DATA3900 - Bachelorproject
Final report

Gruppe 23:
Hansen, Andreas Torres (s338851)
Nguyen, Uy Quoc (s341864)
Ottersland, Anders Hagen (s341883)

Total pages: 23

Last updated:
May 11, 2022

Abstract

Short summary of the project including the result that the group reached.

Preface

This is the report of our bachelor thesis at Oslo Metropolitan University, Faculty of Technology, Art and Design. Our project was done for Accenture, and lasted from January 2022 to May 2022. We have tried to develop a solution for traffic management with self-driving cars and server communication. A physical demonstration of our proposed solution is done with the use of a Raspberry Pi computer working as a car. In this project we have documented the functionality of our system, and our process of making it.

The report is split into 7 chapters. Introduction, research areas, process documentation, implementation, results, discussion and conclusion. At the end we have added a Moscow analysis, sprint overview and our project journal as appendices. Technical terms are in appendix for those with less technical knowledge. (Kan kanskje droppe dette)

We would like to thank everyone that has contributed to our project. We would especially like to thank:

- Ivar Fauske Aasen, Solfrid Johansen and Benjamin Vallestad, representatives from Accenture.
- Dr. Jianhua Zhang, internal supervisor at OsloMet.

Contents

Introduction

1.1 Stakeholders

1.1.1 Students

Andreas Torres Hansen, Software Engineering

Anders Hagen Ottersland, Software Engineering

Uy Quoc Nguyen, Software Engineering

1.1.2 Product owner

Benjamin Vallestad

1.1.3 Supervisors

Professor Jianhua Zhang, Internal Supervisor

Ivar Austin Fauske, External Supervisor

Solfrid Hagen Johansen, Application Development Analyst at Accenture AS, External Supervisor

1.1.4 Client

Accenture AS

Accenture is an international IT consulting firm that operates in 200 different countries across the world. The main office is located in Dublin, and the Norwegian main office is in Fornebu. Accenture has 674 thousand employees internationally, of which 1000 work in Norway (Accenture, 2021). In 2021, Accenture generated a revenue of approximately $50.3 billion (Accenture, 2022).

1.2 Project Description

1.2.1 Project background

In 2020, Accenture supervised a bachelor student-group with the goal of developing a self-driving car. The group developed a model-sized car that uses artificial intelligence to analyze data from sensors and cameras, and makes decisions based on the data. Accenture wants to build further on this project, by exploring the addition of a centralized communication system for the vehicles.

The product owner of the project states various reasons that make it relevant for Accenture. According to Accenture, Norway is one of the countries that is most ready to start utilizing self-driving cars, although not all factors that need to be in place are ready. Self-driving cars alone will also likely not be able to solve all the problems related to today's traffic-problems. Accenture wants to explore if combining self-driving cars with an autonomous management system could provide greater value.

==================

Accenture has in many years offered final year students at OsloMet and Høyskolen i Kristiania interesting projects for their bachelor thesis. In 2020, a group of students from Høyskolen i Kristiania were developing model-sized self driving vehicles using Raspberry Pi in conjunction with machine learning as their project. This year our group was offered to extend this project further; to explore plausible improvement with the addition of a centrilized communication system.

Norway is one of the countries that are ready to utilize self-driving cars, according to Accenture. However, self-driving cars alone is likely not enough to solve all of today's traffic challenges. Hence, the purpose of this project is an attempt to solve the issue by introducing a management system for autonomous vehicles and evaluate the value such system can provide.

1.2.2 Significance

Accenture's values coincide with the project, seeing as they wish to build their knowledge in the form of technology and theory. Accenture also wants to explore the potential positive societal and climate effects a centralized communication system for transportation could provide.

======================

1.2.3 Goals and requirements

The goal of the project is to end up with a prototype that can be shown to interesents and demonstrate how self-driving cars could be combined with a centralized system. For this we will use the self-driving car that Accenture already owns, as a result of the previous bachelor's-thesis done with them. The conditions are that the product must be possible to present, either as a physical or digital showcase. Preferably, the prototype can show at least one situation where the outcome differs depending on the use of self-driving cars plus a centralized system, versus only using self-driving cars. The system should also be scalable, so that more vehicles can be added or removed at a later point in time.

1.2.4 Problem statement

Based on the goals and requirements set by Accenture, we have formulated the thesis question:

> "How can we improve traffic flow, by using a combination of self-driving cars and a centralized communication system?"

Research areas

2.1 Internet of Things (IoT)

The Internet Of Things refers to physical objects that communicate with the use of sensors, cameras, software or other technologies that connect and exchange data. This communication takes place over the Internet or other communication forms. The number of connected IoT-devices in the world is increasing, and it is becoming a big part of society (Sinha, 2021). The field of IoT has also been evolving in recent years due to other technologies becoming more accessible, such as machine learning.

You need not look further than to the smart-home consumer market to find applications in your life, of devices communicating to solve problems. It could also be applied in climate surveillance systems, energy or transportation. The benefits that an internet of connected devices could add billions in value to industries across the world, and to the global economy. In this thesis we will explore the possibilities of using IoT in transportation, more specifically in personal automobiles. The convergence of these fields is more commonly known as IoV, Internet of Vehicles, and it is a central theme of our thesis. An IoV system is a distributed system for wireless communication and information exchange between vehicles through agreed upon communication protocols (Chinese APEC Delegation, 2014). The system could potentially integrate functionality for dynamic information exchange, vehicle control and smart traffic management. In our thesis we will explore these possibilities on a small scale, with the hopes of making a solution that can be scaled up at a later point.

Challenges to be aware of with IoT-systems, and consequently also IoV-systems include, among others, ethical questions about decision-making, physical and digital safety for humans and infrastructure, storage of data and power usage. With the evolution, and increased availability of artificial intelligence (AI), IoT has also evolved to implement algorithms powered by AI. We will discuss the implementation of AI and machine learning further down in the thesis, but it is possible to view some of the problems with IoV-systems as an extension of typical problems with AI systems. In our work with the thesis we have kept these challenges in mind when formulating proposed solutions, and it has affected how we have worked.

2.2 Artificial Intelligence (AI)

Artificial intelligence(AI) leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind. At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving (IBM Cloud Education, 2020). There are two types of Ai, weak Ai and strong Ai. Weak AI, also known as narrow AI, is trained to perform specific tasks such as, voice recognition, data labeling, and in our case, driving cars. Narrow AI is the most common type of AI, and the type used in today's AI systems.

Strong AI is made up of Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI) (IBM Cloud Education, 2020). General AI is a theoretical form of AI where a machine would have reached the same level of intelligence as a human being. Artificial Super intelligence is a theoretical form of AI where the machine has surpassed the intelligence of humans. This type of AI is often presented in movies.

2.3 Machine learning

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior (Brown, 2021).

There are three types of machine learning, supervised learning, unsupervised learning and reinforcement learning. In supervised machine learning the algorithm is given labeled data to help predict outcomes. This means the algorithm uses already historical data. Supervised learning is the most commonly used machine learning. Supervised learning can be separated into 2 different problems when data mining, classification and regression (Delua, 2021). Classification uses an algorithm that assigns test data into the right categories. For example, differentiate between cats and dogs. Regression is another type of supervised learning method that uses an algorithm to understand the relationship between dependent and independent variables (Delua, 2021). Regression is used for predicting mathematical values such as stocks and sales. The prior group used supervised learning to train their camera detection model.

However in unsupervised learning the algorithm uses non-labeled data. The algorithm looks for patterns or trends in the data. This can be used for labeling data or figuring out trends. Customers that buy item x will be more likely to buy item y as well.

Reinforcement learning trains the machine via trial and error and a reward system. This is commonly used to train machines in games or autonomous vehicles. An example is Stockfish 14, one of the best chess engines in the world. (kilde)

About four of five traffic accidents are caused by human error (Ntb, 2002). And in fields that require a specific task to be done, for example chess ai, already surpasses humans. This means if an learns to drive perfectly, four out of five traffic accidents can be avoided. However driving is a complex task that requires a ton of data, and decision making. In addition, for an Ai to give the most accurate outcome a lot of accurate test data are required. Nearly 43% of people in the US are afraid of self-driving cars (Kopestinky, 2022). Giving a self-driving car an accurate driving environment without risking human lives is therefore a challenge.

2.4 Self driving cars

Autonomous vehicles also known as self-driving cars are becoming more and more relevant. We see huge companies such as Tesla and Ruter investing into research. Ruter has implemented a short route that is part of Ruter's transport system where there are only self driving buses. Even though it's unclear exactly when fully automated vehicles will hit the roads it seems like the day is getting closer and closer. It's impossible to predict exactly what's going to occur in a traffic scenario. That's why the use of AI is prefered compared to traditional computing in this setting.

The Society of Automotive Engineers (SAE) currently defines 6 levels of driving automation ranging from Level 0 (fully manual) to Level 5 (fully autonomous) (Synopsys, n.d.). The self driving buses Ruter has implemented is at the third level of automation. This means that there is a driver present to take over if something goes wrong. In this project we will mainly focus on the 4th level of automation. In the fourth level of automatization there is no driver present and the car is fully automated in a specific setting, in our case, a prebuilt track. What differentiate the fourth and the fifth level is that in the fifth level the vehicles can drive in any setting.

2.5 Ethics and security

For an Ai to be able to train there will be a need for training data. Therefore a database saving information about the trips will be needed, in the IoT-system, for the AI to further improve.

When there is saving of data involved it is important to consider personal information. Personal information typically includes name, address, phone number, email and date of birth (Datatilsynet, 2019). In addition it also includes behavioral patterns, for example where you drive to work each day or where you go shopping. In this case, an IoT system that saves the car's trips it's considered saving of personal information. The system could try to anonymize the data, but there will still be possibilities to find patterns in the data. You are allowed to save some personal information as long as you follow certain rules.

Hackers are a problem when it comes to systems controlling entities that can affect human lives. If hackers were able to get control of such a system it could lead to a lot of damage. It is therefore important for a secure security system to be in place. If anyone were to breach the system, it would have to detect that it has been breached, and shut down so the cars can move on their own without the system.

Process documentation

3.1 Development method

The task given to us had few restrictions, furthermore the task was open. This meant that work methodology had to focus more on flexibility rather than planning, it had to be agile. Agile work method focuses on continuous planning throughout the process, and having frequent communication with the client, in our case Accenture. We had meetings with our external supervisors from Accenture once every second week which meant the agile work was a good fit for our project.

We chose to use inspiration from two light frameworks, kanban and scrum. Scrum is an agile, light framework which helps people and teams work together. Scrum describes a set of meetings, roles and tools.

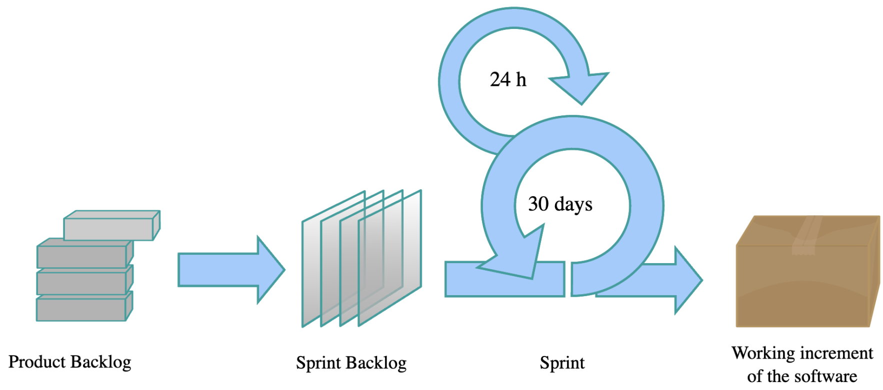

Figure 3.1: Illustration of a generic scrum process.

A sprint is an essential part in using the Scrum framework. Sprints are a fixed time length, often between one and four weeks. In this specified time length the teams do tasks assigned from the sprint backlog . Each sprint starts with a sprint planning and ends with a sprint retro perspective. For our project we found it most viable to plan in increments of two weeks. We chose 2 weeks increments because it was a good balance between work and planning.

Sprint planning is a meeting or event which starts before a sprint. At the sprint planning the teams agree on a goal for the sprint, and the tasks from the backlog that should be worked on that contribute to the goal. Example of report from a sprint planning meeting is attached (Add an attachment from one of our sprint planning meetings). The backlog is a list of functionality the product should contain. In addition we wrote down the tasks for the specific sprints in a document. These tasks were to be finished by the next sprint. After a sprint we would have a sprint retro perspective where we would discuss what went well in that sprint, and what could have been done better. (Reference to an attachment with a picture of a sprint retro perspective meeting).

In addition to the weekly meetings we also had daily standups. Daily standups is a short meeting, usually around five minutes, where each person answers three questions. What did you do last time? What are you doing today? Are there any challenges? We implemented daily standups because it helped our team get on the same page, and it made it easier to plan what each of us had to do that specific day.

Scrum often consists of a team with different roles. As a team of three where we did not feel the necessity to have specified roles, because we usually worked together on our project. However we alternated on being the scrum master. The scrum master's responsibility is to keep track of the backlog and lead the sprint planning meetings.

Our implementation of Kanban was to use a Kanban board as the backlog. A Kanban board is used to visualize where a task is in the work process. Here is an example from our project:

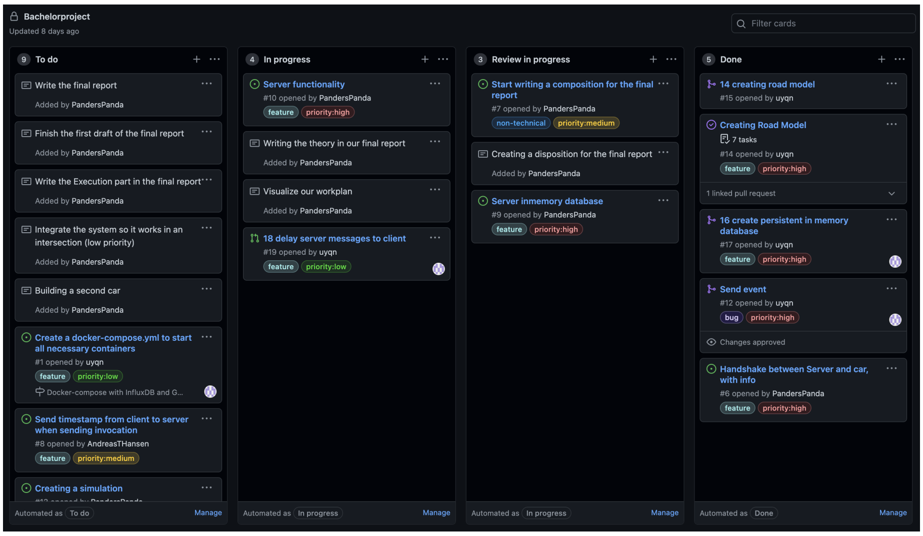

Figure 3.2: Extract of our kanban from Github.

We have four columns that represent which phase a task is in. The backlog is the tasks in the to-do column. When we are working on a task we drag it over to the In progress column. After a task is finished it goes to the "Review in progress", where we test it. If the testing is a success it gets dragged over to the “Done column”. The Kanban board was a great tool to see which tasks to choose for our sprints, and also keep track of where the tasks were in their process.

3.2 Tools and technology

The circumstances surrounding Covid 19 ment that we were not able to meet our supervisors in person at the start, however many of the restrictions cleared up at the later part of the project. Luckily the group was still able to meet physically a few times a week. We still had to use a wide range of tools for communication.

- Email - Formal communication with supervisors and product owner
- Teams- Meetings, and the platform of choice for communicating with the external supervisors on an informal level.
- Zoom- Meetings
- Messenger - communication internally in the group. We used it to send messages to each other when we were not physically together, and to send pictures of code.

The project required us to collaborate while working on different personal computers which can lead to overlapping. Therefore, tools that helped us to work on the project together were important. Here is the tools that helped us:

- Git and Github - Version control of choice
- Google docs - Used to write our journal and other documents that need to get updated regularly, and to share documents.

We also needed text editors that supported our programming languages that we used. The client was built in python while the server was built in C#.

- Pycharm- Text editor for coding in python
- Visual studio code- Text editor for coding in C#
- Thonny - Text editor for coding in python on Raspberry pi

Project planning and documentation was also an important part of the project. Here is a few tools we used for the project planning:

- Github project- Kanban board and creating backlog tasks
- Excel- Used for visualizing our worklan by using tables and a gantt diagram

Implementation

4.1 Pre-project phase

After we had gotten in touch with Accenture and spoken with the supervisors and the product owner, the group had to make a few decisions regarding the direction of the project.

The first choice to make was to either build an AI for the vehicles from scratch or use the AI from the group prior. Building the AI from scratch ment that we could make an AI which integrated our system from the start, however with the time constraint of the project we decided that it would take less time to use the product from the group prior. In addition, the group had more prior experience with networking than with raspberry Pi and AI. We therefore chose to use the AI from the project prior.

During the pre-project phase there was a need for some project planning. During this planning phase we made a brief project plan. The plan contained a backlog with tasks and the estimated time period we would have to work on that current task, and with that a gantt-diagram containing those tasks.

We also used the pre-project phase to get to know Accenture, their guidelines and their workspace.

4.1.1 Choice of programming languages

For the programming languages we used python for the client and C# for the server. The group prior had used python for their raspberry Pi car. This meant that using python made it easier to extend the code from that project. The group also had prior experience with networking in python, therefore making python a prime choice of programming language for the client. C# supports multiple ASP.NET Core libraries which was useful since we were going to work with networking. We also had some prior knowledge coding in C# as well.

4.2 Implementation of Server and Client

As IoT-systems become more complex, there is a need to structure them in layers. The three fundamental layers to an IoT-system consists of the perception layer, network layer and application layer (Zhu et al., 2010, pp. 347-352). In this part we will discuss what our things-layer and network-layer consists of, and explain the theory behind our decisions.

4.2.1 Implementation of Client

Raspberry Pi is a small single-board desktop computer that is commonly used for IoT-projects. There is an enormous ecosystem of compatible devices that allow these computers to interact with the world in various ways. The organization states that they can be used for everything “from music machines and parent detectors to weather stations and tweeting bird houses with infra-red cameras” (Raspberry Pi, n.d.). For our project we have used a Raspberry Pi Model 4, which the previous group used in their project. This is the newest and fastest model, which makes our test results as accurate as possible.

The perception layer is responsible for perceiving the world, and creating data for the network layer to collect and deliver (Zivkovic et al., 2020, pp. 8-9). Devices that contribute to this are, for example, Global Positioning Systems(GPS), cameras and sensors. The raspberry-pi models we worked with utilizes both a camera and a range detection sensor that gets processed by the image recognition algorithm running on the machine. These vehicles play the role of client in our system, as they connect to the server.

The requirements for our client was that it needed to be able to take in commands from the server and respond correctly to those commands. In addition, they had to be able to act by themselves if they were not connected to the server. For that to happen the vehicles had to send their size, position and velocity to the server when they connected initially. The server also needs to send their velocity and position to the server at a frequent rate so that the server can keep track of that information.

4.2.2 Implementation of Server

The network-layer of an IoT-system is responsible for transferring data collected by the things-layer (Zivkovic et al., 2020, pp. 8-9). In our project this is done by a centralized server that is connected to all the clients.

The task required the client, which in our case is the cars, to receive the commands from the server. The group prior had made it such as the car made decisions based on its AI. The AI used picture recognition which could recognize speed limit signs and stop signs. The cars were also programmed to follow a path. Furthermore we needed a server which could send commands to the clients at a frequent rate, based on the information given by the clients. The server's commands need to overwrite the AI that is already on the vehicles.

Here are some requirements needed for the server:

- Needs to be able to send messages to all clients simultaneously, or a specific group of clients.
- Handle increasing traffic
- Little to no delay
- Two way communication between server and client

We initially coded a server by using API. The server was able to receive information from the clients, but for the cars to receive information from the server, they had to host their own servers. The car's main priority should be calculating decisions based on their AI, not hosting a server. It would also be harder to add a server to the code from the previous group. We therefore thought of some other solutions.

Websockets provides a two way communication over a single tcp-connection. This means that both the server and client can send and receive messages from each other. They also allow for a better efficiency than REST API because they do not require the request/response overhead for each message that is sent and received. This solved the issue we previously had with the solo API-solution.

ASP.NET Core Signal R is an open source library that simplifies adding real-time web functionality to apps (Microsoft, 2022). It supports websockets as a real time communication which is perfect for a two way communication. Signal R is often used in games, social networks and gps-apps where information to the clients are needed instantly. It also scales with increasing traffic. Signal R uses hubs where servers and clients communicate with each other. Hubs allows servers and clients to call methods on eachother. Since the server needs to send the cars commands this is perfect for our Iot-system.

The server needed some specific functionalities in correlation with the problem statement stated earlier. We did some research on what caused traffic jams and chose two problems which our server would focus on solving:

- Delayed reaction times when people stop and accelerate
- Increased traffic flow in intersections

The first point happens when a car changes velocity. Therefore the server has to check if any car's have changed their velocity. If yes, the server needs to send a command to all the car's behind a certain distance of the car that changed their velocity. The distance will be calculated by the server based on the change of velocity. The command sent out to the car's behind will be to change velocity based on distance to the car that changed their velocity and the amount of changed velocity.

To increase traffic flow in intersections we can have the server work as some traffic lights. The server can check if cars are closing in on the intersection's position and give one lane the green light and the other the red light. The lights will depend on which car's come first. The difference between a server doing this and a traffic light system is that the server can tell the vehicles to slow down before the intersection, making the traffic flow smoother.

4.3 Implementation of simulation

To show that the solution was reaching the requirements we needed to create a demonstration. A demonstration was also an important part of testing the functionalities of the IoT-system. There were two options which were viable in this case. The first one was to create a virtual simulation using unity or another graphic-program. The other option was to build a physical demonstration with two or more cars. We chose to implement a physical demonstration with two cars. This was because we already had one car finished by the previous group. In addition it was more beneficial for Accenture with a physical demo since they can show it at exhibits.

- That the server can turn off and the cars would still drive on their own.
- Cars communicating through the server and acing based on the server's decisions
- Scalability to the real world

First we built the server with only one single lane road in mind. With this solution the potential of showing off the functionalities of the IoT-system was low. We could potentially have shown that if a car in front slows down, the car behind also slows down. This would show that the system could prevent some traffic, but only in a specific scenario. However we wanted to show that the IoT system could work in more than just one single-lane road. We therefore chose to try to emulate an intersection in the physical demo.

For a physical demo to work we needed to build another car. Luckily the previous group had documented their work and we could follow their process from their final report. We also had the parts provided to us. However we were unable to get the full functionality of the second car since the TPU accelerator used by the previous group was unavailable. The TPU accelerator provided the car with the processing power needed for the AI. This meant that the car was built without the camera. But this had a minor effect on our demo since the other car was fully functional without the system. We solved this problem by letting the server do the decision making for the car with less functionality.

We also needed to emulate a road system which contained the intersection we were going to demonstrate in the demo. Since the AI had not trained to recognize intersections yet we had to code a road system into our server that contained roads, lanes and intersections. We then had to build a physical intersection for our cars with tape that correlates with the emulated intersection.

4.3.1 Calibration of the cars

When the server, vehicles and client were implemented, and the second vehicle built, we did some testing to figure out how the car's behaved when given directions by the server. In this test the vehicles were given a specific velocity and driving distance by the server. When the vehicles arrived at their destination the server would tell them to stop. The car's drove in a straight line.

The vehicles were able to send information, and respond correctly to the servers commands. We also observed that the vehicles drove a different length for each

Power (?)	Length (cm)	Time (s)	Velocity (cm/s)
40	467	8.98	52.00
50	425	7.28	58.38
60	400	6.06	66.01
70	357	5.18	68.92
80	325	4.49	72.32
90	314	4.03	77.92
100	286	3.62	79.01

Table 4.1: Test text

velocity given even though the length was the same. This is because the velocity given to the vehicles is the amount of power going into the car's motors, not the actual velocity of the car's. We wanted the demo to be accurate so our group did some further testing where we wrote down the results.

The data Power was the velocity given by the server. Velocity was the actual velocity in our testing, which is length divided by time. As you can see the velocity was not the same as the power. We then made a graph to visualize the two values. The y-axis was the velocity while the x-axis is the power.

Figure 4.1: Graph of velocity as a function of power

We observed that the correlation between power and velocity seemed linear. This means we could make a specific formula that describes the correlation between the two values. We used linear regression to figure out this formula:

Figure 4.2: Graph of velocity as a function of power with linear regression

The formula we ended up with was as follows: $P = 0.4516v + 36.189$, where P is power and v is velocity, with a mean square error of $R^2 = 0.9653$. When we coded the formula into the vehicles we did another set of testing. We observed that the vehicles drove more or less the same distance for each power given. If we wanted an even more accurate formula we could have tuned the formula with the test results from our new test. Although the results were not hundred percent accurate, we concluded it was accurate enough for our demonstration.

4.4 System security

Security and privacy are important topics for any IoT-system. Because these systems gather and work with huge amounts of data, they are naturally prone to being attacked. And as the systems grow and become more interconnected, with many devices around the world, the imposed risk of such an attack increases drastically [kilde på dette]. Our project is on a small enough scale that we don't expect security to be an extremely important feature for it to work. However, because we want the system to be scalable in the future we have implemented some ideas that we think improves the data security in our system.

4.4.1 Privacy

Throughout our work we discussed how the solution would look in the real world. If all drivers on a road were to connect their cars to the same server it would be easy for anyone with access to that server to track any one driver. Therefore we proposed a solution based on the concept of anonymity (kilde på dette her, men vi må være på skolen for å ha tilgang). This is done by giving every vehicle that connects to the server an anonymous randomly generated identification-number that only lasts as long as the connection to the server is upheld. When the connection closes this number is deleted, and the same vehicle will get a new number the next time it connects to the server. This way we reduce the ability for an attacker to deduct any meaningful information tied to a single driver.

Product documentation

5.1 Client and server

5.2 Demo

To test the solution we have worked on, we made a physical demonstration with two cars that meet at an intersection. We want to test our hypothesis that a combination of a centralized communication system and artificial intelligence can improve traffic flow. To test this, our demonstration shows

5.3 User manual

We have written a manual for people who want to recreate our demonstration. The demonstration could for example be shown off at exhibitions. The manual could also be used for people who want to further test and develop our IoT-system.

First check the IP-address of the internet you are connected to, and the usable ports. Make sure that your computer hosting the server and the vehicles are connected to the same network.

```
$ipconfig getifaddr en0
192.168.56.208
```

Then open the Server solution in your code editor, we have used visual studio. Under the folder “properties” there is a file called launchSettings.json. In that file write in the ip address and the port in the applicationUrl-section:

```
"profiles": {
	"SignalRServer": {
		"commandName": "Project",
		"dotnetRunMessages": true,
		"launchBrowser": false,
		"applicationUrl": "https://192.168.56.208:7058;http
		    ↪ ://192.168.56.208:5048",
		"environmentVariables": {
			"ASPNETCORE_ENVIRONMENT": "Development"
		}
	},
```

After that open the Client solution. Here we used Pycharm as the text editor. Open the config.json document and write in the same ip-address and port:

```
"client": {
	"host": "192.168.56.208",
	"port": 5048,
	"delay": 0.1
},
```

If the vehicles haven't connected to that network before, they need to log on that network. To log on to a new network you will need to connect the raspBerry pi's to a screen. That could be done via the micro usb-port at the raspberry pi. When the raspberry pi has booted up, click on the internet-icon and connect to the same network as the server. If you have connected to the internet it will be saved and the vehicle should connect to that internet automatically when booting up.

If you want to change the velocities of the vehicles it can be done here in the server:

```
public class VehiclesHubDatabase : IVehiclesHubDatabase
{
	private readonly Intersection _intersection;

	private readonly HashSet<Vehicle> _vehicles = new();
	private readonly HashSet<Lane> _lanes = new();
	public int Count => _vehicles.Count;

	private readonly Dictionary<string, Vehicle>
	    ↪ _connectionIds = new();
	private readonly Dictionary<Vehicle, string>
	    ↪ _vehiclesConnectionId = new();
	public Dictionary<Vehicle, Thread?> VehicleThreads { get;
	    ↪ }= new();
	public Thread? GlobalThread { get; set; }
	public Stopwatch Clock { get; } = new();

	public double SpeedLimit => 80;
```

The file is located at VehicleHubDatabase under the Database folder. The variable you want to change is the SpeedLimit.

Right under you can change the length of the roads:

```
public VehiclesHubDatabase()
{
        _intersection = new Intersection().
                AddRoad(new Road {Length = 300}.
                        AddLane(null, true).
                        AddLane()).
                AddRoad(new Road {Length = 300}.
                        AddLane(null, true).
                        AddLane());

        _intersection.ConnectedLanes().
                ForEach(lane => _lanes.Add(lane));
}
```

We have written more about the specifics of the system in our product documentation. Then place the vehicles down at the start of the track, turn on the server and connect to the power banks. The vehicles should automatically connect to the server and start driving after 20-30 seconds.

Discussion

6.1 Evaluation of process

In our process documentation we have described the way the project progressed from beginning to end. In this section we will discuss positive and negative takeaways, with the goal of aiding further work on the project. This will include our approach to work methodology, communication and team building.

6.1.1 Work methodology

As previously mentioned, we chose to apply an agile work methodology based on the fact that our project seemed to be prone to many changes. We included rituals like daily stand-ups, sprint planning meetings and retrospective meetings after sprints. Doing this helped our team understand what difficulties the other members were facing, and gave us time to discuss our obstacles daily.

There were also some elements of agile development that we did not incorporate, either because of lack of experience or lack of time. As explained in the work of Skyttermoen, T. and Vaagasar, A.L.; agile development teams regularly have meetings with the product owner, who give feedback on the strengths and weaknesses of the deliveries (Skyttermoen & Vaagaasar, 2017, pp.120). Our group only had a few meetings with the product owner during the project period, to update him on how far along we were. We did, however, have biweekly meetings with our external supervisors at Accenture who provided us feedback on the work we had done. Because we had the freedom to choose the type of solution we wanted, as long as it met the requirements that were set to us, we didn't need the product owner to be in our biweekly meetings.

6.2 Evaluation of results

6.3 Further work

How to implement self-driving in society in the best way is a question that will take a long time to answer. Through our work with this proof-of-concept we believe we have made an addition to this discussion. Due to time constraints, we have chosen not to implement some features that would make the product work in a more complex environment. These features could be explored if this project were to be further developed.

6.3.1 Scalability

When the knowledge and research has come further regarding AI, there may be a possibility for such an IoT system to be scaled for the real world. There is a need for less traffic in the cities and our results from this project shows that an IoT-system where cars can communicate through a server will increase traffic flow. Scalability is therefore important which we had in mind while coding the server and the client.

Our demo only contains two cars but our server is built in a way where multiple vehicles can connect to it. There are also possibilities to connect other devices to the server, for example traffic lights. However there are no specific functionalities regarding traffic lights on the server as of now. Scaling the IoT system for functionalities with traffic lights is a task for future development. Adding roads and making a more complex road system would also be a task for future development.

6.3.2 Extendability to the real world applications

Edge computing

In the future when self-driving vehicles become more prominent, and the 5G network becomes more available there could be a possibility for IoT-systems handling traffic management. Our group therefore did some research regarding how the system will extend to the real world's applications.

The IoT systems usually follow the fog or edge computing architecture with distributed or even decentralized concepts (Zivkovic et al., 2020, pp 149). This is to prevent overloading on servers handling a lot of data.

One solution to distribution of data is that each road has one server responsible for their respective road. If a road is long it will be split into geographical areas where one server has responsibility for their geographical area. Intersections will have a server handling information from both road's respective servers, since information from both servers are needed to make decisions in intersections.

In the traffic there are a lot of unforeseen situations that can happen. Traditional coding will not be able to cover every outcome in a traffic situation, therefore there will be a need for AI on the servers in addition to the cars.

6.3.3 Risk management

Conclusion

Bibliography

Accenture. (2021). *Earnings report - 2021*. Retrieved March 21, 2022, from https://newsroom.accenture.com/news/accenture-reports-very-strong-fourth-quarter-and-full-year-fiscal-2021-results.htm

Accenture. (2022). *About accenture*. Retrieved March 18, 2022, from https://www.accenture.com/no-en/about/company-index

Brown, S. (2021, April 18). *Machine learning, explained*. Retrieved March 18, 2022, from https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Chinese APEC Delegation. (2014). *White paper of internet of vehicles (iov)*. Retrieved March 18, 2022, from http://mddb.apec.org/Documents/2014/TEL/TEL50-PLEN/14_tel50_plen_020.pdf

Datatilsynet. (2019, July 17). *Hva er en personopplysning?* Retrieved April 18, 2022, from https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/

Delua, J. (2021, March 12). *Supervised vs. unsupervised learning*. Retrieved March 18, 2022, from https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

IBM Cloud Education. (2020, June 3). *Artificial intelligence (ai)*. Retrieved March 18, 2022, from http://ibm.com/cloud/learn/what-is-artificial-intelligence

Kopestinky, A. (2022, March 5). *25 astonishing self-driving car statistics for 2022*. Retrieved March 21, 2022, from https://policyadvice.net/insurance/insights/self-driving-car-statistics/

Ntb. (2002, October 6). *Menneskelig svikt bak de fleste bilulykker*. Retrieved March 18, 2022, from https://www.aftenbladet.no/lokalt/i/Agpw3/menneskelig-svikt-bak-de-fleste-bilulykker

Raspberry Pi. (n.d.). *What is a raspberry pi?* Retrieved April 13, 2022, from https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

Sinha, S. (2021, September 22). *State of iot 2021: Number of connected iot devices growing 9% to 12.3 billion globally, cellular iot now surpassing 2 billion*. Retrieved March 18, 2022, from https://iot-analytics.com/number-connected-iot-devices/

Synopsys. (n.d.). *What is an autonomous car?* Retrieved March 18, 2022, from https://iot-analytics.com/number-connected-iot-devices/

Zhu, Q., Wang, R., Chen, Q., Liu, Y., & Qin, W. (2010). 2010 ieee/ifip international conference on embedded and ubiquitous computing. *IOT Gateway: BridgingWireless Sensor Networks into Internet of Things*.

Zivkovic, C., Guan, Y., & Grimm, C. (2020). *Iot platforms, use cases, privacy, and business models: With hands-on examples based on the vicinity platform* (1st ed.). Springer, Cham.