffer from master */
int current_byte;
int turn_buffer[256]; /*buffer for storing servo angle degrees */
int store_steer_degree = 95;    /* default storing value in buffer */
int current_buffer_write_index = 0;  /*begin write on first index */
/*begin reading on buffer index 220 for a static delay when begin turning
            */
int current_buffer_read_index = 220;
int received_steering_angle = 0;
```

```
/*-------------------------------------------------------------------------
           ----------------------*/
void setup(){
  /*setup of LED and joysticks as output or input, and attach servo motor*/
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(Joystick_x_axle_pin, INPUT);
  pinMode(Joystick_y_axle_pin, INPUT);
  servo.attach(24);
  Serial.begin(9600);          /* allow Serial connection when debugging or
           reprogramming with computer */
  /* Ethernet shield and connection setup */
  int i;
  for(i = 0; i < 50; i++){
/*50 second delay to enable the access point to start up*/
    delay(1000);
  }
  digitalWrite(LED_1, HIGH);  /*turn on LED1 for indicating the ethernet
           connection is initalizing*/
  Ethernet.begin(mac, ip);
  delay(1000);                /* 1 second delay for the initalizing of
           ethernet shield*/

  for(connect_attempt = 0; connect_attempt < Maximum_connect_attempts;
           connect_attempt++){
/* if a connection with master was established, turn on LED 2 to indicate
           drivers of that*/
    if(slave.connect(master, port)){
      digitalWrite(LED_2, HIGH);
    }
    delay(1000);
/* and after 1 second, turn of LED 1 to indicate drivers that ethernet
           initalizing has ended */
    if(slave.connected()){
      digitalWrite(LED_1, LOW);
      break;
   }
  }

  int filling_buffer_index;
  for(filling_buffer_index = 0; filling_buffer_index < 256;
           filling_buffer_index++){
    turn_buffer[filling_buffer_index] = 95;
  }
}
```

```
/*-----------------------------------------------------------------------*/
void loop(){
  /*--------------READ 4x4 KEYPAD-----------------------------*/
  char key = keypad.getKey();
  /*---------------READ JOYSTICKS-----------------------------*/
  int x_axle_value = analogRead(Joystick_x_axle_pin);  /* read value on x-
          axle pin */
  delay(2);                   /* delay 2ms for value to be read properly */
  int y_axle_value = analogRead(Joystick_y_axle_pin);  /* read value on y-
          axle pin */
  delay(2);                   /* delay 2ms for value to be read properly */
  int servo_steering_value, motor_speed_value;          /* define values
          for later appliance */

  /*read values on joystick pins are values of 0-1023 but due to how I
          mounted these joysticks on controller,
    the values will be read in negative way. The map-function normally has
          input as:
     (source, fromLowest, fromHighest, toLowest, toHighest) which means I
          should normally have used it like
     (source, 0, 1023, toLow, toHigh) but I had to use it like (source,
          1023, 0, toLow, toHigh).
    Since the joysticks are center-positioned when not used, I had to
          filter out the values from the furthest down
    to the center posistion which was, due to the negative reading, values
          512-1023. I included a dead-zone of the
    accelerator joystick from 512-530 to remove the accidently reading of
          the joystick when center posistioned because
    then values would vary between 510-515. */

  servo_steering_value = map(x_axle_value, 1023, 0, 40, 140);
  if(y_axle_value < 530)
    motor_speed_value = map(y_axle_value, 530, 0, 0, 255);
  else
    motor_speed_value = 0;


  if(active_convoy == false){
    if(acc == false){
      digitalWrite(BRAKE_A, LOW);
      analogWrite(PWM_A, motor_speed_value);    /* if neither convoy nor
            acc is active, */
      servo.write(servo_steering_value);        /* apply joystick values to
            motor and servo */
      delay(2);
    }
```

```
  else{
    if(measured_distance < 5)
      current_acc_speed = 0;                    /* stop motor if distance
          to obstacle is too close */
    else if(measured_distance < 10)
      current_acc_speed -= 20;                  /* decelerate motor if
          distance to obstacle is fairly close */
    else if(measured_distance < current_acc_distance)
      current_acc_speed -=2;                 /* decelerate little when
          distance is not so short */
    else if(measured_distance > current_acc_distance && current_acc_speed
          < reference_acc_speed)  /*distance further than set, */
      current_acc_speed += 5;                /* accelerate */
    if(current_acc_speed < 0)
      current_acc_speed = 0;
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, current_acc_speed); /* apply acc speed to motor */
    servo.write(servo_steering_value);/*apply joystick values to servo */
    delay(2);
  }
}
else{
  if(measured_distance < 5)
    wifi_speed = 0;         /* stop motor when obstacle is very close */
  else if(measured_distance < 10)          /* decelerate hard since
          wifi_speed will be restored */
    wifi_speed -= 50;             /* by the received value from Wi-Fi */
  else if(measured_distance < 20)
    wifi_speed -= 30;                       /* decelerate fairly hard when
          distance is not so long */
  if(wifi_speed < 0)
    wifi_speed = 0;
  if(wifi_speed >= 30){/* enough speed to make motor axle start moving */
    wifi_steering = turn_buffer[current_buffer_read_index];  /*read wifi
          steering from buffer */
    current_buffer_read_index++;             /* next index */
    if(current_buffer_read_index == 256)
      current_buffer_read_index = 0;         /* start over from buffer
          beginning */
  }
  digitalWrite(BRAKE_A, LOW);
  analogWrite(PWM_A, wifi_speed);           /* apply motor speed */
  servo.write(wifi_steering);               /* apply servo angle value */
  delay(2);
}
```

```
delay(12); /* Intentionally delay for proper ULTRASOUND SENSOR USE */


/*---------------- READ ULTRASOUND SENSOR ---------------*/
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
digitalWrite(trigPin, LOW);    /* reset signal */
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);   /* trig a signal */
delayMicroseconds(10);
digitalWrite(trigPin, LOW);    /* stop triggering */
long dur = pulseIn(echoPin, HIGH);  /* read the echo */
measured_distance = microsecondsToCentimeters(dur);  /* convert to
          distance */

delay(12); /* Intentionally delay for proper ULTRASOUND SENSOR USE */

/*-----------------------HANDLE KEYPAD BUTTONS-----------------------*/
switch (key){
 /*ACC ON*/
 case 'A':                                          /* key is A */
  if(active_convoy == false && acc == false){    /* check if convoy is
          inactive and acc is off */
   acc = true;                                       /* turn acc on */
   current_acc_speed = motor_speed_value;                /* save current
          speed to acc speed */
  }
  else if(active_convoy == false && acc == true){ /* check if convoy is
          inactive but acc is on */
    acc = false;                                     /* turn acc off */
  }
  break;
 /* "JOIN CONVOY"/"ABORT CONVOY" - REQUEST */
 case 'B':                                          /* key is B */
    if(active_convoy == false){
      slave.write('R');    /* if convoy is inactive, send join request */
      digitalWrite(LED_1, HIGH); /* and turn on LED 1 to inform driver */
    }
    else{
      slave.write('C');/*if convoy is active, send abort convoy request*/
    }
    break;
```

```
   case '#':
      /* when acc is on (only when convoy is inactive) and currently not
            already setting speed,
set speed parameter to true */

      if(acc == true && active_convoy == false && set_acc_speed == false){
        set_acc_speed = true;
      }
      else if(acc == true && active_convoy == false && keypad_input_value >
            0 && set_acc_speed == true){
        if(keypad_input_value >= 100)                /* when acc is turned
            on and input speed is higher or equal to 100, */
         current_acc_speed = keypad_input_value;             /* save it for
            applying to current speed (100 is just a feasible minimum
            speed)*/
        keypad_input_value = 0;             /* reset incrementing value */
        set_acc_speed=false; /*reset current setting speed state to false*/
      }
      break;

    case 'D':
      /* like speed setting on acc, the distance setting method work the
            same */
      if(acc == true && active_convoy == false && set_acc_distance ==
            false){
        set_acc_distance = true;
      }
      else if(acc == true && active_convoy == false && keypad_input_value >
            0 && set_acc_distance == true){
        if(keypad_input_value > 20)                  /* feasible minimum
            distance between vehicles */
          current_acc_distance = keypad_input_value;
        else
          current_acc_distance = 20;      /* if set to lower, the default
            value will replace the input */
        keypad_input_value = 0;        /* reset the incrementing value */
        set_acc_distance = false;         /* reset current setting distance
            state to false */
      }
      break;

    case '0':    /* check digits */
    case '1':
    case '2':
    case '3':
```

```
    case '4':
    case '5':
    case '6':
    case '7':
    case '8':
    case '9':
      int digit = (key - '0');            /* change from char to integer */
      keypad_input_value = digit + (keypad_input_value * 10);


/* replace the current keypad_input_value value depending on input */
/* the replacing is active so its happening after every button press */
/* and not calculated after every specifik buttons have been pressed */
/* works for 1-5 digits. Example: */
/* input = 5 -> */
/* keypad_input_value = 5 + (0*10) = 5 */
/* input = 120 -> */
/* keypad_input_value = 1 + (0*10) = 1  ->  */
/* keypad_input_value = 2 + (1*10) = 12  -> */
/* keypad_input_value = 0 + (12*10) = 120 */
  }
  delay(10); /* Intentionally delay for proper USDR SENSOR use */

  /*-----------------------------TA EMOT FRÅN MASTER----------------------
           -----------*/
  if(slave.available()){
    byte read_byte = slave.read();  /* read a byte */
    switch (read_byte){
     case 'A':
      if(active_convoy == false){
       slave.write('A');            /* write acknowledge echo */
       active_convoy = true;        /* activate convoy */
       digitalWrite(LED_1, LOW);
       digitalWrite(LED_2, HIGH);    /* indicate convoy is active */
      }
      break;

     case 'C':
       active_convoy = false;      /* close the convoy */
       break;

     case '#':
       for(current_byte = 0; current_byte < Maximum_command_length;
            current_byte++){
```

```
          read_byte = slave.read();
/*convert byte to char value and store in buffer */
          command_buffer_char[current_byte] = (char)read_byte;
          if(read_byte == 'E'){
/* scan the buffer and read the received values of motor and servo */
            sscanf(command_buffer_char, "steering_angle=%d,motor_speed=%d",
             &received_steering_angle, &received_motor_speed);
            wifi_speed = map(received_motor_speed, 0, 255, 0, 200);

/* part of steering algorithm - save steering values to buffer for later
             readings */
            if(wifi_speed >= 30){
              turn_buffer[current_buffer_write_index] =
             received_steering_angle;
              current_buffer_write_index++;
              if(current_buffer_write_index == 256)
                current_buffer_write_index = 0;
            }
            break;
          }
        }
    }
  }
  delay(20);
}
/* ----KONVERTERA ULTRALJUDSENSORNS TID TILL AVSTÅND------------------*/
long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

Code listing 3.3

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <String.h>
#include <Servo.h>
#include <Keypad.h>

const int
PWM_A = 3,      /* pin that controls the pulse width modulation value of
            the motor */
DIR_A = 12,     /* pin that sets the direction of the motor */
BRAKE_A = 9,    /* pin that controls whether the brake is on or off */
VRx = A9,      /* pin that reads the X-axle values on the right joystick */
VRy = A8,       /* pin that reads the Y-axle values on the left joystick */
LED_1 = 26,     /* pin for LED. Represents ethernet connection states */
LED_2 = 27;     /* pin for LED. Represents convoy connection states */

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] =
  {{'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}};
byte rowPins[ROWS] = {28,31,32,33};
byte colPins[COLS] = {37,34,35,36};
int count = 0;
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

Servo servo;

byte mac[] = {0xDE, 0xAD, 0xBE, 0XEF, 0xFE, 0xDE};
IPAddress ip(192,168,0,10);
/* IPAddress slave(192,168, 0, 11); */    /* inactive backup connection
            setup */
unsigned int localPort = 8888;
EthernetUDP Udp;

boolean active_convoy = false;
boolean active_request = false;
boolean close_convoy = false;

char command_buffer_char[UDP_TX_PACKET_MAX_SIZE];
```

```
byte command_buffer_byte[UDP_TX_PACKET_MAX_SIZE];
int current_index;

unsigned long start_time_request, start_time_ack, start_clock_broadcast,
            end_time_request,
              end_time_ack, end_clock_broadcast = 0, broadcast_time = 4000,
            time_out_limit = 10000;
boolean time_out = false, timing_out_request = false, timing_out_ack =
            false, compute_time = false;

/* Variables for broadcasting from slave to master, broadcasting is LED_2
            flashing every time slave
   send a broadcast signal */
boolean broadcast_LED = false, timing_out_broadcast = false;
unsigned long led_start_time = 0, led_end_time;



/*-----------------------------------------------------------------------*/
void setup(){
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  servo.attach(47);
  Serial.begin(9600);    /* Allow serial communication with computer when
            troubleshooting */

  int i;
  for(i = 0; i < 30; i++)
    delay(1000);            /* Allow the router to start up before Arduino
            ethernet initialize */
  digitalWrite(LED_1, HIGH);  /* Turn on LED_1 to indicate Ethernet shield
            is initializing */
  Ethernet.begin(mac, ip);    /* Ethernet init */
  Udp.begin(localPort);        /* UDP init */
  delay(1000);                  /* Allow init time */
  digitalWrite(LED_1, LOW);    /*Turn off when init is done */
}
/*-----------------------------------------------------------------------*/
void loop(){
 /*----------------------READ 4x4 KEYPAD----------------------------*/
 char key = keypad.getKey();/*read keypad and save key to character "key"*/
 delay(5);                      /* allow some time for keypad to be read */
```

```
if(active_request == true && key == 'A'){      /* if a convoy request is
          active and master pressed 'A',
                                                  send a confirmation char
          'A' and wait for echo */
  /* Udp.beginPacket(slave, localPort); */      /*inactive backup
          connection setup */
  Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());/*create UDP packet */
    Udp.write('A');                                 /* write 'A' to it */
    Udp.endPacket();                                  /* send packet */
  timing_out_request = false;             /* request timeout is passed */
  timing_out_ack = true;           /* allow timeout for the echo of 'A' */
  start_time_ack = millis();                          /* start timing */
  digitalWrite(LED_1, HIGH);/* indicate that master has accepted convoy */
}
if(active_convoy == true && key == 'C'){  /*if master close the convoy */
   /* Udp.beginPacket(slave, localPort); */      /*inactive backup
          connection setup */
   Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());    /*create UDP
          packet */
     Udp.write('C');                                /* write 'C' to it */
     Udp.endPacket();                                     /* send packet */
   active_convoy = false;                           /* close the convoy */
   compute_time = false;/* do not compute time when convoy is inactive */
}
/*-------------READ JOYSTICKS-------------------------------*/
int steering_angle = analogRead(VRx); /* read right joystick x-values */
delay(2);
int motor_speed = analogRead(VRy);    /* read left joystick y-values */
delay(2);

/*joysticks are center positioned when not used, it benefits the steering
          to automatically steer back to center when
  released. However it is problem for accelerator joystick. When center
          positioned, the y-value is half way to max which
  would make the motor to go on half speed if directly applied. The values
          must therefore be mapped so center positon is
  equal to zero for the motor. For the left joystick, the center position
          to top position represents values 512-0.
  The mapping should therefore be inverted with 512 as lower value and 0
          as higher value. To prevent false reading of joystick
  or just accidently touching of joystick, the lowest value is decreased
          to 500. This allows a small "dead-zone" for the joystick.
  */

steering_angle = map(steering_angle, 1023, 0, 40, 140);  /* map joystick
          values to proper servo angle values (limit: 0-180) */
```

```
if(motor_speed < 500)
  motor_speed = map(motor_speed, 500, 0, 0, 255); /* map to proper motor
          values */
else
  motor_speed = 0;                    /* if joystick is moved down from
          center position, it should not affect the motor speed */

digitalWrite(BRAKE_A, LOW);
analogWrite(PWM_A, motor_speed);
servo.write(steering_angle);
delay(2);

if(compute_time){
  end_time_request = millis();          /* clock the current time for the
          request convoy timeout variable */
  if(end_time_request - start_time_request > time_out_limit &&
          timing_out_request == true)
    time_out = true;                    /* if 10 seconds has passed
          without master accepted convoy, timeout! */
  if(end_time_ack - start_time_ack > time_out_limit && timing_out_ack ==
          true)
    time_out = true;                    /* if 10 seconds has passed
          without the acknowledged echo from accepting has returned,
          timeout! */
  if(end_clock_broadcast - start_clock_broadcast >= broadcast_time &&
     end_clock_broadcast - start_clock_broadcast < time_out_limit){
     broadcast_LED = true;              /* if time between two broadcast
          messages was within edge value range, make LED flash */
     led_start_time = millis();         /* clock time for how long LED
          should light up during a flash */
  }
  else if(end_clock_broadcast - start_clock_broadcast > time_out_limit)
    time_out = true;                    /* if broadcast waiting time has
          been over limit, make a timeout */
}
if(time_out == true){                    /* if timeout is in action */
  if(timing_out_request == true){
    active_request = false;      /* if timing out was due to slow
          aknowledge of request, reset request variable */
    timing_out_request = false;
  }
  if(timing_out_ack == true){
    active_request = false;      /* if timing out was due to slow/loss UDP
          ack char 'A', reset request variable */
    timing_out_ack = false;
  }
```

```
  if(timing_out_broadcast == true){
    active_convoy = false;       /* if timing out was due to no broadcast,
          close the convoy and reset timing out variable */
    timing_out_broadcast = false;
  }
  time_out = false;              /* reset time out variable */
  compute_time = false;          /* quit compute time */
}
if(broadcast_LED == true){                /* make LED light up during a
          second when flashing */
  digitalWrite(LED_2, HIGH);
  led_end_time = millis();
  if(led_end_time - led_start_time >= 1000){
    digitalWrite(LED_2, LOW);
    broadcast_LED = false;
  }
}

/*------READ ETHERNET DATA-----------------------------------------*/
int packets_available = Udp.parsePacket();
if(packets_available){
  char packet = Udp.read();
  switch (packet){
   case 'A':                    /* if received acknowledge echo 'A' */
    active_convoy = true;       /* activate convoy */
    active_request = false;     /* reset request convoy variable */
    timing_out_ack = false;    /* reset acknowledge timing out variable */
    digitalWrite(LED_1, LOW);   /* turn off LED_1 */
    break;

   case 'B':
     /* broadcasting from slave vehicle every fourth second */
     start_clock_broadcast = end_clock_broadcast;
     end_clock_broadcast = millis();
     break;

   case 'C':
     /* Udp.beginPacket(slave, localPort); */      /*inactive backup
          connection setup */
     Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  /* create UDP
          packet */
       Udp.write('C');                                /* write 'C' to it */
       Udp.endPacket();                                   /* send packet */
     break;
```

```
    case 'R':
      active_request = true;                  /* got join convoy request */
      start_time_request = millis(); /* start timeout countdown variable */
      timing_out_request = true;  /* set timing out to be due to request */
      compute_time = true;                /* allow time to be computed */
      digitalWrite(LED_1, HIGH); /* turn on LED_1 and LED_2 for a second */
      digitalWrite(LED_2, HIGH);
      delay(1000);  /* one second delay is okey for the timing out range */
      digitalWrite(LED_1, LOW);
      digitalWrite(LED_2, LOW);
    }
  }
  if(active_convoy == true){
    /*---------------------WRITE ETHERNET DATA---------------------*/
    sprintf(command_buffer_char, "#servo=%d,motor=%dE\n", steering_angle,
            motor_speed);  /*write motor speed and servo degree values
            to the buffer that then sends away */
    for(current_index = 0; current_index < UDP_TX_PACKET_MAX_SIZE;
            current_index++){     /* UDP_TX_PACKET_MAX_SIZE is default
            maximum buffer size, set in EthernetUDP.h */
      command_buffer_byte[current_index] =
            (byte)command_buffer_char[current_index];     /* convert buffer
            values to byte before send */
      if(command_buffer_char[current_index] == 'E')
            /* last char in buffer */
        break;
    }
    /* Udp.beginPacket(slave, localPort); */        /*inactive backup
            connection setup */
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());/*create UDP packet */
      Udp.write(command_buffer_byte, UDP_TX_PACKET_MAX_SIZE);
            /* write buffer to it */
      Udp.endPacket();                              /* send packet */
  }
}
```

Code listing 3.4

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <String.h>
#include <Servo.h>
#include <Keypad.h>
#include <Wire.h>

const int
PWM_A = 3,      /*pin that controls the pulse width modulation value of the
            motor */
DIR_A = 12,     /*pin that sets the direction of the motor*/
BRAKE_A = 9,    /*pin that controls whether the brake is on or off on the
            motor*/
trigPin = 48,    /*pin that trig the signal of the USDR sensor*/
echoPin = 49,    /*pin that listen to the echo signal of the USDR sensor*/
Joystick_x_axle_pin = A8,        /*pin that reads the X-axle values on the
            right joystick */
Joystick_y_axle_pin = A9,        /*pin that reads the Y-axle values on the
            left joystick */
LED_1 = 40,         /* pin for LED. Represents Ethernet initializing */
LED_2 = 41;         /* pin for LED. Represents active convoy */

const byte ROWS = 4;
const byte COLS = 4;

/*matrix that represents the 4x4 keypad*/
char keys[ROWS][COLS] =
  {{'1','2','3','A'},
   {'4','5','6','B'},
   {'7','8','9','C'},
   {'*','0','#','D'}};
byte rowPins[ROWS] = {30,31,32,33};    /*match the row pins on the matrix
            with Arduno pins*/
byte colPins[COLS] = {37,34,35,36};    /*match the column pins on the
            matrix with Arduino pins*/
//create a keypad object by the 4x4 matrix*/
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

/* ACC variables */
int acc_sum = 0;
int default_acc_distance = 30;
int previous_acc_distance = 30;
int current_acc_distance = 30;
```

```
long measured_distance = 0;
boolean set_acc_distance = false;
boolean set_acc_speed = false;
boolean acc = false;
int current_acc_speed = 0;
int reference_acc_speed = 0;

Servo servo;

/* Ethernet variables */
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xDD};
IPAddress ip(192, 168, 0, 11);
/* IPAddress master(192, 168, 0, 10);*/
unsigned int localPort = 8888;
EthernetUDP Udp;
char command_buffer_char[UDP_TX_PACKET_MAX_SIZE];  /*buffer for commands*/
int current_char;                          /*loop variable for use when
           parsing the receiving buffer from master */

/*Convoy variables*/
boolean active_convoy = false;
boolean receive_data = false;
int received_motor_speed = 0;        /* received value from sscanf */

int wifi_speed = 0;
int wifi_steering = 95; /* default servo angle for center position */
byte servo_value, motor_value;
int store_steer_degree = 95;
int received_steering_angle = 0;      /* receive value from sscanf */

/* time variables */
unsigned long start_time_request, start_time_cancel, start_broadcast_time,
           end_time_request, end_time_cancel, end_broadcast_time,
           time_out_limit = 10000, broadcast_time = 4000;
boolean time_out = false, timing_out_request = false, timing_out_cancel =
           false, compute_time = false;
/*-----------------------------------------------------------------------
           ----------------------*/
void setup(){
  /*setup of LED and joysticks as output or input, and attach servo motor*/
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(Joystick_x_axle_pin, INPUT);
  pinMode(Joystick_y_axle_pin, INPUT);
  servo.attach(24);
```

```
  Serial.begin(9600);          /* allow Serial connection when debugging or
            reprogramming with computer */

  /* Ethernet shield and connection setup */
  int i;
  for(i = 0; i < 50; i++){  /*50 second delay to enable the access point to
            start up*/
    delay(1000);
  }
  digitalWrite(LED_1, HIGH);  /*turn on LED1 for indicating the ethernet
            connection is initalizing*/
  Ethernet.begin(mac, ip);
  Udp.begin(localPort);
}
/*--------------------------------------------------------------*/
void loop(){
  /*-----------READ 4x4 KEYPAD---------------------------------*/
  char key = keypad.getKey();
  /*--------------READ JOYSTICKS-----------------------------------*/
  int x_axle_value = analogRead(Joystick_x_axle_pin);  /* read value on x-
            axle pin */
  delay(2);   /* delay 2ms for value to be read properly */
  int y_axle_value = analogRead(Joystick_y_axle_pin);  /* read value on y-
            axle pin */
  delay(2);   /* delay 2ms for value to be read properly */
  int servo_steering_value, motor_speed_value;          /* define values
            for later appliance */

  /*read values on joystick pins are values of 0-1023 but due to how I
            mounted these joysticks on controller,
    the values will be read in negative way. The map-function normally has
            input as:
     (source, fromLowest, fromHighest, toLowest, toHighest) which means I
            should normally have used it like
     (source, 0, 1023, toLow, toHigh) but I had to use it like (source,
            1023, 0, toLow, toHigh).
    Since the joysticks are center-positioned when not used, I had to
            filter out the values from the furthest down
    to the center posistion which was, due to the negative reading, values
            512-1023. I included a dead-zone of the
    accelerator joystick from 500-512 to remove the accidently reading of
            the joystick when center posistioned because
    then values would vary between 510-515. */

  servo_steering_value = map(x_axle_value, 1023, 0, 40, 140);  /* map raw
            joystick values to servo degree */
```

```
if(y_axle_value < 500)
  motor_speed_value = map(y_axle_value, 500, 0, 0, 255);      /* map raw
         joystick values to motor speed */
else
  motor_speed_value = 0;

/* if convoy is inactive, set speed either acc speed if acc is activated
         or the read value from joystick and set servo degree angle to
         read joystick value.
   If convoy is active, set speed and servo degree angle to read values
         from master via Wi-Fi.
   Measure distance to obstacle if acc is activated or convoy is
         activated, since the speed is not controlled with joysticks */

if(active_convoy == false){
  if(acc == false){
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, motor_speed_value);
    servo.write(servo_steering_value);
    delay(2);
  }
  else{
    if(measured_distance < 5)/* very short distance to osbtacle, stop! */
      current_acc_speed = 0;
    else if(measured_distance < 10)                 /* fairly short
         distance, decelerate hard */
      current_acc_speed -= 30;
    else if(measured_distance < current_acc_distance)  /* distance to
         osbstacle shorter than set distance, decelerate */
      current_acc_speed--;
    else if(measured_distance > current_acc_distance && current_acc_speed
         < reference_acc_speed)  /*distance further than set, */
      current_acc_speed += 5;                           /* accelerate */
    if(current_acc_speed < 0)                           /* lower limit */
      current_acc_speed = 0;
    if(current_acc_speed > 255)                        /* higher limit */
      current_acc_speed = 255;

    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, current_acc_speed);
    servo.write(servo_steering_value);
    delay(2);
  }
}
else{
```

```
  if(measured_distance < 5)
    wifi_speed = 0;
  else if(measured_distance < 10)
    wifi_speed -= 50;
  else if(measured_distance < default_acc_distance)
    wifi_speed -= 20;
  if(wifi_speed < 0)
    wifi_speed = 0;
  digitalWrite(BRAKE_A, LOW);
  analogWrite(PWM_A, wifi_speed);
  servo.write(wifi_steering);
  delay(2);
}
delay(12); /* Intentionally delay for proper ULTRASOUND SENSOR USE */

/*---------------- READ ULTRASOUND SENSOR ---------------*/
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long dur = pulseIn(echoPin, HIGH);
measured_distance = microsecondsToCentimeters(dur);
delay(12); /* Intentionally delay for proper ULTRASOUND SENSOR USE */

/*--------------------HANDLE KEYPAD BUTTONS-----------------------*/
switch (key){
 /*ACC ON*/
 case 'A':                                          /* key is A */
  if(active_convoy == false && acc == false){    /* check if convoy is
         inactive and acc is off */
   acc = true;                                       /* turn acc off */
   current_acc_speed = motor_speed_value;         /* save current speed
         to acc speed */
   reference_acc_speed = current_acc_speed;    /* save reference copy */
  }
  else if(active_convoy == false && acc == true){ /* check if convoy is
         inactive but acc is on */
    acc = false;                                     /* turn acc off */
  }
  break;
 /* "JOIN CONVOY"/"ABORT CONVOY" - REQUEST */
```

```
case 'B':                                        /* key is B */
   if(active_convoy == false){
     /* Udp.beginPacket(master, localPort); */
     Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  /*create UDP
         packet */
       Udp.write('R');                             /* write 'R' to it*/
       Udp.endPacket();                            /* send packet */
     digitalWrite(LED_2, HIGH); /* and turn on LED 2 to inform driver */
     start_time_request = millis();               /* clock time and save
         it to start_time variable */
     compute_time = true;              /* allow computing time */
     timing_out_request=true; /*turn on timing out for joining request*/
   }
   else{                                      /* if convoy is active */
     /* Udp.beginPacket(master, localPort); */
     Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());     /* create
         UDP packet */
       Udp.write('C');                        /* write 'C' to it */
       Udp.endPacket();                       /* send packet */
     start_time_cancel = millis();                              /* clock
         time and save it to start_time variable */
     timing_out_cancel=true; /*turn on timing out for canceling convoy*/
   }
   break;

case '#':
   /* when acc is on (only when convoy is inactive) and currently not
         already setting speed,
     set speed parameter to true */

   if(acc == true && active_convoy == false && set_acc_speed == false){
     set_acc_speed = true;
   }

   else if(acc == true && active_convoy == false && acc_sum > 0 &&
         set_acc_speed == true){
     if(acc_sum >= 100)                /* when acc is turned on and
         input speed is higher or equal to 100, */
      current_acc_speed = acc_sum;             /* save it for applying
         to current speed (100 is just a feasible minimum speed)*/
     acc_sum = 0;                      /* reset incrementing value */
     set_acc_speed = false;            /* reset current setting speed
         state to false */
   }
   break;
```

```
  case 'D':
    /* like speed setting on acc, the distance setting method work the
         same */
    if(acc == true && active_convoy == false && set_acc_distance ==
         false){
      set_acc_distance = true;
    }
    else if(acc == true && active_convoy == false && acc_sum > 0 &&
         set_acc_distance == true){
      if(acc_sum > 20)                  /* feasible minimum distance
          between vehicles */
        current_acc_distance = acc_sum;
      else
        current_acc_distance = 20;      /* if set to lower, the default
          value will replace the input */
      acc_sum = 0;                     /* reset the incrementing value */
      set_acc_distance = false;         /* reset current setting distance
          state to false */
    }
    break;

  case '0':    /* check digits */
  case '1':
  case '2':
  case '3':
  case '4':
  case '5':
  case '6':
  case '7':
  case '8':
  case '9':
    int digit = (key - '0');            /* change from char to integer */
    acc_sum = digit + (acc_sum * 10);   /* replace the current acc_sum
         value depending on input */
/* the replacing is active so its happening after every button press */
/* and not calculated after every specifik buttons have been pressed */
/* works for 1-5 digits. Example: */
/* input = 5 -> */
/* acc_sum = 5 + (0*10) = 5 */
/* input = 120 -> */
/* acc_sum = 1 + (0*10) = 1  ->  */
/* acc_sum = 2 + (1*10) = 12  -> */
/* acc_sum = 0 + (12*10) = 120 */
}
delay(10); /* Intentionally delay for proper USDR SENSOR use */
```

```
/* compute different time-outs */
if(compute_time == true){ /* computing time only when convoy is active */
  end_time_request = millis();            /* get current time in
        milliseconds from system boot */
  end_time_cancel = end_time_request;
  end_broadcast_time = end_time_request;
  if(end_time_request - start_time_request > time_out_limit &&
        timing_out_request == true)
    time_out = true;                    /* if 10 seconds is reached
        without acknowledge the join request, time out! */
  if(end_time_cancel - start_time_cancel > time_out_limit &&
        timing_out_cancel == true)
    time_out = true;                    /* if 10 seconds is reached
        without acknowledge the cancel request, time out! */
  if(end_broadcast_time - start_broadcast_time > broadcast_time &&
        active_convoy == true){
    /* Udp.beginPacket(master, localPort); */
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());   /* create UDP
        packet */
      Udp.write('B');                       /* write 'B' (Broadcast) */
      Udp.endPacket();                                  /* send
        broadcast packet every fourth second in active convoy */
    start_broadcast_time = end_broadcast_time; /* reset time variable */
  }
}
if(time_out == true){                   /* if time out is in action */
  active_convoy = false;                /* deactivate convoy */
  if(timing_out_cancel == true)
    timing_out_cancel = false;          /* if timing out was due to
        cancel request, reset its variable */
  if(timing_out_request == true)
    timing_out_request = false;         /* if timing out was due to
        joining request, reset its varable */
  compute_time = false;   /*don't compute time when convoy is inactive */
  time_out = false;          /* reset time out after this block */
}

/*---------------RECEIVE FROM MASTER-------------------------------*/
int packets_available = Udp.parsePacket();   /* check if there are
        packets waiting for reading */
if(packets_available){
  char packet = Udp.read();                  /* read a character */
  switch (packet){
    case 'A':       /* if we read 'A', master has accepted convoy */
```

```
        if(active_convoy == false){             /* only if convoy was
            inactive before reading 'A' */
          /* Udp.beginPacket(master, localPort); */
          Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  /* create UDP
            packet */
            Udp.write('A');               /* write 'A' to packet */
            Udp.endPacket();              /* send UDP packet away */
          active_convoy = true;           /* activate convoy */
          compute_time = true;            /* allow time to be computed */
          start_broadcast_time = millis();       /* clock the broadcast
            time variable */
          digitalWrite(LED_1, LOW);  /* turn off LED_1 and turn on LED_2 */
          digitalWrite(LED_2, HIGH);
        }
        break;

      case 'C':      /* if we read 'C', master sent slave "close convoy" or
confirmed slave's close request */
        if(active_convoy == true){ /* only when convoy is already active */
          active_convoy = false;                  /* close convoy */
          digitalWrite(LED_2, LOW);               /* turn off LED_2 */
          if(timing_out_cancel == true)           /* if master confirmed
            slave's close request, */
            timing_out_cancel = false;  /* reset timing out variable */
          compute_time = false;               /* cancel computing time */
        }
        break;

      case '#':                     /* indicate receiving a buffer */
        Udp.read(command_buffer_char, UDP_TX_PACKET_MAX_SIZE);  /*read
            buffer */

        /* scan buffer into servo angle and motor speed variables */
        sscanf(command_buffer_char, "servo=%d,motor=%dE",
            &received_steering_angle, &received_motor_speed);
        wifi_speed = map(received_motor_speed, 0, 255, 0, 200);  /*map
            motor speed to proper physical speed according to master */
        wifi_steering = received_steering_angle;



        /* BACKUP (INSIGNIFICANT CODE FOR NOW)
        for(current_char = 0; current_char < Maximum_command_length;
            current_char++){
          char read_packet = Udp.read();
          command_buffer_char[current_char] = read_packet;
```

```
          if(read_packet == 'E'){
            sscanf(command_buffer_char,
            "steering_angle=%d,motor_speed=%dE", &received_steering_angle,
            &received_motor_speed);
            wifi_speed = map(received_motor_speed, 0, 255, 0, 200);
            wifie_steering = received_steering_angle;
            break;
          }
        }
        */
    }


  }
  delay(20);
}
//------------------CONVERT USDR's TIME TO DISTANCE------------------
long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

This section of appendix shows the statemachines represented by each code listing.

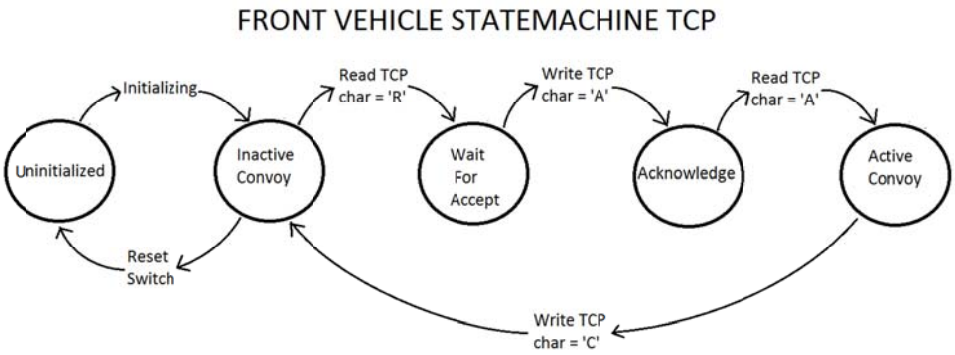


Figure 6-1: Statemachine of front vehicle with TCP

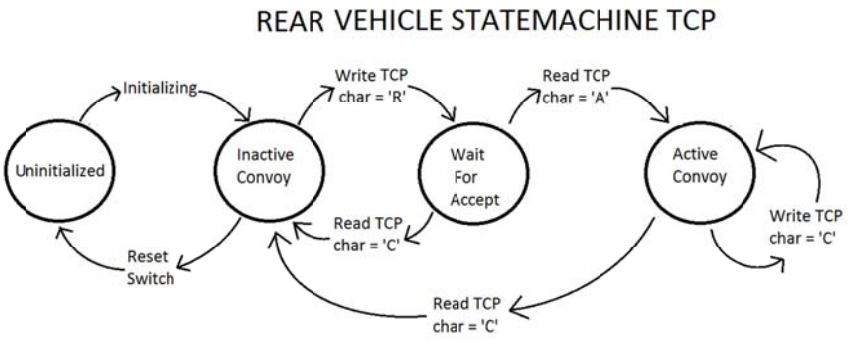


Figure 6-2: Statemachine of rear vehicle with TCP

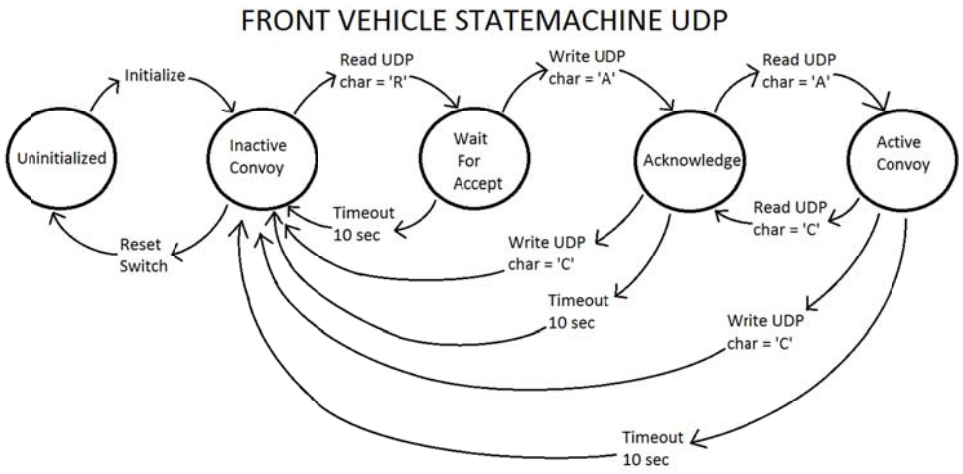


Figure 6-3: Statemachine of front vehicle with UDP

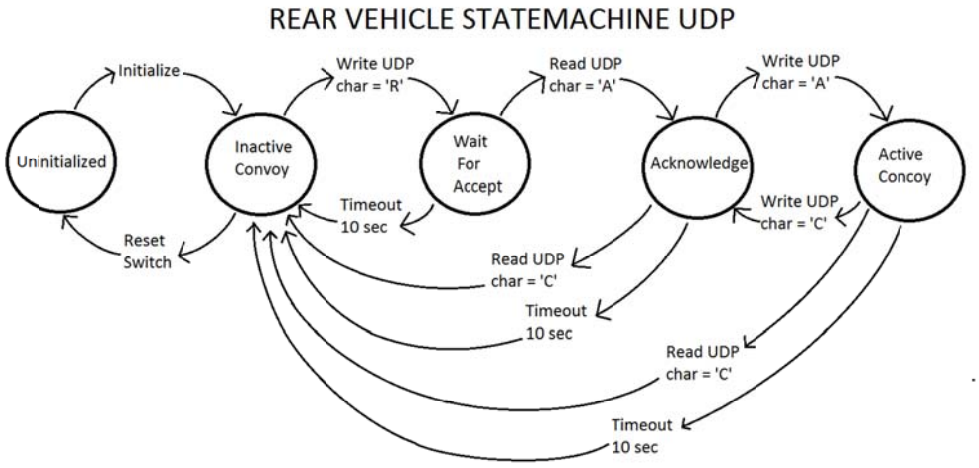


Figure 6-4: State machine of rear vehicle with UDP

TRITA-ICT-EX-2014:96