UNIVERSITY OF OULU

Voice-controlled in-vehicle infotainment system

University of Oulu
Faculty of Information Technology and
Electrical Engineering / Information
Processing Science
Master's Thesis
Jere Mourujärvi
7.4.2020

Abstract

Speech is a form of a human to human communication that can convey information in a context-rich way that is natural to humans. The naturalness enables us to speak while doing other things, such as driving a vehicle. With the advancement of computing technologies, more and more personal services are introduced for the in-vehicle environment. A limiting factor for these advancements is the impact they cause towards driver distraction with the increased cognitive stress load. This has led to developing in-vehicle devices and applications with a heightened focus on lessening distraction.

Amazon Alexa is a natural language processing system that enables its users to receive information and operate smart devices with their voices. This Master's thesis aims to demonstrate how Alexa could be utilized when operating the in-vehicle infotainment (IVI) systems. This research was conducted by utilizing the design science research methodology. The feasibility of voice-based interaction was assessed by implementing the system as a demonstrable use-case in collaboration with the APPSTACLE project. Prior research was gathered by conducting a literature review on voice-based interaction and its integration to the vehicular domain. The system was designed by applying existing theories together with the requirements of the application domain.

The designed system utilized the Amazon Alexa ecosystem and AWS services to provide the vehicular environment with new functionalities. Access to cloud-based speech processing and decision-making makes it possible to design an extendable speech interface where the driver can carry out secondary tasks by using their voice, such as requesting navigation information. The evaluation was done by comparing the system's performance against the derived requirements.

With the results of the evaluation process, the feasibility of the system could be assessed against the objectives of the study: The resulting artefact enables the user to operate the in-vehicle infotainment system while focusing on a separate task. The research proved that speech interfaces with modern technology can improve the handling of secondary tasks while driving, and the resulting system was operable without introducing additional distractions to the driver. The resulting artefact can be integrated into similar systems and used as a base tool for future research on voice-controlled interfaces.

Keywords
Automatic Speech Recognition (ASR), Amazon Alexa, Design Science Research, Route navigation, Amazon Web Services (AWS)

Supervisor
PhD, Teemu Karvonen

Foreword

Firstly, I would like to thank my former employer Professor Pasi Kuvaja for accepting me to the M3S research unit as a research assistant, giving me the opportunity to write my thesis on this topic and being a part of the project APPSTACLE. I am grateful for all the help and support I received as I was working under your supervision. My appreciation goes to Teemu Karvonen for supervising my thesis and supporting me throughout the process. I would also like to thank the M3S research unit and its staff for providing me with a supportive and motivating environment to work in.

I would like to thank Arun Sojan Kudakacheril for being a co-supervisor for this thesis. I also want to thank him as a colleague, for helping me solve various challenges and as a friend, for providing support and the kind words of encouragement for me during difficult times. Thank you for being there for me.

Furthermore, I would like to give my thanks to Ahmad Bani Jamali who acted as my opponent for this thesis. His support has been tremendous, and I value the amount of time that was spent to provide me with guidance and suggestions for improvement in this thesis.

I am grateful for my family for supporting me throughout my time here at the university. I would also like to thank my friends, colleagues and the Saunaclub for the great company throughout the years.

Jere Mourujärvi

Oulu, April 7, 2020

Abbreviations

AI	Artificial Intelligence
API	Application programming interface
APPSTACLE	open standard APplication Platform for carS and TrAnsportation vehiCLEs
ASR	Automatic Speech Recognition
AVS	Alexa Voice Service
AWS	Amazon Web Services
CALO	Cognitive Assistant that Learns and Organizes
CDI	Compact Driver Interface
DARPA	Defense Advanced Research Projects Agency
DSR	Design Science Research
GPS	Global Positioning System
IDE	Integrated development environment
IPA	Intelligent Personal Assistant
HCI	Human-Computer interaction
HMI	Human Machine interaction
NHTSA	National Highway Traffic Safety Association
NLP	Natural Language Processing
OEM	Original Equipment Manufacturer
POI	Point of Interest
SDK	Software development kit
VODIS	Voice Operated Driver’s Information Systems

Contents

1. Introduction

In recent years, voice interaction and intelligent personal assistants have gained more popularity in both commercial and industrial areas, as Intelligent Personal Assistants (IPAs) are utilized to provide speech-based support in business processes (Hüsson & Holland, 2019). With the constant flow of new technology being introduced and theoretical concepts becoming reality before our very eyes, giving focus on issues that accompany these newly introduced concepts is welcomed. Since the beginning of automotive development, the in-vehicle platform has constantly been a target for innovation. The first electronic systems for the cars were introduced in the '60s. Nowadays most of this functional innovation comes from developing software into the automotive systems (Mössinger, 2010). Developers are introducing new ways for the driver to interact with the in-vehicle systems. However, the process of driving a vehicle limits the level of interaction a driver can have with novel interfaces. Developing new interfaces and functionalities to use within the car leaves innovators struggling with the safety-critical aspects and requirements of the automotive environment. (Chen et al., 2008)

Human-computer interaction (HCI) can be described as follows: *Human*, meaning the user, users or a sequence of users. *Computer*, meaning any technology from general desktops to large or embedded systems, with non-computerized parts. And *interaction,* meaning the direct or indirect communication that happens between the human and the computer. A user interacts with the computer to accomplish something. The interfaces of communication are for example. keyboards, controllers, touch screens and other devices designed for user input (Dix, 2004). Human-machine interaction (HMI), similar to HCI, is the concept of interaction between human and a machine. This idea of interaction between the human and a machine has derived from how reactive the computer-based artifacts have become throughout the years (Suchman, 1990). In the context of this study, HMI is addressed as the interaction between humans and a vehicle with heightened technological capabilities.

Visual and physical interaction capabilities of the driver are already occupied by the task of keeping the vehicle safely on the road. When designing systems for the vehicular environment, Smith (2000) emphasizes that keeping your eyes on the road and hands on the wheel are the most obvious requirements for keeping physical and visual distractions to a minimum when operating a vehicle (Smith, 2000). Driver distraction has a major contributing factor to traffic accidents in general (Patel, Ball & Jones, 2008). This has led to having minimized driver distraction as one of the key goals in HMI. These HMI products generally focus on controlling multiple in-car functionalities, such as navigation, telephony, vehicle dynamics, etc. and they fall under one of the following categories: haptic based and voice-based HMI (Weng et al, 2016). Without figuring out a new approach to the issue, the innovators are left with just a handful of communication interfaces to safely introduce new interaction methods.

Speech is a form of communication that is used to verbally express us, and it can be carried out while performing other tasks. Voice-based interaction has been proven useful in the domain of Intelligent Personal Assistants (IPAs), where microphone enabled devices provide information and operate with other smart devices in the household. Voice-based interaction with an IPA is an activity that is not dependent on the user's physical or visual attention. Customers with disabilities have also benefitted from using IPAs, as they provide extended control over daily tasks that are otherwise too difficult to achieve (Gao, Pan, Wang & Chen, 2018). Due to this, voice interaction can be said to be the most suitable method of communication between the driver and the vehicle. (Weng et al., 2016) Introducing a speech interface for an in-vehicle context has not been possible in the past, as the technical requirements for developing such a tool were too difficult to fulfill with the available technology. Factors such as embeddedness, voice recognition and robustness cause difficulties that have been challenging to solve in the past. (Hansen, Kim & Angkititrakul, 2008)

What kind of improvements can the utilization of speech-controlled IPAs provide for the in-vehicle environment? The sheer existence of cloud-computing could help to solve the prevalent technical challenges that exist in the vehicular environment. The issue of handling secondary tasks in the vehicle has received notable recognition from car manufacturers, and some manufacturers have already implemented speech recognition features into their vehicles. In these vehicles, speech recognition facilitates features such as dictating emails and text messages, playing music, provide navigation capabilities and start the car engine (Khan, Akmal, Ali & Naeem, 2017). Weng et al. (2016) describe the in-vehicle dialog system development as a process that "requires multidisciplinary expertise in automatic speech recognition, spoken language understanding, dialog management, natural language generation, and application management, as well as field system and safety testing" (Weng et al., 2016, p. 49).

The objective of this Master's thesis is to find out if modern voice-based interaction solutions can be utilized to create a feasible voice-controlled infotainment system for the vehicular environment. In this context, the infotainment system means a system that is responsible for conveying information to the users of the system. In the context of this study, a navigation-based system is built to provide the user with navigation capabilities. In the field of information technology, it is common for new technologies to come by and solve issues unsolvable by prior technologies. This study was an attempt to fill the research gap of utilizing modern technologies to solve the long-lasted issue of drivers being distracted by non-essential services in the automotive environment. Modern voice-based interaction solution refers to the currently available Intelligent Personal Assistants IPAs, namely the Amazon Alexa (Amazon, 2010b). Alexa is utilized in creating a speech interface to operate the infotainment system.

The reasoning behind choosing Amazon Alexa over the other IPA candidates was based on the early introduction of the Alexa Skills Kit, which made it easier for third-party developers to create new Alexa skills. While Google's own IPA, Google Assistant, displayed similar customization capabilities as Alexa with its own platform called "Actions on Google", there were several key points between that contributed to the decision process. In development, Alexa skills can be customized to interact with third party services. For example, retrieving weather information by requesting it by utilizing Amazon Alexa and the AWS Lambda serverless cloud computing platform (Kusuma et al., 2019). A large number of existing applications, design guidelines and implementations made with Alexa was considered a significant difference during the decision process (Kinsella, 2019). Learning to design and develop applications was determined to be easier with a technology backed by a larger and more mature community.

The implemented system is evaluated by measuring it against the issues and solutions identified in prior research, as well as the requirements derived from the project use case. Introducing the extensive usability improvements that the voice-based interaction technology provides for the safety-critical vehicular environment is the main objective for conducting this research. The resulting work could bring us closer to a situation where common tasks in the vehicle can be solely operated by voice. Although the road to self-driving vehicles being a commodity is still long ahead of us, paving the road starts from the process of easing the drivers to use the navigational- and other infotainment systems with varying non-invasive interfaces to support the traditional driving process.

This thesis was conducted in collaboration with the APPSTACLE project in which the author of this thesis was a member of (ITEA3, 2016a). Project APPSTACLE's goal was to provide an open and secure car-to-cloud and cloud-to-car platform. This platform provides an open ecosystem where software development and deployment can be done while adhering to the quality requirements of the automotive industry. The wide availability of vehicles capable of utilizing this platform and the existence of the ecosystem opens the market for improving legacy and creating new features for the automotive context. The project started in October 2016 and was concluded in December 2019. Participants included companies and universities from multiple countries: Finland, Germany, the Netherlands, and Turkey. (ITEA3, 2016b) As a result of the project, the open-source project and platform called Eclipse Kuksa were released in November 2019. This platform interconnects vehicles to the cloud and provides developers with a set of software and its IDE. Eclipse Kuksa is now the project that carries out the vision of the project APPSTACLE. (Eclipse, 2019)

The results of this research contribute both for the knowledge base, the application domain as well as the automotive industry in the following ways:

- Provide insight on speech recognition in vehicles by utilizing the Design Science Research (DSR) methodology to design, develop and evaluate a voice-controlled navigation system.
- Demonstrate the capabilities of currently available speech recognition technologies.
- Provide a use-case for the APPSTACLE project that demonstrates the use of third-party services within the car-to-cloud platform.
- Present evaluation results that show how Amazon-based services can be used and improved to provide cloud-based ASR.
- Use these evaluation results to describe the advantages and disadvantages of this artifact and identify the room for improvement.

This thesis follows the Master's Thesis template provided by Halonen (2017) from the University of Oulu. The template is used to guide students to write a Master's Thesis that follows the regulations of the University of Oulu. The structure of the thesis is as follows. The paper starts by defining the research problem and the questions. Next, the design science research methodology is introduced and described. Then a literature review is conducted to help understand and summarize the existing knowledge related to speech interfaces, the in-vehicle environment and the integration of these two areas. From the findings of the literature review and the domain-specific needs of the use-case, a set of requirements is generated. The system is designed and developed based on these requirements. The implementation of the system is done in small cycles to ensure alignment with the requirements. The system is evaluated by analyzing how the implemented system meets the requirements. Finally, the results are discussed along with the limitations and identified opportunities for future research.

2. Research problem and Methodology

The objective of this thesis is to design develop and assess a voice-controlled interface to be used in an in-vehicle environment. To achieve this goal, the following research questions were defined:

> RQ1. How a voice-controlled interface can be integrated into the in-vehicle context?
> RQ2. What are the advantages and disadvantages of a voice-controlled interface in the in-vehicle system?

By answering these research questions, the thesis can provide insight on how a speech controlled in-vehicle navigation system should be designed and developed, and if there still exist problems that the current commercial solutions cannot answer to. The first research question is directed towards defining the design- and development process. By answering it, the reader should have a clear idea of how the system is being built and what are the technologies that are utilized in the process. The second research question is directed towards evaluating the designed artefact to see what kind of effects the target solution has in the application domain. Whether these effects are positive or negative and what areas do they cover, the results reveal how well the designed solution could be utilized in the application environment.

The knowledge base was examined through the process of a literature review. The purpose of the review was to gather existing literature that covers the research topic from the following three aspects: Speech interfaces and automatic speech recognition, the in-vehicle environment as the application domain and the speech recognition application process to the in-vehicle environment. For non-scientific literature, websites that were created by the companies and organizations behind the tools and services used in the study were examined.

The results of the literature review were used to construct literature-based requirements that were utilized during the design, development, and evaluation of the design artefact. The literature review provided both the literature-based requirements that acted as the basis for the design- and development work, and the evaluation criteria for assessing the rigorousness of the resulting artefact against the combined evaluation criteria. Literature-based requirements included both technical- and application domain-specific requirements. These requirements were merged to finally come up with a set of requirements that provide a strong basis for the design part of the research process.

The research question 1 was answered by examining the currently existing methods and applications that are implemented in the vehicular environment. As the basis of the voice-controlled interface is previously known (Alexa), the task of domain integration remains to be answered by investigating the currently existing methods and applications.
The research question 2 was answered by evaluating the designed system against the constructed requirements. The evaluation process provides us verification on each of the advantages and disadvantages by evaluating the features of the tool and the challenges of the environment case by case.

2.1 Research method

As this thesis aims to build and evaluate a new system based on empirical research, the study will follow the design science research (DSR) methodology as the approach method for this thesis. Hevner and Chatterjee (2010) provide a formal definition for Design science research:

> "Design science research is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artefacts, thereby contributing new knowledge to the body of scientific evidence. The designed artefacts are both useful and fundamental in understanding that problem. (Hevner & Chatterjee, 2010, p. 5)"

DSR enables organizations to address tasks that improve effectiveness with the use of information technology. There needs to be a balance between the focus on technological artifacts and maintaining a theory base. Neglecting the theory can result in the development of a well-designed artifact without actual use in the organizational setting (Hevner, March, Park & Ram, 2004). Hevner (2007) emphasizes the importance of providing a clear and consistent theoretical base for the design and execution of high-quality design science research projects. It is essential to understand and communicate the design science research process to gain support from other information system professionals and to make design science research seen as a credible source of information among other design science-oriented research communities. (Hevner, 2007)

As stated by Hevner (2004), DSR is conducted in collaboration between the business needs of the application environment and the applicable knowledge from the knowledge base. Primarily, rigorousness in DSR is achieved by evaluating the quality and efficiency of the artifact by utilizing the knowledge base for computational and mathematical methods (Hevner, 2004). In DSR, the task of conducting a literature review is done to assess a selection of appropriate theories and methods that can be utilized in the construction and evaluation of the designed artefact.

Answering a research problem with the use of DSR means that there are multiple different DSR frameworks to choose from. Deciding on a suitable framework for a study is a process of identifying the general needs and projected outcomes of your research, whether it is to focus on increasing industrial relevance, or to provide valuable meta-artifacts for progressing the knowledge base, or to provide something in between. In the case of this study, the Three Cycle Framework by Hevner (2007) was chosen as the one to be followed for fulfilling this research. This framework borrows an existing framework from Hevner (2004), with a new overlaid focus on the implementation of three inherent cycles, as seen in Figure 1.

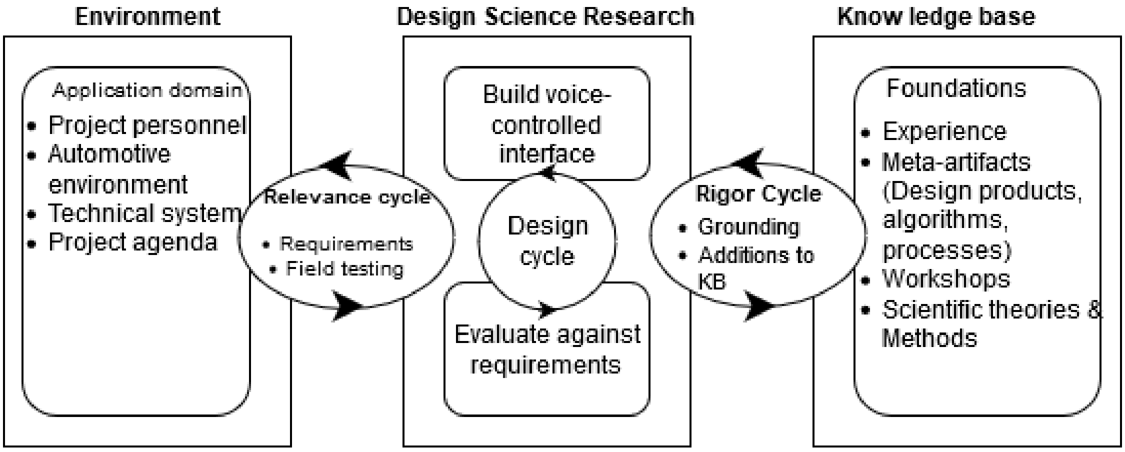


Figure 1. Design Science Research Cycles based on this research context (Hevner, 2007).

The essay from Iivari (2007) portrays the key properties of the design science research paradigm. Hevner supports the analysis from Iivari and portrays the theses in Iivari's essay into three design research cycles. Relying on these research frameworks paves the road towards a well-structured DSR study that maintains a balance between relevancy and rigorousness of the study (Iivari, 2007).

The left side of Figure 1 represents the contextual environment; The application domain in which the designed artifact would prove its usefulness by either solving an existing problem or simply filling a spot open for opportunities. In the context of this study, the APPSTACLE project and the selected target environment along with its technical stack will fill the position of the contextual environment. These will act as the sources for enforcing the relevancy of the study. As Hevner (2007) states, good design would often begin by identifying and representing the opportunities and problems in their application domain (Hevner, 2007). Identified opportunities provide a reason for improvement even without recognizing any problems (Iivari, 2007).

The right side of the figure contains the knowledge base, which is composed of foundations and methodologies. Hevner et al. (2004) describe that the knowledge base provides prior research and results in the forms of artefacts such as foundational theories, models, frameworks and methods. These artifacts can be results from workshops, design products and algorithms from similar DSR works and scientific theories in general. In contrast to behavioral science, rigorousness in DSR is often achieved by applying computational and mathematical methods to evaluate the quality of the research work. (Hevner et al., 2004). In the context of this study, the theoretical knowledgebase consists of studies and workshops that have resulted in meta-artifacts that reveal existing issues and problems within this topic area.

2.2 Design Cycles

Hevner (2007) defines the DSR cycles in the following manner:

Relevance cycle: The process of identifying potentiality inside the application domain. Upon initiation, it provides design science research an application context where one can gather both the requirements for the research and the evaluation criteria for the finished research work. A cycle is iterated if the designed artifact cannot maintain utilization in practice. In this study, the relevance cycle consists of gathering requirements from the application domain, which in this case is represented by the project and its purpose. It is important to ensure that the designed requirements and other artefacts of this study stay relevant to the target application domain in which the designed system is considered to be deployed.

Rigor cycle: The Rigor cycle provides past knowledge for the research project. To recognize the designed artefacts as research contributions for the knowledge base, researchers need to thoroughly research and reference the existing knowledge base. While basing design science research on supporting theories is seen as an important aspect of the research, finding supportive theories in for every single DSR project is deemed unrealistic (Iivari, 2007). The research contributions of the DSR project to the knowledge base are what sells it to the academic audience: Extensions of original theories and methods made during the research and the resulting meta-artifacts and experiences generated by the design and evaluation of the artifact in the application environment.

The application of the rigor cycle in this study can be seen in the following process: Assessing that the gathered knowledge correctly aligns with the relevancy of this study. This means acknowledging how the selected studies and workshops are relevant to the subject of voice-based interaction and the automotive environment. If there are issues discovered and solutions proposed, do they support the rigorousness of this research? With these existing research materials, this DSR study gains the means of justification and evaluation guidelines for the resulting artefact.

Design cycle: The main internal cycle of the DSR project. The cycle iterates around constructing, evaluating and refining the design of the target artifact. The cycle gains the requirements for the design from the relevance cycle while design- and evaluation methods are gained from the rigor cycle. Maintaining the balance between each of these iteration cycles is essential for constructing the artifact (Hevner, 2007).

The environment of this research is the automotive domain, that the underlying project APPSTACLE is focused on improving. The system is intended to be used in collaboration with existing devices that provide infotainment for the in-vehicle environment. The goal of this tool is to provide insight into using an alternative method of communication with these existing navigation- and infotainment systems available in the car. Without the voice-controlled interface available, operating the existing navigation systems is preferred to be done before starting the driving process. Operating the navigation system while driving impacts the safety of the driver and other vehicles in the vicinity of the car, as the focus of the driver is directed towards the navigation interface and one of the hands leaves the steering wheel to operate it.

As the development of the artefact was a core process of this research, the development of the artefact was done in cycles. Each cycle had a specific objective to accomplish and they were evaluated in a follow-up evaluation cycle. The cycles of this research work are presented as follows:

Cycle 1. Gathering project requirements: With the literature requirements summarized, a separate relevancy cycle was initiated for gathering application domain-based requirements from project personnel. These requirements were evaluated with the support of the project personnel with relevant expertise in the subject.

Cycle 2. Developing the final requirements: The final requirements were developed by merging the rigorous requirements gathered from the literature review and the relevant requirements generated with the project personnel. Cycle completion was done by successfully carrying out the following evaluation cycle. Evaluation of this cycle was done by analyzing the requirements with the other members of the project while considering the limitations on time and resources. Depending on the nature of the requirement, the iteration was conducted by refactoring the requirement or removing it completely.

Cycle 3. Selecting the system components: The objective of this cycle was to choose the components for carrying out the required tasks. The chosen components must be suitable to work with the designed system structure and they must follow the set requirements. The completion of this cycle was confirmed when all the components were selected, and they were assessed to function together to fulfill the requirements.

Cycle 4. Designing the system structure. The objective of this cycle was to design the structure for the components of the system. The requirements gained from the prior cycle were utilized to design the system. This cycle was estimated completed when the designed structure would align with the requirements that define the system or the parts that were under the design process.

Cycle 5. Developing the system: The objective of this cycle was to develop the system to fulfill the features that were defined in the requirements. Each part of the system was merged as defined in the designed structure, which resulted in the creation of the designed artefact. The completion of this cycle was confirmed when the following evaluation cycle was defined completed by the author.

After the cycles were completed, the system in its whole was evaluated by assessing the functional- and performance capabilities against the prior requirements. The assessment procedure measured the functional capabilities of the system by comparing them against the traditional voice-operated system, Alexa and its skills. Successfully evaluating the artefact with the laboratory setting that simulates the vehicular domain as its environment allowed the research questions to be answered thus solving the research problem. Having solved the research problem provides new knowledge towards the knowledge base in the form of designing a voice-controlled interface for the in-vehicle environment by using the Amazon Alexa.

3. Prior research

This chapter introduces the prior literature that was gathered in the literature review process. The literature is related to the subjects of voice-based human-machine interfaces and the in-vehicle system. To provide the reader with an understanding of the core subject, it is important to explain the concept by introducing the literature under the following topics areas: Speech interfaces from past to present, Description of the in-vehicle environment and Existing implementations of speech interfaces in an in-vehicle environment.

3.1 Speech interfaces

Speech recognition has been an attractive research topic since the 1950s. The basis for voice or speech identification technology was pioneered in the 1960s by Texas Instruments (Khan et al., 2017). Since that time the topic has been investigated in hopes of designing a speech interface that can provide a highly user-friendly human-machine interaction experience (Herbig, Gerl & Minker, 2010a). The aggressive research and development processes have brought speech recognition technology into the present, where the technology has been integrated into the mainstream (Khan et al., 2017).

In the study by Hunt (2002), the domain of automatic speech recognition was said to be mainly dominated by automatic dictation software from 1990 to mid-2000. In automatic dictation, users engage in a dialogue with computers to edit, format and correct textual input. The first versions of said dictation software required its users to learn how to pause between each word, making the dictation process time-consuming. Later, the dictation software evolved to work much more efficiently by requiring no pauses, using larger vocabularies and ensuring higher precision. PC Speech recognition by dictation was still not sparking much curiosity. (Hunt, 2002)

Hunt also investigated some factors that may have had an indirect relation to the use of dictation technology: Automated dictation technology heavily relied on new computers that could respond to the processing power- and memory consumption requirements. Another factor was that automatic dictation was directly competing with the keyboard and mouse, that many people had already learned to use when interacting with personal computers. It was true that the speaking to another human was largely different from speaking with a dictation system, as similarly to efficient typing, one would need to learn how to speak to a dictation system (Hunt, 2002). Zheng, Liu & Hansen (2017a) describe that commercial focused voice-based human-machine interfaces are typically established by utilizing two modules presented in Figure 2.

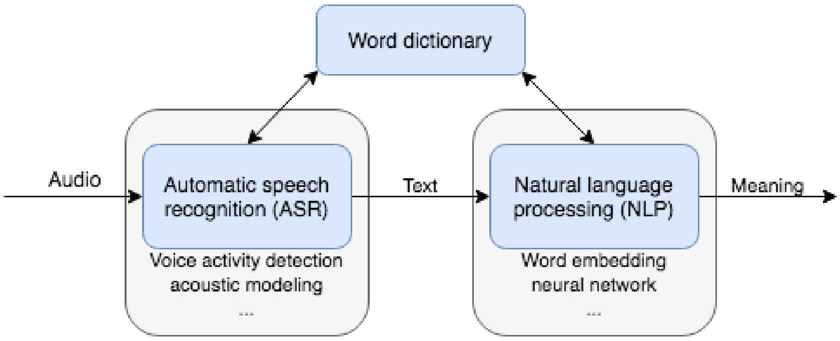


Figure 2. System overview of a voice-based navigation interface (Zheng et al., 2017a)

As seen in Figure 2, the speech recognition process starts with the module called Automatic Speech Recognition (ASR). Its task is to process the audio streams into sequences of words or sentences, which are then carried over to the second module called Natural Language Processing (NLP) (Zheng et al., 2017a). For human-vehicle navigation dialog systems, NLP has two major tasks: In navigation-based applications, the task of **intent detection** is to decide if the retrieved text is a sentence that is related to navigation or not. Next up is the task of **semantic parsing**, where the retrieved text is analyzed to figure out which words of the sentence are related to the context (of navigation) and which are not. An example of this task is provided by Zheng et al. (2017b) in the following Table 1, where the sentence is labeled against the semantic role chunking method presented by Hacioglu, Pradhan, Ward, Martin & Jurafsky (2004), where each word is tagged in the format of Inside Outside Beginning (IOB). In this sentence, the context of the Point of Interest (POI) is “a Japanese restaurant” and the context of the navigation search area is “near here”. (Zheng et al., 2017b; Hacioglu et al., 2004)

Table 1. Semantic parsing example (Zheng et al., 2017b)

Sentence	Context
is	O
there	O
a	B-point.of.interest
Japanese	I-point.of.interest
restaurant	I-point.of.interest
near	B-search.area
here	I-search.area

There have been multiple efforts devoted towards the field of general-purpose voice-enabled intelligent personal assistants (IPAs). The U.S. Defense Advance Research Projects Agency (DARPA) has funded several key programs related to IPAs. The DARPA Communicator -project is seen as a major early effort towards developing robust multi-modal speech-enabled dialog systems that carry advanced conversational capabilities to engage with humans in varying types of interactions. DARPA funded the CALO (Cognitive Agent that Learns and Organizes) project, which eventually generated several spinoff projects, such as the well-known SRI International’s Siri intelligent software assistant. (Weng et al. 2016)

Speech interfaces of the present are generally identified as tools with the capability of retrieving speech containing audio input and processing the speech externally by utilizing cloud computing technologies. Of these interfaces, the most well-known are the ones provided by the big tech companies, such as Apple Siri and Google Voice Actions (Zheng et al., 2017a). The interest in speech-based dialog systems took a large step forward in 2011 when Apple acquired the first voice-enabled Intelligent Personal Assistant (IPA) called Siri and launched it on the iPhone. With the availability of massive computational powers provided by cloud computing technologies, other voice-enabled IPAs and AIs began to get integrated into dialog systems. These newly developed IPAs allowed users to operate devices, access information and fulfill personal tasks in a more versatile manner. The list of notable IPAs includes titles like Apple Siri, Amazon Echo, Google assistant and, Microsoft Cortana IBM Watson and Baidu. (Weng et al., 2016)

The study by Gao et al., (2018) focused on understanding how the consumers were using the Amazon Echo -voice-activated wireless speaker. This task was carried out by analyzing consumer reviews of the product from the Echo's Amazon product page. The following statements were supported by the qualitative analysis: Hands-free is one of the top features that many reviewers enjoy. Remote voice interaction of the device allows its users to issue commands while they are busy with other tasks, such as cooking. There was a small subset of the customers that in their reviews mentioned other competitor's assistants and gave Echo a 1- or 2-star rating, and the majority of these reviews focused on how Echo's capability to answer questions was inferior to its competitors. Other reviewers commented that the Echo device does hear and understand your words, but it is not able to answer them. (Gao et al., 2018)

Unlike the other IPAs that are available in the market, Alexa is designed to handle speech in the form of providing skills instead of holding a conversation: Speaking to it invokes different skills that are designed to be activated upon hearing certain keywords, such as "Hey Alexa, how is the weather outside?" This phrase would activate a built-in skill that seeks the weather information from the location you had set beforehand. Amazon's website describes Alexa skills as voice-activated apps that add capabilities to your Alexa-enabled device (Amazon, n.d.). The feature that motivated this study of improving the vehicular context was the vast customization capabilities that the Alexa can provide: As it is possible to connect the voice-operated IPA to Amazon's cloud platform, the AWS widens the usability of Alexa skills even further by opening up new ways to interact with other services that are designed to work seamlessly with one another. (Amazon, 2011)

3.2 In-vehicle environment

Driving a car is a process that requires complete visual attention from the driver. Along with the basic functionalities, cars are equipped with varying functions that serve different purposes related to tasks that improve the driving experience, such as lights and window wipers. Drivers are expected to learn to use these functionalities without giving them much visual attention, which is already prioritized for driving the car. (Sandnes, Huang, Yo-Ping., Huang, Yeh-Min., 2008)

One of the most prevalent issues of the in-vehicle environment is the amount of additional noise that accompanies the driving process (Hansen, 2008; Hunt, 2002; Shozakai, Nakamura & Shikano, 1998; Hong, Rosca & Balan, 2004). Shozakai et al. (1998) list and categorize the noises of the in-car environment into four types, as shown in the following Table 2.

Table 2. Noises of the in-car environment (Shozakai et al., 1998)

	Known	Unknown
Stationary	Engine etc.	Road, wind, air-conditioner, etc.
Non-stationary	Car stereo speaker, navigation guide, traffic information guide	Bump, wiper, winker, conversation, noise from passing a car to the opposite direction

As a solution for the challenge of reducing visual demand in driving, alternative methods of interaction have been proposed. These methods vary from tactile, touch-based interfaces to interfaces controlled by voice. These methods were intended to be usable without expending the already reserved visual focus. One of the alternatives that do not utilize speech recognition was proposed by Sandnes et al. (2008): A tactile interface that represents a chording keyboard, which is used to issue commands that operate various functionalities in the car (Sandnes et al., 2008).

Kirson (1995) Describes a Compact Driver Interface (CDI) for use in the in-vehicle navigation and route guidance system. For inputting commands, the design uses a tactile input device, which enables the driver to operate the system without having to look at the controls, and the design is similarly designed for the driver not having to divert their attention from the task of driving to receive navigation instructions, as the system responds to the driver by voice. While the interface design communicates in a way opposite to this study, the principles of appreciating the driver distraction are still apparent. (Kirson, 1995)

Driver awareness has been one of the major safety concerns since the development of automobiles. In 2008, there were over six million vehicular accidents in the USA which killed 42,000 people and left 3 million people injured. Driver distraction is estimated to contribute to 20-30 percent of all crashes on motor vehicles. As stated by National Highway Traffic Safety Association (NHTSA) in the study by Hansen et al., (2008), driver distraction is classified into four distinct types: visual, auditory, biomechanical (physical) and cognitive distraction. These distraction types are not mutually exclusive, for example, operating a phone includes all four forms of actions. (Hansen et al., 2008)

3.3 Speech interface and the in-vehicle environment

To design a speech interface for the in-vehicle environment, various issues have been identified that are necessary to be solved in the design process. Without a solution for all these issues, no speech interface can be developed to operate within the standards of safety and robustness that are expected in the in-vehicle environment. Several modern speech interfaces built for the automotive domain have been released in the industrial market. They provide sub-tasks such as speech interface, navigation, and telematic services. Further detail on the technicalities is however not informed. (Zheng et al, 2017a)

Speech recognition began to appear in cars made for production in early 2000. During that time, speech recognition appeared in the form of dictation systems and was generally competing with the traditional keyboard and mouse users. One of the first projects that intended to realize a robust speech interface for the in-vehicle environment was the

VODIS (Voice Operated Driver's Information Systems). The project ultimately led to the design and development of a speech control system, targeted towards the in-vehicle environment, such as navigation and the car stereo (Shozakai et al, 1998). However, introducing speech recognition to the in-vehicle environment meant that there was no competition with the keyboard anymore. It was also noted that the in-vehicle environment was knowingly noisy, and drivers could not be expected to wear headsets mounted with microphones to interact with the system (Hunt, 2002).

Barros & Boucher (1996) from Liikkuva Systems discovered the following challenging problems via spending over two years researching and developing navigation technologies:

1. The success of ITS (Intelligent Transport System) applications depends solely on the accuracy of the map utilized by the program.
2. The most important issue in the ITS movement is the personal safety implications: Integration of voice recognition and speech synthesis is necessary for an In-vehicle navigational system. Accuracy of the road information is directly associated with the acceptance of the product.
3. The capability of altering a route calculation requirement while in transit.
4. The integration of technologies from different companies and service provides is a major problem. To develop useful and reliable navigational systems, cooperation between participating companies is a fundamental requirement, panning over government, private and educational entities.

They conclude that the success and overall effectiveness of ITS movement in the US are dependent on the two following statements: Cooperation between private technological firms, government and educational entities and Constant development of ITS technology integrating software programs and making the technology easily accessible for the common user. (Barros & Boucher, 1996)

Herbig et al (2010a) identify multiple in-car speech recognition aspects that are open for improvement: Users with no familiarity towards speech recognition systems should be able to both safely participate in road-traffic and simultaneously operate the systems. Speech recognition enabled infotainment- and navigation systems should not be personalized to a single user. However, some studies provide insight and reasoning on designing and developing personalized in-vehicle information systems: Moniri, Feld & Müller (2012) base their motive on providing a more comfortable and accessible experience to people from different user groups. This would impact safety, as the users would spend less time making the automotive experience comfortable for themselves, as the system does it for them (Moniri, Feld & Müller, 2012). However, Herbig et al (2010a) explain that if the speech interface is only used by a small set of users e.g. five recurring speakers, it is useful to train the system with the users' speech patterns to improve accuracy and efficiency (Herbig et al. 2010a).

As the system is being developed for an embedded environment, efficiency in both computation power and memory consumption are important design parameters (Herbig et al., 2010b). This statement is supported by Qian, Liu & Johnson (2009), who presented a fast decoding algorithm for a Mandarin speech recognition system. The proposed speech recognition strategy could improve the speech recognition speed by six-fold and use half of the memory with little accuracy degradation when compared to the baseline system (Qian et al., 2009). Similar decoding algorithms were proposed by Chung, H., Park, Lee, Chung, I. (2008) with similar intentions of improving the efficiency and memory management when recognizing POIs on an in-car navigation device (Chung et al., 2008). The research conducted by Qian et al (2009) focused on the issues on the embedded ASR systems that still lacked the proper solutions against the computational requirements. The research team proposed a fast decoding algorithm to help solve the efficiency issue with accessing a very large vocabulary with a system working with the limited speed- and memory size constraints of an embedded environment. (Qian et al., 2009)

Hataoka, Araki & Matsuda (2008) address additional problems that would require resolving before speech recognition can be utilized in any real environment: Usability problem describes how every interface should have a transparent navigation model, and how ASR systems generally lack this kind of functionality. This leads to the user not knowing if their input has been recognized or not. If the input is misrecognized, the user cannot understand why this misrecognition occurred and how to manage their next action (Hataoka et al., 2008). Transparency in the navigation model helps to build the robustness of the system, as constant feedback from a speech-controlled system keeps the user aware of what it does with the given input. This is especially important in the context of this study, as spoken dialogue will act as the medium for both the input and verification of the given input.

Out of vocabulary (OOV) problem, addressed by Hataoka et al., describes how meanings and locations may have multiple ways to express them, and how it is essential for ASR systems to handle this kind of situations. Improving ASR interfaces to solve the OOV problem was identified as one of the most urgent issues to be solved in the future (Hataoka et al., 2008). This problem was also referred to in the related works by Jeong, Kim & Lee (2003), where a framework designed to improve speech-driven information retrieval was introduced. One of the goals of the said framework was to solve the OOV problem (Jeong et al., 2003).

Robustness (in computer science) is the factor of how well the system can recover and function if/or an error occurs due to erroneous or unexpected input. Robustness in the ASR systems is recognized in various parts of the process. Hansen et al. (2008) address the variety of background noises existing in a vehicular environment to be one of the research challenges that must be addressed to achieve speech interactive systems that are reliable and natural to use. Also, speakers or drivers will experience cognitive stress from driving which would further modify their vocal effort to overcome noise levels within their ears. (Hansen et al., 2008). In the vehicular environment, the speech signal may experience degradation as noise belonging to the in-vehicle environment may interfere with it. The need for robustness is synonymous with dealing with the noises caused by the external and internal sources, as described in Table 2 (Hataoka et al., 2008; Kadambe, 2002).

Since the 1970s, there have been dozens of attempts to utilize speech enhancement techniques in hopes of suppressing the noise to improve the overall feasibility of speech-based interaction with the vehicular components. (Hong et al., 2004). In the study by Ding, He, Yan, Zhao & Hao (2008), both low-cost and robust mandarin speech recognition was being proposed. The noise suppression was conducted by applying two algorithms to firstly suppress the background noise and then to introduce a gain function to the remaining noise-reduced spectral components. (Hong et al., 2004; Ding et al., 2008).

Additional noise received while a speech interface is on standby affects how precisely the voice is recognized (Ding et al., 2008). These additional noises could end up making creating speech recognition errors. The study by Tam et al., (2014) highlights the importance of detecting speech recognition errors in ASR: Recognition errors directly affect the natural language understanding of a system with no suitable mitigations against it, such as an error-correcting dialogue manager (Tam et al., 2014). Related to this issue, the study by Jeong et al. (2003) focuses on decreasing speech recognition errors through a semantic-oriented approach. This approach aims to improve the recoverability from a situation where an erroneous query has been introduced to the ASR. The Semantic-oriented correction process tries correcting the erroneous input by assessing several related candidates and comparing these candidates with words from a domain dictionary, in hopes of discovering candidates with the most similarities to the original input. (Jeong et al., 2003)

The issue of handling different styles of pronunciation due to dialects and foreign language has also been studied in prior research. While pronunciation has been identified as a challenge for speech processing in multiple studies (Hunt, 2002; Zheng et al., 2017), the study by Ding et al. (2008) utilized the method of training the acoustic model with multiple accented mandarin speech databases that have been modified with pre-recorded in-car noise to simulate the target environment (Ding et al., 2008).

The introduction of the speech interface to the in-vehicle context reveals unique challenges. Zheng et al. (2017) point out that the technical challenges of implementing an in-vehicle dialog system contain similarities in implementing dialog systems towards general purposes. Context-related challenges are identified from in-vehicle data collection efforts and the development of an in-vehicle dialog system. The task of developing a natural spoken language understanding navigation dialogue system has been one of the highest demanded in recent years. It is desired to have an intelligent dialogue system, holding the conversational capability that of an assistant. The driver of a car would be providing the desired location information in a way that is natural for humans. This leaves the dialogue system alone with the task of understanding each bit of information gathered and coming up with the desired destination up for navigation (Zheng, 2017). Hunt (2002) describes the potentiality of the hands-free features that in-car speech recognition could provide: Being able to control the air-conditioning and the radio without taking your hands off the wheel is both convenient and a safety asset. Hunt also argues that the most interesting in-car application of speech recognition would be for navigation systems, as such systems offer great help in guiding drivers to their destinations in territories unfamiliar to the driver. (Zheng, 2017; Hunt, 2002)

Cloud-based intelligent assistance systems have been popularizing the in-vehicle speech recognition due to the high level of language understanding accuracy. However, the impact of cloud-based speech systems is seen to be limited to the ASR accuracy and the naturalness of Text-to-speech for dialog system development (Weng et al., 2016). Even after the popularization of these assistance systems, one of the persisting issues is how the speech recognition feature can still distract the driver, which leaves lots of consumers unsatisfied. (Weng et al., 2016; Khan et al., 2017)

3.4 Requirements derived from the literature

To successfully develop and evaluate a speech interface for the in-vehicle environment, it is crucial to acknowledge the environmental limitations and context-wise expectations that come with developing one for a specific environment. These expectations and environmental limitations were collected from prior literature and built into literature-derived requirements, which later are merged with the application context requirements to form the final requirements. Identified issues and solutions that are relevant to the design study are summarized into objectives to be fulfilled with the designed tool. These objectives will also act as the evaluation criteria for assessing the tool.

From the prior studies, the following requirements were retrieved. In Table 3, the term *tool* refers to the chain of applications and devices that together form the artifact in question.

Table 3. Requirements for developing a speech interface for the in-vehicle environment

Requirement	Description
R1	The tool should be capable of recognizing and understanding spoken language (Ding et al., 2008).
R2	The tool understands multiple meanings for the same location, acknowledging the OOV problem (Hataoka et al., 2008) (Jeong et al., 2003).
R3	The tool should function in a transparent fashion (Hataoka et al., 2010).
R4	The tool should be usable without visual distraction (Sandnes et al., 2008) (Hansen et al., 2008).
R5	The tool should be usable with minimal auditory distractions (Hansen et al., 2008).
R6	The tool should be usable with minimal cognitive distraction (Hansen et al., 2008).
R7	The tool should be usable without physical distraction (Hansen et al., 2008).
R8	The tool should be error-tolerant in speech recognition (Tam et al., 2014) (Jeong et al., 2003).
R9	The tool should function in the embedded environment (Herbig et al., 2010b) (Qian et al., 2009).
R10	The tool should understand speech without making specifications (Herbig et al., 2010a).
R11	The tool should withstand in-vehicle noises (Hataoka et al. 2010) (Kadambe, 2002) (Herbig, 2010a) (Ding, 2008).

As shown in Table 3, the retrieved requirements follow a trend of being more targeted towards how the tool is designed and how it would affect its users in the target application domain. These requirements provide a strong basis on the evaluation guidelines to be generated together with the domain-specific requirements.

4. Design

After gathering the requirements derived from the literature review, the development of the system could begin. The development process used the iterative development methodology as it was described in chapter 2.

4.1 Requirements development

Requirements for developing the system were designed to provide a structure of how the designed system would be built and to provide the evaluation criteria for the developed tool to be evaluated against. The developed requirements were constructed from two domain areas: The first domain was the existing knowledge base that provided the state-of-the-art methods and other relevant research work used in both development and evaluation. The second domain was the project

The sections follow the process of introducing the whole toolchain as a concept for the reader, to ensure straightforwardness: The reader understands what are the tools themselves, how do these tools operate regarding the context of this study, what kind of information do these tools relay when they interact with each other and how the process is seen by the end-user.

4.1.1 Gathering project requirements

These requirements were compiled from use-case meetings held with the local APPSTACLE project members and discussing with project members from Germany during other events related to the project, including hackathons and project status meetings. In these use-case meetings, the local project group discussed with the author about how voice-based communication methods could be utilized with the system designed in the APPSTACLE project. The local project group consisted of research fellows and PhD students, while the other project members involved with the requirement specification were working in the engineering industry. With the collaboration of project members local and international, a list of requirements could be compiled and validated at the end of May 2019. The list of requirements can be seen in Table 4. Two terms indicate the following items: *system* indicates a computational device intended to be accessed, and the *tool* indicates the designed artifact. The wording used to represent the requirement levels have been modified to match the environment of a use-case representation instead of representing the real-world environment.

Table 4. List of requirements derived from the project personnel.

Requirement ID	Requirement description
R1	The system must be operable with the tool by using speech.
R2	The tool should work within the same environmental limitations as a normal car.
R3	The tool should not introduce situations that interfere with the driving task.
R5	The tool should work with multiple users.
R6	The tool should provide navigation functionality in the vehicle.
R7	The tool should understand English locations.
R8	The tool should understand Finnish locations.
R9.	The tool should implement route planning.
R10.	The tool should be fault-tolerant: Withstand erroneous speech recognition situations.
R11.	The tool should withstand temporary service breakdowns: disconnections etc.

As shown in Table 4, the requirements that were collected with the help of the project personnel were more practical and function oriented. That is expected, as the amount of knowledge towards the other technologies utilized in this research was limited among the project personnel.

4.1.2 Developing the final requirements

With the project requirements specified and validated, they can now be combined with the requirements derived from the literature. This combining process resulted in us a list of requirements that can be seen in Table 5. The requirement order was based on what areas the requirements focus on and what tools are involved in each requirement.

For the final requirements, similar requirements were merged and requirements that were not directly relevant to the project were left out. The included requirements were deemed relevant for evaluating the artefact. The requirements were defined to provide evaluation guidelines for the following functionalities: The voice interaction tool functions how ASR based systems should function. The voice interaction interface can operate the navigation component by voice. The requirements should fulfill at least some of the in-vehicle domain-specific requirements, such as the expected robustness in error-correcting and voice recognition.

These domain-specific requirements define how much the user of the system has to provide attention to the tool for getting their task fulfilled. For example, if the system fails to provide the correct address, the user has to interact with the system again, which introduces unnecessary interference on driving. The requirements are designed to be fulfilled within the time- and resource limitations of this project. In a real-world situation, these requirements would be accompanied by requirements that would also focus on how efficiently each requirement is fulfilled.

Table 5. List of final requirements

Requirement ID	Requirement description
R1	The system must be operable with the tool by using speech.
R2	The tool should recognize the normal spoken language.
R3	The system must provide navigation functionality.
R4	The system should withstand erroneous situations in speech recognition
R5	The system should function in a vehicular environment.
R6	The system should provide route-planning.
R7	The system should function with multiple users.
R8	The system should inform the user of its functionality
R9	The tool should understand multiple meanings for the same location
R10	The system should be usable with minimal auditory distractions.
R11	The system should be usable with minimal cognitive distraction.
R12	The system should be usable without physical distraction.
R13	The system should be usable with minimal visual distraction.
R14	The tool should not introduce situations that interfere with the driving task.
R15	The system should withstand temporary service breakdowns.
R16	The tool should understand English locations.
R17	The tool should understand Finnish locations.

As shown in Table 5, the finalized requirements cover both the areas of development and the evaluation criteria against the target environment. Based on the limitations of this study, wording the requirements has been more lenient to suit the research topic of explaining how a voice-controlled interface can be designed, instead of estimating if this artefact fulfills the safety-critical aspects of the application domain.

4.2 Selecting the system components

This section describes and illustrates the tools and devices that were utilized in the artefact design process.

4.2.1 Echo Dot

The Echo Dot is a small, tube-shaped device that is equipped with lights, speakers and a microphone. It is a smart device that belongs to the Amazon Echo product family. It is designed for providing a communication interface between the end-user and the Cloud-based service called Alexa (Chung, Park & Lee, 2017). Communication between these services is handled via connecting to the internet by setting it up on a wireless network.

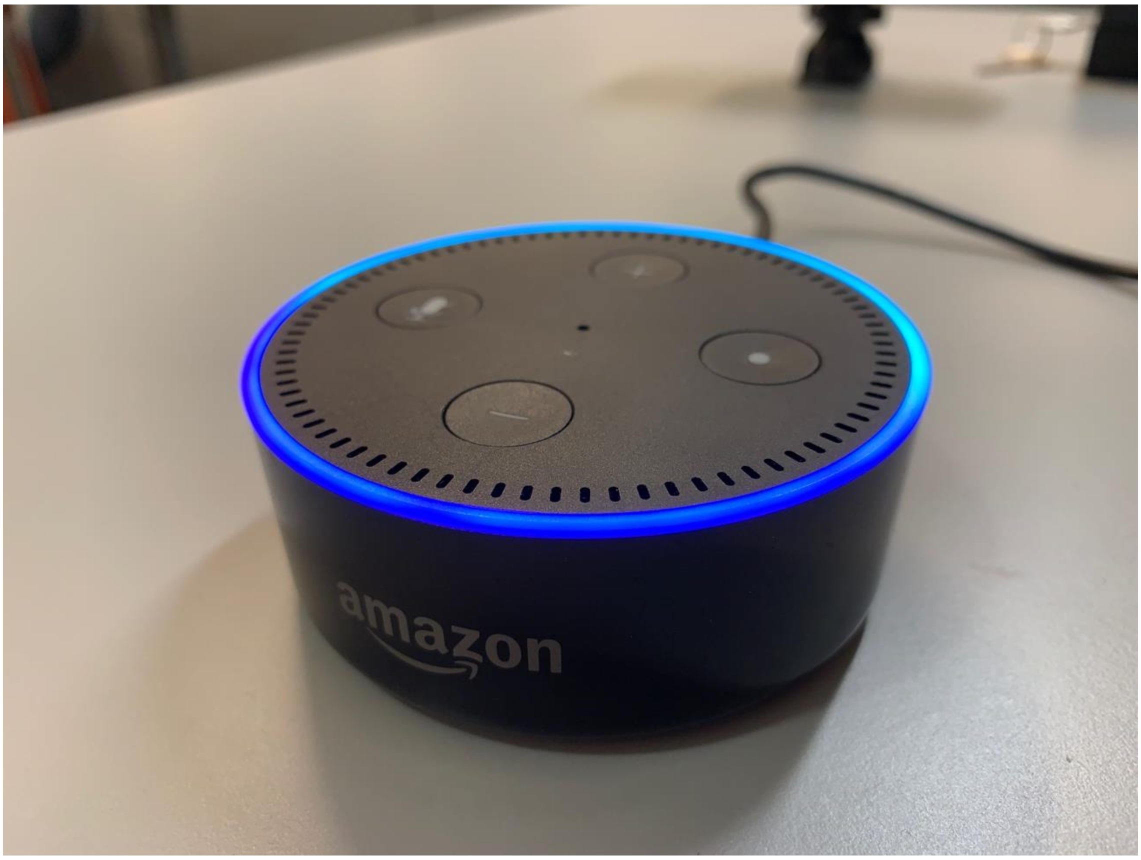
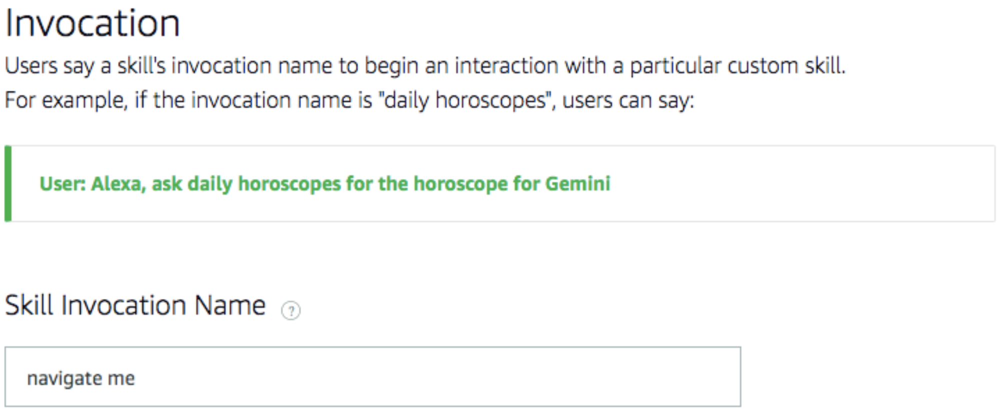


Figure 2. The product Amazon Echo Dot

Echo dot relays the phrase that the user spoke out to a cloud-based service that uses Automatic Speech Recognition to figure out what does the phrase contains. After estimating the best candidates for each word inside the phrase, the estimated phrase (utterance) is then relayed to the Alexa -service.

4.2.2 Amazon Alexa

Alexa acts as the command interface between the user and the navigation service. The service is based on skills, which are defined as the apps of the Alexa. These apps enable the user to perform various tasks and engage with other smart services by using your voice (Amazon, 2019). Upon invoking the Echo Dot device by speaking out a phrase to it, Echo records it and transmits the audio signal to the Alexa Voice Service (AVS), which is responsible for parsing the audio into a recognizable command that invokes a specific skill. AVS takes the audio through the process of speech recognition and natural language processing to get an estimate of the phrase that was spoken to the Echo device (Amazon, 2010a). The figures and screenshots shown in this section are taken from the web-based Alexa developer console, located in the AVS.



Invocation

Users say a skill's invocation name to begin an interaction with a particular custom skill.
For example, if the invocation name is "daily horoscopes", users can say:

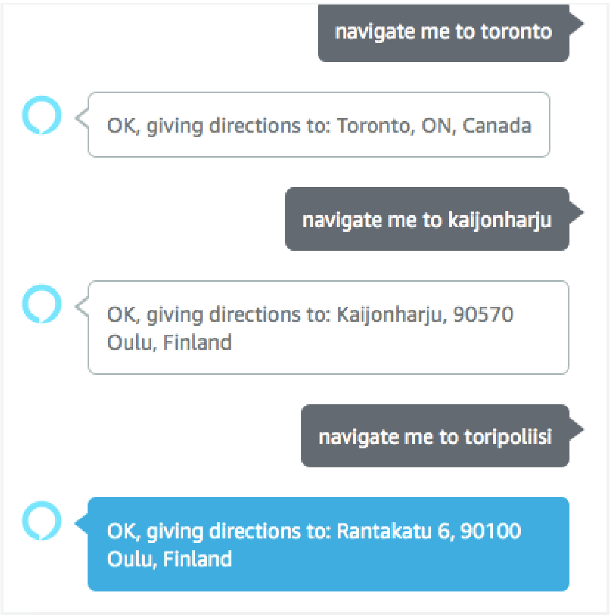
User: Alexa, ask daily horoscopes for the horoscope for Gemini

Skill Invocation Name

navigate me

Figure 3. Screenshot from the AVS illustrating the process of invoking the custom Alexa skill with the words “navigate me”

The custom skill is invoked by interacting with the Echo device by and including the invocation name in the spoken sentence, along with the desired location. This interaction process is illustrated in Figure 4.



navigate me to toronto

OK, giving directions to: Toronto, ON, Canada

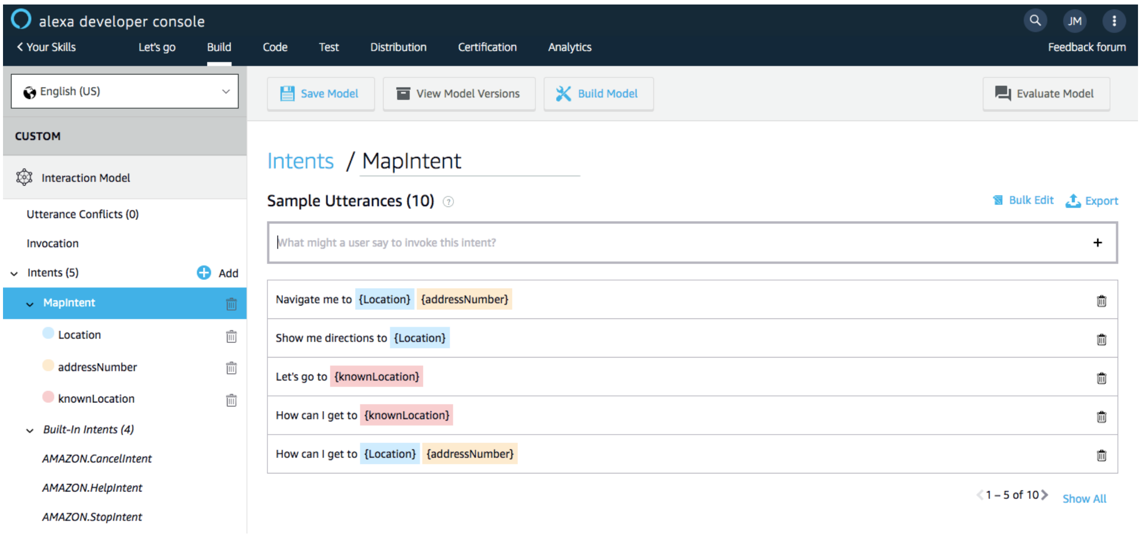
navigate me to kaijonharju

OK, giving directions to: Kaijonharju, 90570 Oulu, Finland

navigate me to toripoliisi

OK, giving directions to: Rantakatu 6, 90100 Oulu, Finland

Figure 4. Screenshot from AVS illustrating how the Alexa skill is invoked.



Navigate me to {Location} {addressNumber}

Show me directions to {Location}

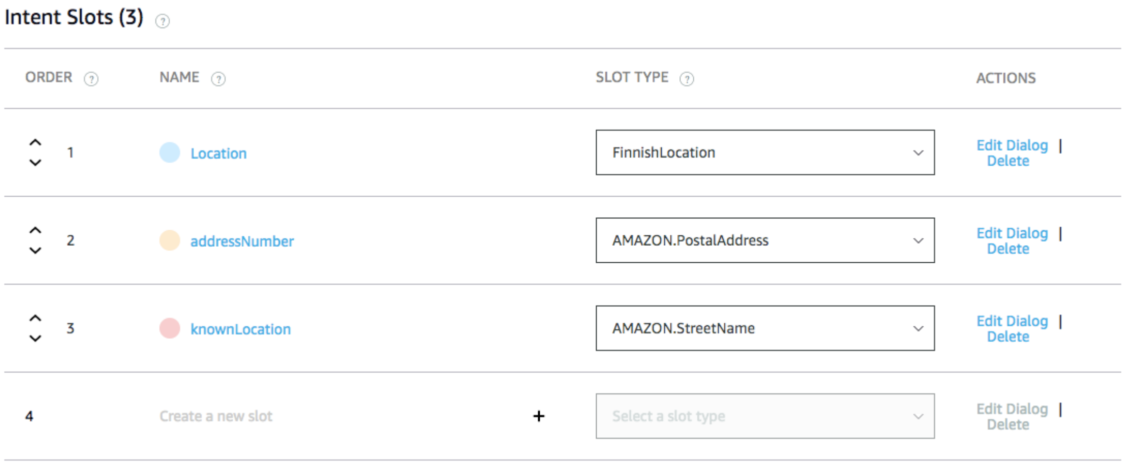
Let's go to {knownLocation}

How can I get to {knownLocation}

How can I get to {Location} {addressNumber}

Figure 5. Screenshot of the Alexa developer console MapIntent functionality, showing how the custom skill can be invoked.

Sample utterances are a list of different types of phrases that activate the custom intent. This process can be seen as setting up the semantic parsing process of the designed system: Instructing the skill to classify each type of word found in the sentence into pre-determined categories. As seen in Figure 5, *MapIntent* can be activated not only by mentioning the invocation name 'navigate me', but by using similar sentences that involve the intention of traveling and the desired location (the POI). As per the design principles of Alexa skills, the designer of the skill has to ensure that each sentence synonymous with the intent of navigating is captured. There are multiple ways of describing the desired location. These location-describing words are captured with the usage of Intent slots, as illustrated in the following Figure 6.

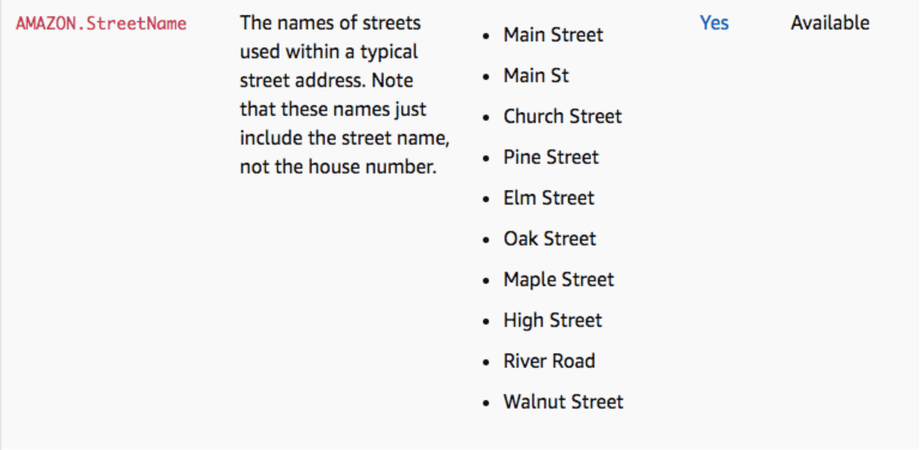

Intent Slots (3)

ORDER	NAME	SLOT TYPE	ACTIONS
1	Location	FinnishLocation	Edit Dialog | Delete
2	addressNumber	AMAZON.PostalAddress	Edit Dialog | Delete
3	knownLocation	AMAZON.StreetName	Edit Dialog | Delete
4	Create a new slot +	Select a slot type	Edit Dialog | Delete

Figure 6. Intent slots that capture the user's desired location, located in the AVS.

As shown in Figure 6, different intent slots can be designated to capture different types of location defining information. Slots need to have a specified slot type that defines what kind of information the slot can capture. The intent slots named *addressNumber* and *knownLocation* utilize the slot types that are pre-made by Amazon. These slot types are populated with numerous types of values that are generally known to be used in these slot types. *AMAZON.StreetName*

There are pre-made list types available for hundreds of different categories, such as names of different animals, titles of books and names of cities in a specific language. The following Figure 7. lists the type of contents located in *AMAZON.StreetName*.



AMAZON.StreetName	The names of streets used within a typical street address. Note that these names just include the street name, not the house number.	- Main Street - Main St - Church Street - Pine Street - Elm Street - Oak Street - Maple Street - High Street - River Road - Walnut Street	Yes	Available

Figure 7. Example of contents in the AMAZON.StreetName -Slot type, located in the AVS.

As shown in Figure 7, the contents of the list types are designed to contain all of the possible examples that fit into the specified category. These skills are designed to provide various kinds of help to the user: Information about weather and reading out text articles from various sources. There are also skills that help the user to operate other smart devices, such as turning on lights, setting the AC and remotely open the garage door. The skills can be modified to your liking, and users can create custom skills that fit their own needs (Amazon, n.d.).

In the context of our study, a custom skill was developed for Alexa. This skill instructs the service to handle incoming intent requests in a specific fashion that is visualized in Figure 8.

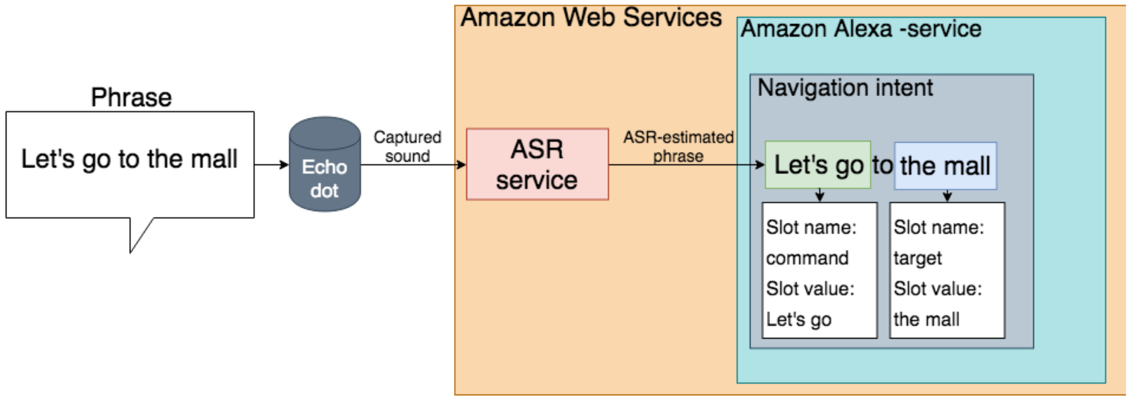


Figure 8. Illustration of how a phrase is captured and processed into the Alexa service.

Upon receiving voice lines that Alexa recognizes as intents related to the navigation skill, Alexa relays this information to the cloud platform running the code that handles the confirmation of the location and further relays the confirmed information to the in-car computer interface.

4.2.3 AWS services: Lambda and IoT

AWS (Amazon Web Services) is a subsidiary of Amazon, designed to provide cloud-computing services and platforms to customers ranging from individuals to large organizations and companies. The services running in AWS are designed to work seamlessly with each other and they function on a pay-as-you-go -fashion.

For the context of our study, AWS provides us the platform for monitoring and customizing the designed speech interface. The Alexa skills service is connected to a chain of services hosted in the AWS called Lambda and IoT. Figure 9 portrays how each of these services communicates with each other and achieve the aforementioned functionality.

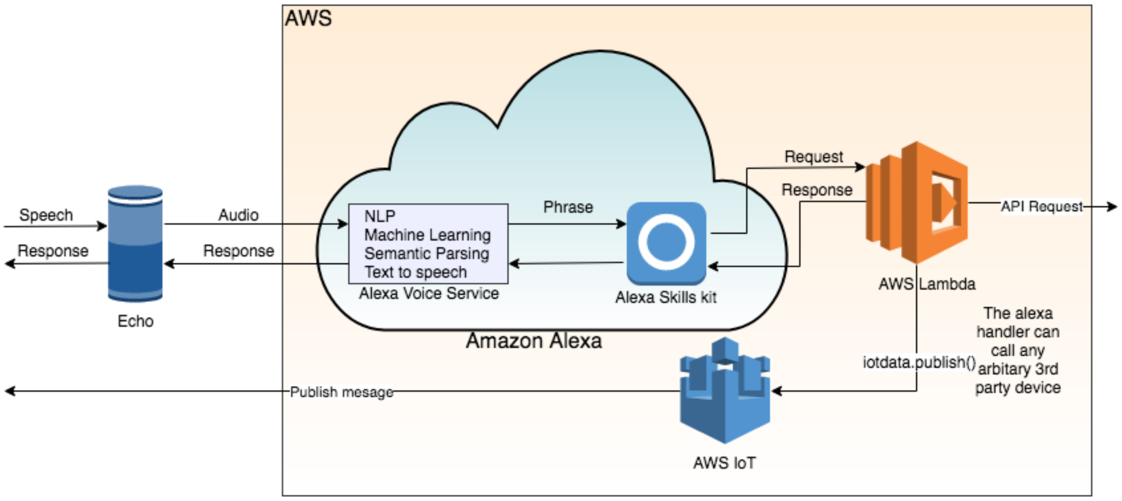


Figure 9. Illustration of Amazon services and how they interact with each other

The program running inside the Lambda service provides handling for each custom intent that has been set up to trigger the lambda function. Figure 10 illustrates the designer view: a function that has been set up inside the AWS Lambda service.

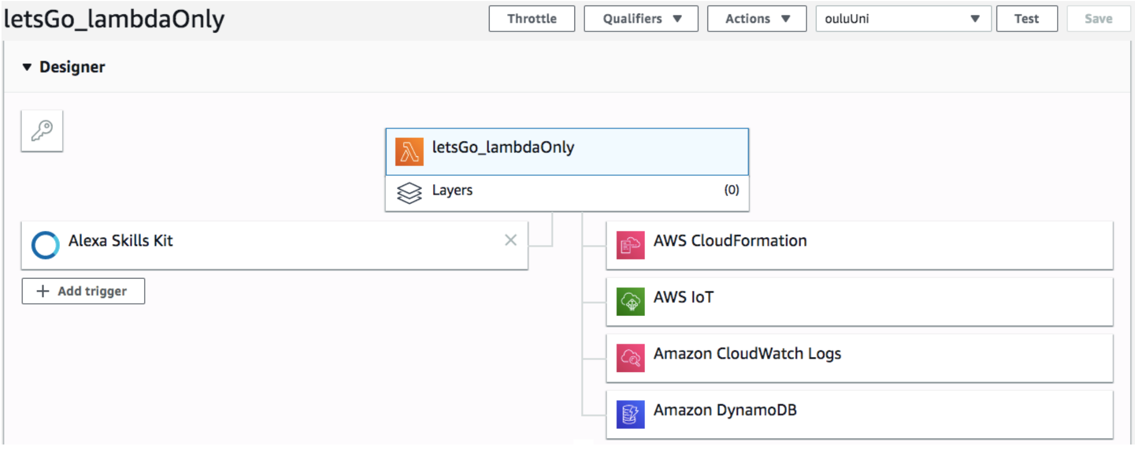
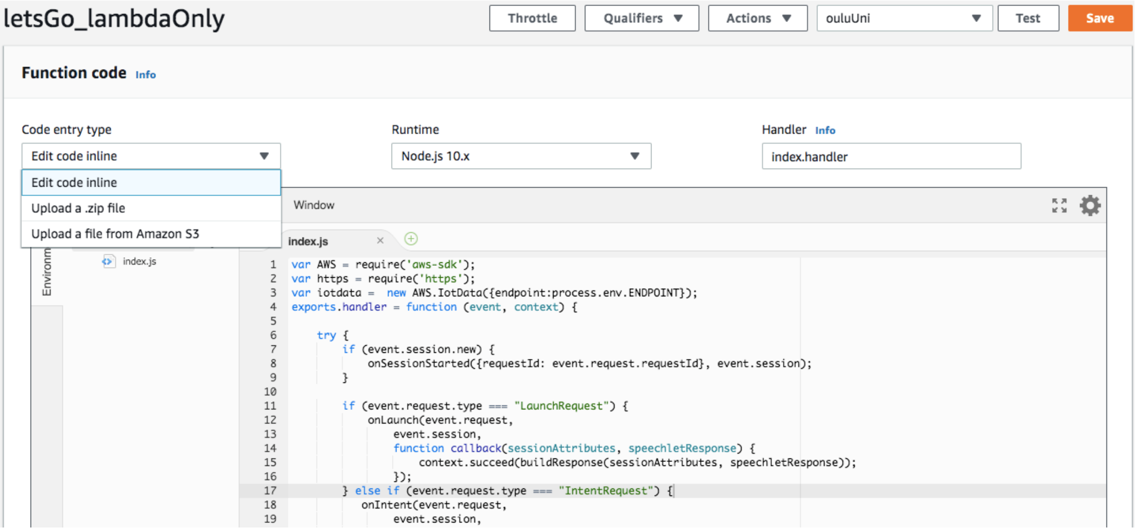


Figure 10. The designer tab in AWS Lambda provides a general view of the lambda function and its resources.

The left side of the resources view is reserved for the resources that are designated to trigger the lambda function. The right side shows the resources that the function has been permitted to utilize. The services called AWS CloudFormation and DynamoDB are there to provide scalability for serving multiple users and storing information related to the service managed by the lambda function. For this study, these two services are used for storing log files of the lambda service for debugging purposes. These logs are made viewable with the CloudWatch Logs -service. AWS IoT service provides the capability for the lambda function to trigger a “publish to broker” -command, which is then received by navigation tool subscribed to this same broker.

As Lambda is a web-service provided by AWS, the configuration of the lambda functions can be done in multiple ways. Figure 11 illustrates how the intent handling function can be customized from the web page.



Code entry type: Edit code inline

Runtime: Node.js 10.x

Handler Info: index.handler

```
var AWS = require('aws-sdk');
var https = require('https');
var iotdata =  new AWS.IotData({endpoint:process.env.ENDPOINT});
exports.handler = function (event, context) {

    try {
        if (event.session.new) {
            onSessionStarted({requestId: event.request.requestId}, event.session);
        }

        if (event.request.type === "LaunchRequest") {
            onLaunch(event.request,
                event.session,
                function callback(sessionAttributes, speechletResponse) {
                    context.succeed(buildResponse(sessionAttributes, speechletResponse));
                });
        } else if (event.request.type === "IntentRequest") {
           onIntent(event.request,
                event.session,
```

Figure 11. Web-based inline code editor provided by AWS Lambda.

As seen in Figure 11, Lambda accepts functions in a variety of languages by providing them specified runtime environments. Code doesn’t necessarily have to be entered via the inline code editor; it can be uploaded to the service directly with a zip file or by utilizing the uploading services provided by AWS.

By utilizing Lambda, the Alexa skill can now be configured to activate programs that further extend the functionality of our speech interface: Opening the communication between the speech interface and the component providing us the navigation capabilities, the in-car computer interface.

4.2.4 Google Maps APIs: Geocode and Directions

In this study, Google Maps services are being utilized to verify the locations of our requests and to provide the routing guidance to fulfill the navigation task. The calls made for the Google Maps API are visualized in the system architecture diagram located in Appendix B1. Calling the API requires the user to acquire an API key to identify the caller and authenticate their request. The designed system calls the Google Maps API in two separate instances, **geocoding** for verification and **directions** for route guidance. The calls along with their parameters are visualized in Figure 12.

Query 1: geocode api

/api/geocode/json?address={DESIRED_POI}&key={API_KEY}

Query 2: directions api

/api/directions/json?origin={ORIGIN_LATLONG}"&destination={DESIRED_POI_LATLONG}&key={API_KEY}

Figure 12. The API calls with the parameters explained.

The calls illustrated in Figure 12 are received by the Google Maps API service and the service returns an answer to each call. The contents of this answer can vary from error codes to the estimated location of the desired POI made in the API call.

4.2.5 Navigation interface: Android-based smart device

In the context of this study, a smart device capable of running an Android-based operating system is used as the in-vehicle navigation interface. The smart device runs a custom application that is set up to accept messages from the AWS IoT communication service. The main functionalities of this android application are explained in more technical detail in the development section. This section introduces the navigation device along with its functions and libraries that are utilized within the navigation tool.

Figure 13. illustrates the navigation device and an example of how it would be utilized within the in-vehicle environment. The navigation interface should be safely observable from the viewpoint of the driver. The navigation interface can be run from other Android enabled smart tools available. Our example can be mounted on the dashboard but having an in-car computer run the navigation software instead is a viable option as well. This way the application can be on the car computer screen, usually located in the middle of the dashboard.

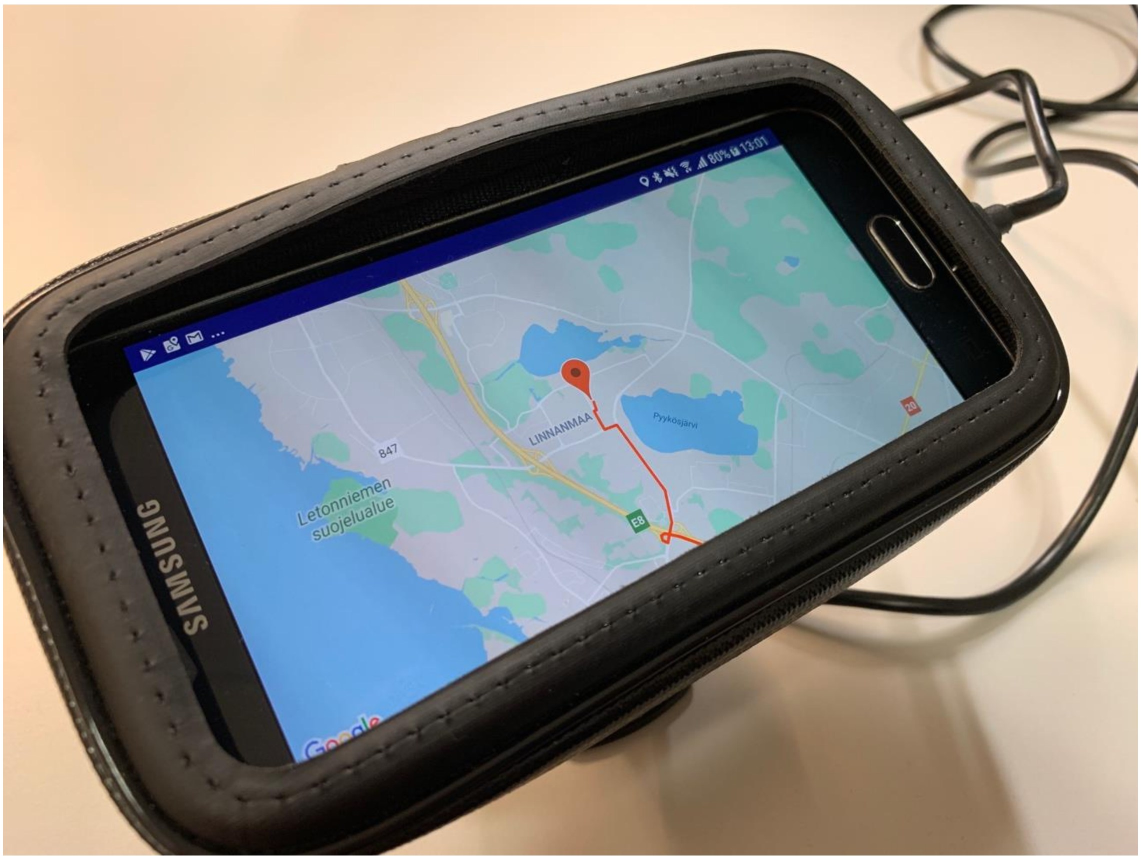
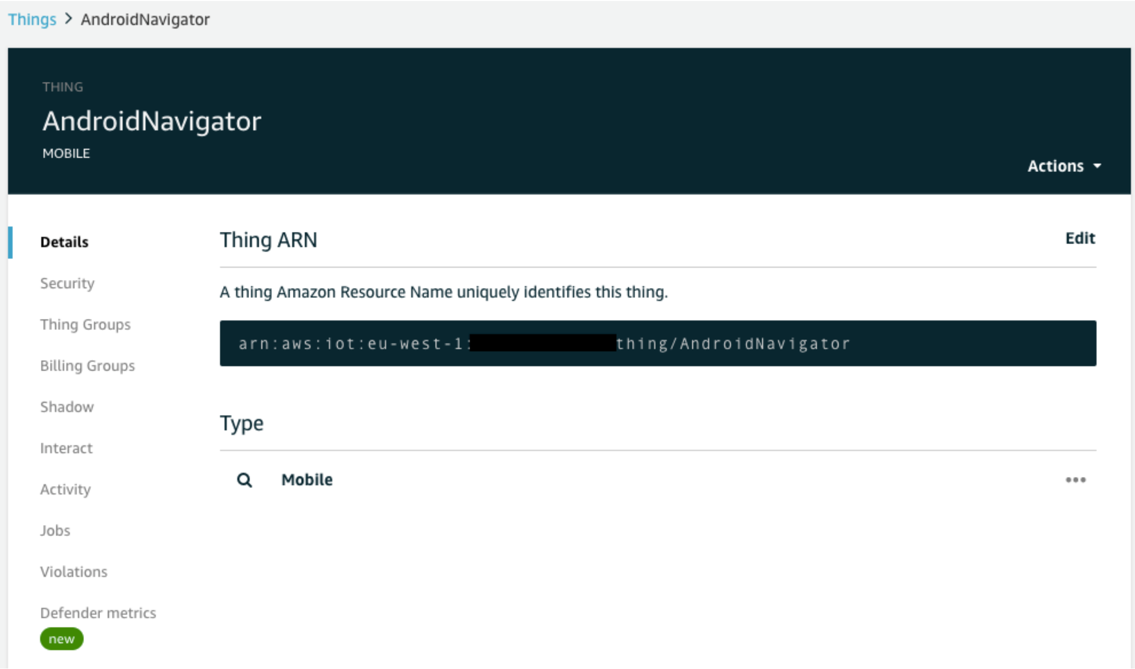


Figure 13. Navigation interface in the form of Samsung S6 -smart device, mounted on a stand.

The android-based navigation interface utilizes the MQTT messaging protocol to receive location requests from the Lambda function activated by the custom Alexa skill. To enable the communication between these two applications, the AWS IOT service is set up to provide an endpoint in which these two applications can publish and receive messages from one to another. Figure 14 illustrates how the android application is given an AWS endpoint as it is assigned as a Thing. Figure 15 shows how the android-based navigation device connects with the AWS IoT with the assigned endpoint. The information shown from the connection is displayed in their designated text boxes.

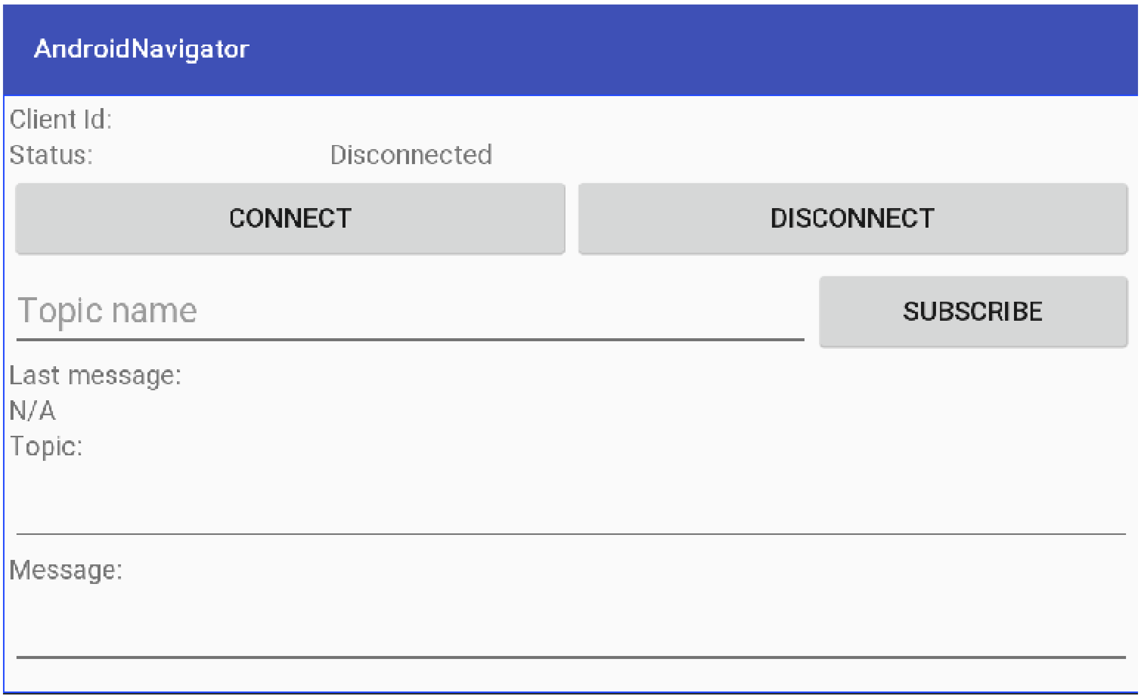


A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:eu-west-1:[redacted]thing/AndroidNavigator
```

Mobile

Figure 14. AndroidNavigtor is assigned as a Thing in the AWS IoT



Client Id:
Status: Disconnected

Last message:
N/A
Topic:

Message:

Figure 15. MQTT subscription activity in the navigation interface.

As the navigation interface is activated by receiving a message via the MQTT subscription, the navigation interface displays a new window where the Google Maps SDK for Android is utilized to provide a map utility for this research. A visual representation of the Maps utility is provided in the figures located in chapter 4.4.2.

4.3 Designing the system structure

This chapter defines how the overall system structure was designed and how its components were designed to function with one another. This design process included the description of the system workflow, accompanied by figures for visual presentation.

4.3.1 General structure

The structure of the system was based around three main components that were deemed necessary to fulfill the main research challenges. The selected main components were chosen based on the examination of the existing literature and the component's estimated ability to provide the desired functionality. These components were Amazon Echo and Alexa, Amazon Lambda and the Android-based navigation interface. Amazon Echo and its underlying services were used for providing the required ASR and NLP functionality. Amazon Lambda services were used to program the retrieved voice commands to verify the POI and transmit the verified location. The navigation interface was used to retrieve the verified location from the Lambda function and provide the user with routing guidance between this location and the current location of the user, thus fulfilling the research objective.

The architectural structure of the system is visualized in Appendix B1. The system is designed to run in an in-vehicle environment, but as how the system design requirements emphasize mobility and embeddedness of the application environment, the entirety of the system could be run in any kind of environment where verbal communication with the communication interface is not restricted, and visual contact can be made with the navigation interface. The user initiates the process by speaking to the Echo device, which then transmits the received speech as an audio signal towards the Alexa services, where the speech is processed, parsed and transmitted to the Lambda function in JSON. Lambda function confirms the location with the Geocoding API and transmits the now verified location back to the Alexa and towards the navigation device by AWS IoT. The android-based navigation device retrieves this location information via MQTT and requests navigation data from the Directions API. The user is given both the verified location spoke by Alexa and the map with the route from their current location to the requested POI is shown on the navigation interface.

4.3.2 Network design

Figure 16 illustrates the network setup for the designed voice-based interaction system. To enable remote connectivity between the Amazon Echo and the navigation interface with the AWS and API -services, a mobile hotspot utilizing a 4G network was used. MQTT connection handled the transmission of POI information between the AWS IoT and the navigation interface. API requests were conducted by using HTTPS GET requests, which returned us the payload based on the correctness of the request.

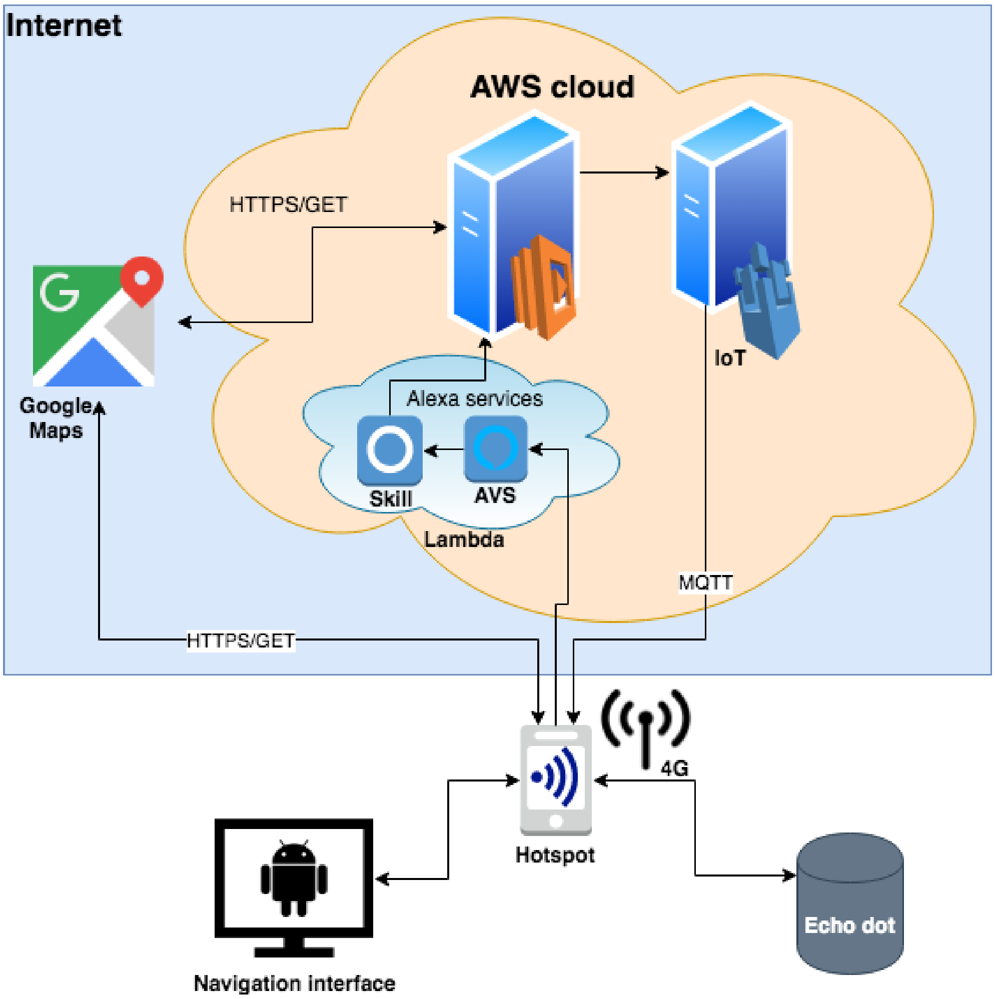


Figure 16. Illustration of the network setup.

With the illustrated network setup, the designed system can be utilized with more mobility available. However, network availability has to be ensured for the device to provide a mobile hotspot.

4.4 Developing the system

In this section, the development process of the complete system is presented. Each of the components and the necessary main modules and libraries that are relevant to the research question. All the modules, functions and services that are relevant to proceeding the study are presented here. Each section is accompanied by visual and textual descriptions of their contents and functionalities.

4.4.1 Alexa & Lambda

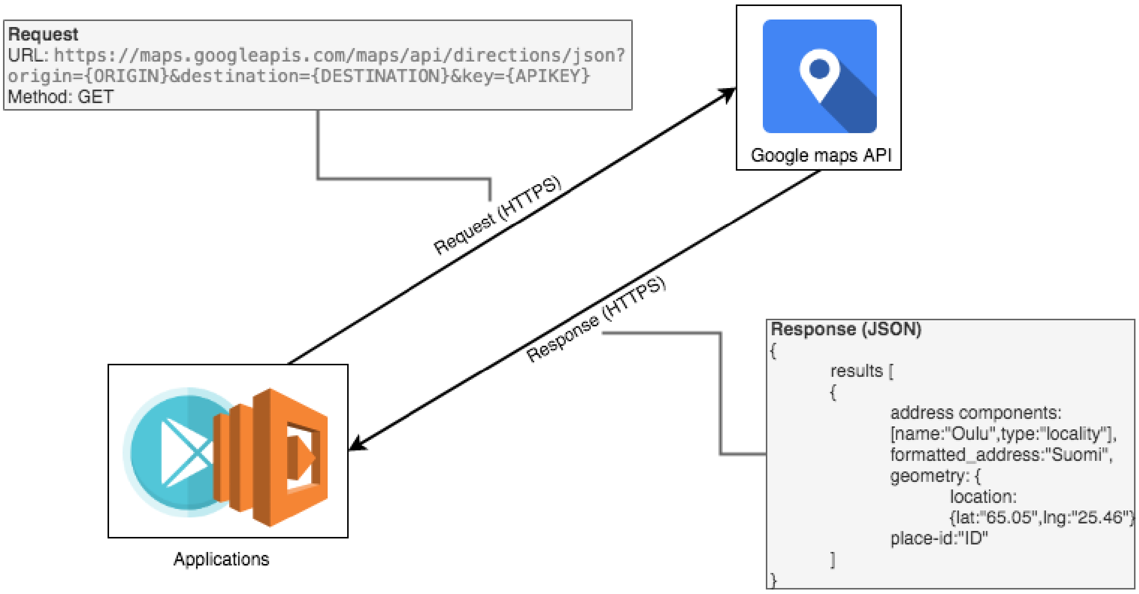
Alexa acts as the command interface between the user and the navigation service. Upon receiving voice lines (utterances) that Alexa recognizes as intents related to the navigation skill, Alexa relays this information to the cloud platform running the code that handles the confirmation of the location and further relays the confirmed information to the in-car computer interface. After Alexa confirms the processed audio signal to contain speech that matches the invocation request of the MapIntent -skill, Alexa categorizes the contents of the spoken audio signal into the pre-determined slots and sends this information to the AWS Lambda. A JSON code example of this is provided in Appendix A1.

The Alexa service categorized the desired POI “Oulun yliopisto” into the slot called Location, where the slot had been taught to capture pre-defined locations for this study. The lambda is now activated after receiving the *intentRequest*. Upon activation, the lambda function activates an *intentHandler* based on the received *intentRequest*. This activation process is shown in Appendix C1. After calling the correct *intentHandler*, the values of the *intentRequest* are gathered with the *handleMapIntentRequest* function, which is presented in Appendix C2.

This snippet of code checks which one of the location slots contains the desired POI. If the POI is in the Location slot, the *addressnumber* -slot could also contain information. If both slots contain values, they are combined as one to generate a viable address. The retrieved address or location is then placed into the command -variable and that variable is given as a parameter for the verifying function *getConfirmedLocation*.

The following lines of code are from the getConfirmedLocation function, which resides in the program activated by the skill invocation from Alexa. The program is hosted in AWS lambda. The function *getConfirmedLocation* is presented in Appendix C3. This function aims to confirm the validity of the address or location received from the user. The confirmation process attempts to retrieve a location that corresponds to the user’s desired POI, which now resides in the *location* parameter. This is done by calling the Geocode -API to confirm the validity of the requested location. As we are working with a function that awaits a response from a remote location, the function is asynchronous.

The interaction procedures between the application and the Google API are visualized in the following Figure 17. The API request task is described in chapter 4.2.4.



```
Request
URL: https://maps.googleapis.com/maps/api/directions/json?
origin={ORIGIN}&destination={DESTINATION}&key={APIKEY}
Method: GET
```

```
Response (JSON)
{
        results [
        {
                address components:
                [name:"Oulu",type:"locality"],
                formatted_address:"Suomi",
                geometry: {
                        location:
                        {lat:"65.05",lng:"25.46"}
                place-id:"ID"
        ]
}
```

Figure 17. Request and response model example between the client(s) and the API server.

As seen in the code snippets presented in Appendices A2 and A3, making an API request delivers us a JSON -file that contains the results of our query. These results are dependent on whether the request was successful, and whether the requested location was valid or not. The following two JSON code snippets show us the difference between a successful and a failed geocoding query. A successful query for the Geocoding API verifies that the desired POI retrieved from the user is a valid location. If the API request fails, the desired POI is not deemed as a valid location and the user is prompted to provide a new location.

After successfully retrieving either one of these JSON files from the Geocoding API, the contents of this retrieved JSON file are inspected with the remainder of the *handleMapIntentRequest* -function, as shown in the appendix C4. In this function, it can be seen that the results of the JSON retrieval are set into the object called "obj". This object is then inspected to see if the request was successful or not. Upon retrieving a successful request, the user is informed of this by calling the callback function and using the retrieved address from obj.results[0].formatted_address as the location indicator. Two separate processes happen here:

Process 1: The user has confirmed the verified address and how it is being projected on the navigation interface. Alexa receives a JSON file with the instructions to speak out the phrase located in the *response* -JSON object, provided in Appendix A4. This process lets the user of the system know if the request they inquired was successfully handled or not.

Process 2: The coordinates of the confirmed location are sent to the navigation interface with the use of sendLocation -function, which utilizes the AWS.IotData -library that handles message publishing between the lambda function and the navigation interface. The setup- and publishing of the message are visualized in Appendix C5. In this function, the parameters *params* contains the pre-defined MQTT topic where the navigation interface has subscribed to. The payload contains the confirmed location coordinates *coords* as its parameters and is preceded by the tag *"GOTO:"*, which identifies the message as the one that initiates the routing functionality in the navigation interface.

After these two processes have completed, the lambda function turns itself off and it can be activated again by re-invoking the MapIntent skill.

4.4.2 Navigation interface

The application handling the navigation interface and running on the android-operated smart device has been set up to listen to a customer-specific AWS IoT client endpoint that corresponds to the pre-defined client- and customer identification keys. This endpoint acts as the MQTT -based message broker that the lambda function publishes to and the navigation interface subscribes to. All the client identifying information (e.g. Api-key, client id) are stored in the Gradle build automation tool's environment configuration file *gradle.properties*.

Upon subscribing to the shared messaging topic with the lambda function and successfully receiving a message, the message contents are carried into the *MapsActivity* -class, which handles both routing and displaying the navigation interface. The code that initiates the MQTT subscription process and starts the message handling function *subscribe* are handled by the *MainActivity* -class as shown in Appendix C6. For now, the function proceeds upon receiving a message that contains the routing -specific tag "*GOTO:*". After the message has been confirmed to contain a location, the received message is added as extended data to the soon introduced activity class *MapsActivity*. The GoogleMap object starts its initialization process when the MapsActivity is accessed. First, a private field *mMap* is declared for the class MapsActivity:

```
private GoogleMap mMap;
```

The MapsActivity sets a callback object which triggers when the GoogleMap object is ready to be used:

```
mapFragment.getMapAsync(this);
```

The code snippet located in Appendix C7 shows the process that generates the map and invokes the other associated methods that manipulate the map: Adding the user's current location and moving the camera on top of it. The location of the user is retrieved by utilizing LocationManager to capture the Global Positioning System (GPS) coordinates in *getLocation()*. The method *datadownload.execute()* calls the DataDownload class method *execute()*, which initiates the process of retrieving the routing guidance between the current location and the desired POI retrieved from the MQTT transmission in MainActivity.

To generate the routing between our current location and the desired POI, the Google Maps API is used similarly as during the confirmation of the location in the Lambda function. This time the directions API is called to provide us the routing directions from the *origin=* location of this navigation device to the location *destination=*, that was prompted by the user and confirmed with the use of lambda function running the geocode request. This process can be found written in Appendix C8.

The function *getDirections* takes the API request URL combined with the origin and destination information as its parameter, named *requestURL*. The function carries out an HTTP GET request on the address located in *requestURL*. The google maps backend answers to this request by taking the location-based information from the URL and attempts to provide a response in the form of a JSON file. The JSON file resulting from the API request is shown in Appendix A5. For a better viewing experience, only one of the *step* -objects are left in the code snippet.

A routing plan between our location and the target location has now been successfully retrieved. For the context of this study, no progression is made any further with this information, as proving that the system can successfully retrieve and provide routing instructions was deemed enough to prove the feasibility. However, an example of how this information can be used to provide route guidance between two places is briefly introduced in the next section.

The routing data exists within the JSON in two formats: the textual *html_instructions* used for speaking out the movement instructions and in the form of encoded polylines. In this example, the polylines are utilized to "draw" the route between these two locations into the navigation interface. The code snippet for this is located in Appendix C9. It visualizes the process of decoding each of the *polyline.points* -object found in the JSON file. Every *steps* object contains a *polyline.points* -object to be decoded, which are found in every *legs* -object, which are found in the *routes* -object of the delivered JSON file. The decoded *polyline.points* object provides us a list of GPS coordinates in the form of latitude and longitude values. These values dictate the start- and endpoints for all the lines to be drawn on the map, and they are stored in the *lineOptions* -object of the *PolylineOptions* -class.

When all the lines are added to the *lineOptions* -object, it is used as the parameter for the method *addPolyLine* that is called on the GoogleMap object. The results of this process can be seen in Figure 18, as the red-colored route drawn between two locations on the map.

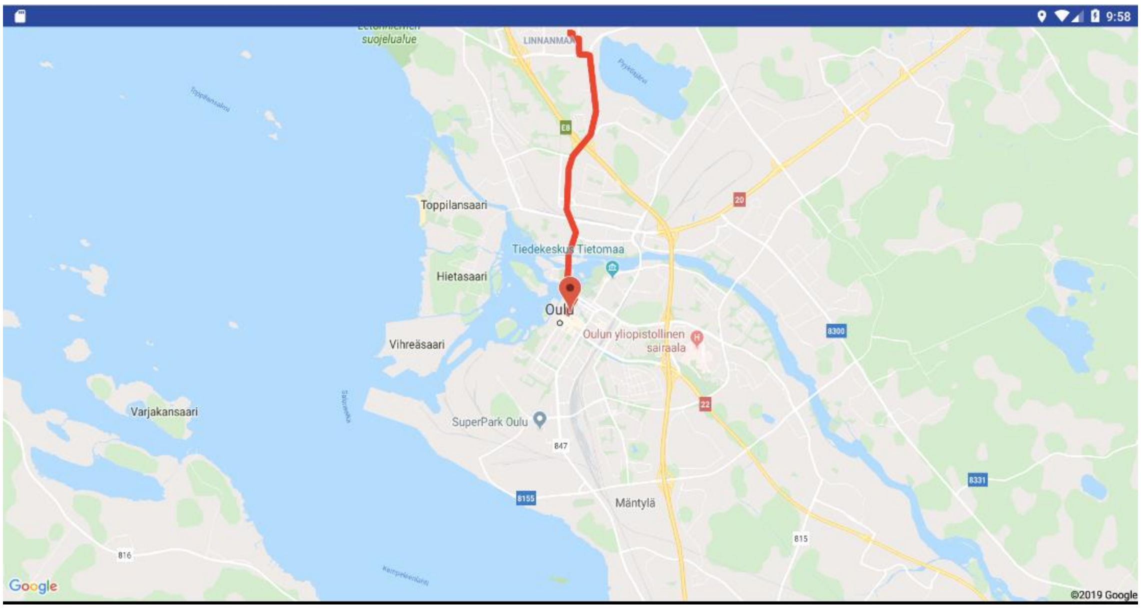


Figure 18. The resulting navigational view is displayed in the smart device.

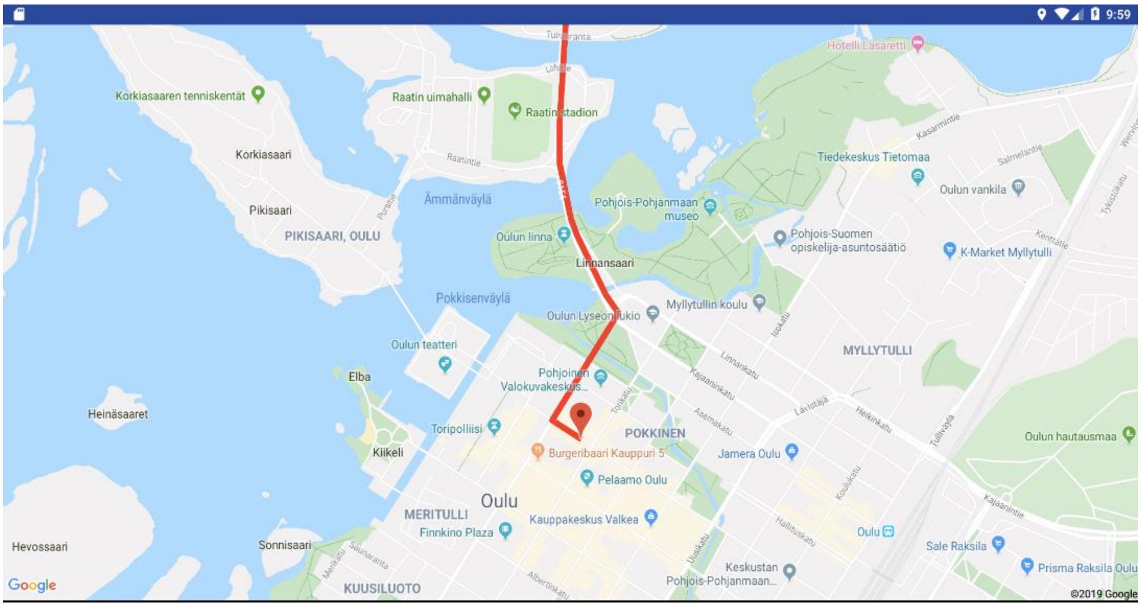


Figure 19. By zooming in, the routing guidance can be viewed more precisely.

As seen in the first view illustrated in Figure 18. The red line represents the routing information that was parsed from the JSON file. In Figure 19, the route can be seen to follow general traffic guidelines. The user has now gained the knowledge of how to traverse between their current location and the desired POI, therefore fulfilling the use case.

5. Evaluation

This section presents how the artifact was evaluated against the requirements that were generated by merging before the implementation of the system. The evaluation procedure was carried out by the author throughout each of the design cycles. Requirements focusing on the system were evaluated as the development process of the complete system was concluded. Evaluation of the tool was carried out in a similar fashion. The design- and development process was carried out from February 2019 to November 2019. The evaluation process was carried out by the author in a laboratory setting, as the focus of the evaluation was to assess the functionality of the artifact against each requirement. This was made to prove that the system fulfills the purposes defined in the requirements section. Evaluation of how this system fulfills these requirements in a real vehicular environment was not within the scope of this research.

In addition to the evaluation process, a separate demonstration of the specified requirements was presented during the use-case demonstration showcase as seen in Figure 20. The demonstration of the use-cases was held alongside the final project meeting in Eindhoven, the Netherlands in mid-December 2019.

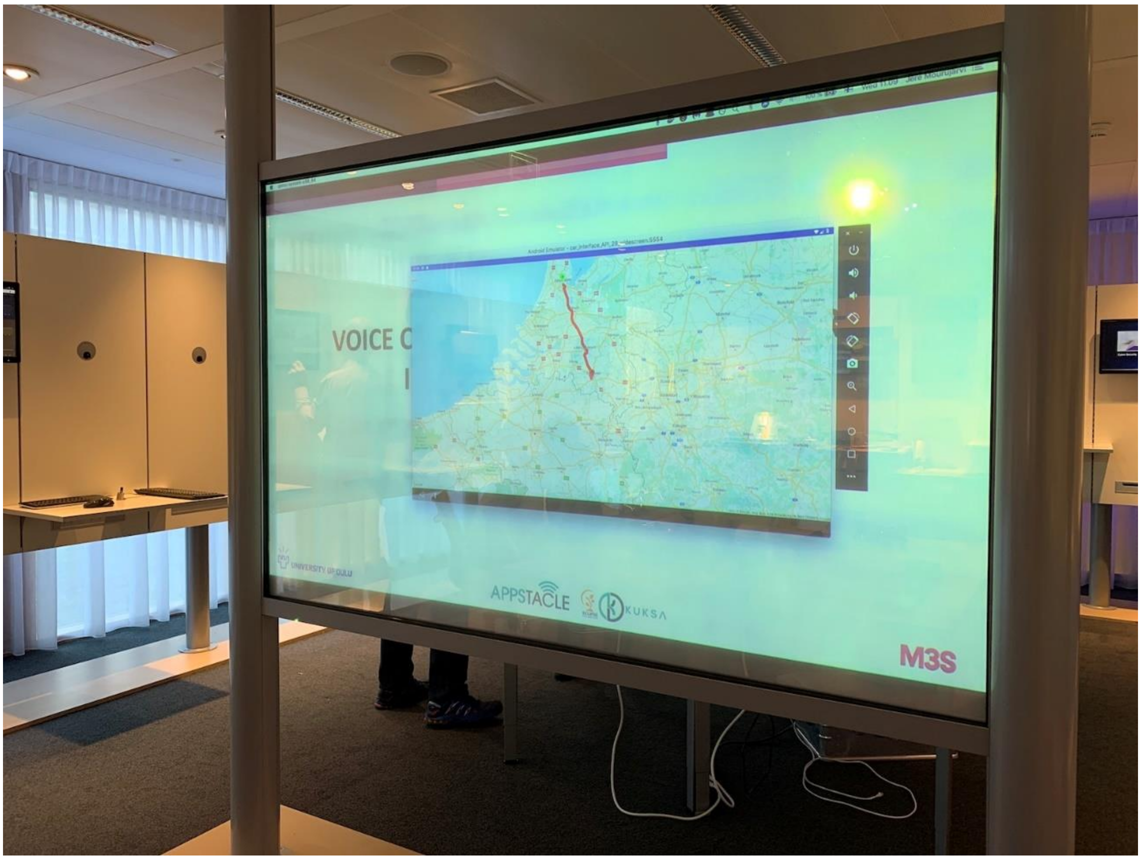


Figure 20. Demonstration of the use-case and its navigation functionality.

Each of the requirements is marked with a description of the said requirement and a status attribute that discloses if said requirement evaluation was deemed achieved, partially achieved or not achieved. Each requirement also has an evaluation point which contains a description of how said requirement had achieved its status.

R1: The system must be operable with the tool by using speech
Status: Achieved
Evaluation: The tool was capable of providing operations with the use of speech, as long as the custom skill was invoked properly.

R2: The system should recognize normal spoken language
Status: Partially achieved
Evaluation: As per the design principles of Alexa, the conversation had to be initiated with the invoke phrase, limiting the ways to interact with the tool. To fully achieve this requirement, a list of all possible phrases synonymous with the invoke phrase would need to be made.

R3: The system must provide navigation functionality
Status: Achieved
Evaluation: The system could successfully verify and provide the desired POI on the navigation interface.

R4: The system should withstand erroneous situations in speech recognition
Status: Partially achieved
Evaluation: The system could re-prompt the user to provide the desired POI again, but in instances where a similar-sounding but different POI was found, the re-prompting did not occur. This is a design-level issue, which stems from the existence of multiple locations with similar or same names. Prompting the user to choose one of the possible locations was not within the scope of this study and will be further discussed in the limitations. However, the Alexa service can be configured to ask for confirmation on all prompts.

R5: The system should function in a vehicular environment
Status: Achieved
Evaluation: The system could provide its functionality while deployed in a vehicular environment.

R6: The system should provide route-planning
Status: Achieved
Evaluation: The system could successfully retrieve the direction information from the current location to the desired POI.

R7: The system should function with multiple users.
Status: Achieved
Evaluation: The tool accepts speech from multiple users.

R8: The system should inform the user of its functionality
Status: Partially achieved
Evaluation: The system gives feedback by voice on how each command is processed: A successful command makes the system return a spoken phrase "Navigating to {POI}" while transmitting the information to the navigation interface. Incorrect queries re-prompt the user to try again. Related to R4, the existence of multiple locations is not informed to the user of the system.

R9: The tool should understand multiple meanings for the same location
Status: Not achieved

Evaluation: The fulfillment of this requirement is dependent on how well the context-specific skill types (e.g. AMAZON.StreetName) are constructed to include these synonymous POIs. In the context of this study, the missing POI synonyms were added to the self-made list of locations. This process should be carried out for all possible locations and synonyms, which was not realistically possible.

R10: The system should be usable with minimal auditory distractions.
Status: Achieved
Evaluation: The user interacts with the tool by invoking the function and stating the desired POI.

R11: The system should be usable with a minimal cognitive distraction
Status: Partially achieved
Evaluation: The user expends cognitive resources on interacting with the system in a similar fashion as it would expend when interacting with a traditional navigation system. However, re-prompts create additional cognitive distraction as the user has to specify the desired POI either more clearly or by coming up with synonyms for the POI.

R12: The system should be usable without physical distraction
Status: Partially achieved
Evaluation: The user does not have to physically interact with the system and the navigation device when it is running. Starting up the system still requires the user to select the navigation tool from the device. This can be done before starting the driving process, but in cases where a new desired POI comes up, the application still requires physical attention to start it.

R13: The system should be usable with minimal visual distraction
Status: Partially achieved
Evaluation: While interacting with the voice-interface was possible without making visual contact, interaction with the navigation interface still requires visual focus from the user. This, however, is an issue that persists with any tools and devices that are introduced to the vehicular environment.

R14: The system should not introduce situations that interfere with the driving task
Status: Partially achieved
Evaluation: The system can mostly be used without it interfering with the driving task. However, in situations such as the ones in R11, R12, and R13, the driver is required to expend more focus on the tool than normally.

R15: The system should withstand temporary service breakdowns
Status: Achieved
Evaluation: As of now, the system requires connectivity with the external services only upon invocation. Temporary disconnects and service breakdowns do not interfere with successful invocation requests.

R16: The tool should understand English locations
Status: Achieved
Evaluation: The tool is capable of understanding the majority of American and British locations.

R17: The tool should understand Finnish locations
Status: Partially achieved

Evaluation: Unlike American, British and German locations, Amazon does not provide us a list of Finnish locations, so the task of making the skill capable of recognizing Finnish locations is left to the developer of the skill. A small list of Finnish locations was made for this research, to successfully ensure its functionality with said locations.

Table 5. Requirement statuses on a table.

R1 **Achieved**	**R2** **Partially Achieved**	**R3** **Achieved**	**R4** **Partially Achieved**	**R5** **Achieved**	**R6** **Achieved**	**R7** **Achieved**	**R8** **Partially Achieved**	**R9** **Not Achieved**
R10 **Achieved**	**R11** **Partially Achieved**	**R12** **Partially Achieved**	**R13** **Partially Achieved**	**R14** **Partially Achieved**	**R15** **Achieved**	**R16** **Achieved**	**R17** **Partially Achieved**	

As seen in Table 5, the majority of the requirements were at least partially fulfilled. Based on the general nature of the requirements and the evaluation process itself, the results indicate that utilizing a system with similar qualities in the automotive environment would provide positive results. However, these results would be largely dependent on the resources invested in building the system to expand the vocabulary of the AVS. The designed artefact would not fulfill the safety-critical requirements of the automotive environment as it is, but the evaluation results prove that expending resources towards this concept would lead to a positive outcome when utilizing modern technologies, such as using cloud-computing with ASR and NLP processes.

6. Discussion

The objective of this research was to design and develop a voice-based interface to be used in the vehicular environment, to investigate the benefits of the hands-free, voice-based operation and to improve the safety of driving for everyone involved. The research problem was approached through the utilization of DSR methodology. The design- and development process was divided into cycles that were iterated to produce a system capable to fulfill the desired functionality. The designed artefact was evaluated against the identified requirements, which resulted in the artefact at least partly fulfilling most of the set requirements. The requirements that were left unachieved were due to limitations based on resource- and time capabilities and the strictness of the environmental requirements. The limitations are discussed in the next section, along with methods of solving the limitations.

This DSR method was conducted to answer the research questions described in chapter 2. The first research question**, how a voice-controlled interface can be designed for the in-vehicle system?** was answered by the design work that was conducted during this research. The voice-operated infotainment system design started by first choosing the Amazon Alexa as the user interface for providing the voice-based control functionality. This decision was based on the possibility to customize the Alexa service to provide us the required functionality, as mentioned in chapter 3.1. The development process utilized the Amazon Alexa and the associated Amazon services with the intent to extend the usability of the customized Alexa skill. The android-based navigation interface and the APIs from Google were utilized to simulate an infotainment system. The development process followed the steps described in the development chapter 4: Gather requirements, design system structure, setup Alexa by constructing the custom skill base along with the invocations, intents and sample utterances. Next steps were to generate a lambda function where the functionality of the custom skill is defined, set up an AWS IoT service to transmit the POI data from lambda to the navigation interface, implement the navigation interface and its connectivity with the lambda function and the google maps API and provide the user with a visual representation of the route from its location to the desired POI.

The answer to the second research question, **what are the benefits and challenges of a voice-controlled interface in the in-vehicle context?** was provided by the collaborative process of examining the literature and evaluating the results of the conducted DSR. The main benefits of the designed voice interface navigation system identified from the literature and the DSR process were that the task of physically operating in-vehicle infotainment systems can be replaced with the voice-based operation. The usage of Amazon Alexa as the voice-based interface for navigation provides similar improvements to operating secondary tasks while being busy with the driving task, similar to how the reviewers experienced in the paper by Gao et al., (2018).

The main challenges of the designed system were related to both the technical properties of the Alexa and the nature of speech-based communication: To enhance robustness in a speech-recognition based control system, the underlying service that is handling the ASR and NLP processes need to be trained to the context of the deployed target environment. In the case of this study, this means that the developed voice-control system needs to be trained to function correctly in the in-vehicle environment. Training a foreign language to the service would require a separate training process to be carried out.

As for the nature of speech-based communication, the user of the system has to expend cognitive resources to initiate a conversation with the developed system: The custom skill was invoked by speaking out the phrase “Alexa, navigate me to POI”. This kind of conversation is not considered natural, thus requiring the user of the system to learn how to invoke the custom skill to achieve the desired functionality.

The development of an in-vehicle dialog system has benefits for the driver both in the present and the future: It has the capability of reducing driver distraction from the services currently available through connectivity. In the future, with vehicles being fully autonomous, the in-vehicle dialog system plays the role of assisting the passengers occupying the automated vehicle in various tasks (Weng et al., 2016). This thesis provides insight into the topic of voice-controlled in-vehicle infotainment systems by designing a system that utilizes the currently available technologies that directly improve the ASR- and NLP processes in voice-based interaction. This work aims to fill the research gap in utilizing currently available commercial products in new environments. Cloud computing can be seen as a direct step forward in the area of improving both ASR and NLP processes.

While speech-driven communication has been reported to provide improvements in the vehicular environment by allowing the driver to stay focused on operating the vehicle while operating secondary tasks by voice, such tasks do still cause driver distraction (Hansen et al., 2008). Similar issues were observed when evaluating the designed system. The developed system could enable the task of operating the navigation system solely by voice, but these tasks still induce distraction to the driver in both auditory- and cognitive areas of distraction (Hansen et al., 2008)

6.1 Limitations:

APPSTACLE project was one of the main motivators for this study, as the research topic was selected based on both the interests of the author and the needs of the project itself. The research was conducted throughout the project. The results gained from the design and development process provided insight into the use-cases fulfilled by our project team in Oulu. Another source of motivation for the study was the selection of Amazon Alexa as the tool that provides the voice-controlling functionality for the designed system. The ease of integrating the service into other services relevant to the study was a deciding factor behind utilizing this set of Amazon-based services. A similar study by utilizing other services and tools available would introduce variability and fill the gaps in the knowledge base on this subject. In the event of someone willing to expand this study further, the source code for both the lambda function and the navigation tool is in the following address: https://github.com/jmouru/voice-interface.

Some factors highly relevant to the vehicular context were not taken into consideration as heavily as they should have been when developing the requirements. One such factor is the sensitivity of the processing time in the integrated electrical devices: As the vehicle is constantly moving, network services cannot be constantly guaranteed on the road. This means that relying on external processing is not very ideal (Zheng et al, 2017a). Such requirements were all the efficiency-based requirements that would define a time window in which the artefact should be able to function. Now the requirements were fulfilled as long as the artefact provided the required functionality. This is also a viable option for further extending the research in this topic area. While the designed system could provide its desired functionality without introducing additional physical or visual distractions, auditory and cognitive distractions are distraction types that seem to persist in the vehicular environment no matter what kind of improvements are being directed at them. Solutions to these distraction challenges can surely lessen the amount of distraction so that they introduce less of it than the currently existing sources, such as listening to the radio, glancing at the speedometer or just conversing with another passenger.

In the evaluation chapter, requirements such as R4 introduced new-found challenges for the designed artefact: During the verification of the desired POIs, some POIs defined by the user would have multiple possible locations that correspond to the desired POI. During the design- and development process, such situations could not be handled by the application, which in turn would simply return the first candidate from the list of possible locations. Solving this issue could be done in future studies: Utilizing the DSR methodology to design a decision management tool that holds the possible values and selects one with the support of estimation calculations or prompting the user to choose one.

To scale the design- and development work to fit into a Master's thesis, a notable amount of keywords that indicate the requirement levels of each requirement were modified for leniency. Using the keyword "should" instead of "must" enables the possibility to describe the process more thoroughly if the requirement cannot be fulfilled. If the designed system were to be directly implemented into an actual in-vehicle environment, the specified requirements would be worded more strictly. During the development process of this research, Amazon published a new SDK that is specifically made for integrating Alexa into the in-vehicle environment. This study does not utilize this SDK but acts as its own system to prove the feasibility of this experiment with the tools and components provided (Amazon, 2018).

Testing the designed tool in a real-world environment introduces the evaluation process external interference from things such as internal- and external noises, unstable connectivity and evaluation settings too difficult to limit properly. If the designed tool was to be evaluated in such environments, a baseline for functionality should be assessed beforehand. As the scope of this study was to assess how voice-operated interface can be integrated into the automotive environment, the results of this study can be now specified as that baseline for further evaluation in the real automotive environment. Any non-functional requirements that measure how the system functions in situations other than the dedicated laboratory setting were not considered to be suitable for evaluation. If this design was evaluated to be suitable for larger-scale deployment to the real automotive domain, questions related to scalability, reliability, and rigorousness of the design become more relevant.

Due to time- and resource limitations, the challenge of overcoming robustness in terms of noise management was not evaluated during the study. The concept of evaluating this specific requirement area would require redirecting the scope of this study, which in this case was providing the voice-interaction into the vehicular environment. To evaluate the robustness aspect of the design more broadly and systematically, the utilization of in-car noise-simulating tools would be advised. In the study by Ding et al., (2008), the acoustics model was trained by adding pre-recorded car noises on the samples used to train the model. Similarly, the study by Khan et al. (2017) focused on improving voice recognition in the vehicle by implementing a software-based solution that utilized signal processing principles while providing students with engineering education. The study introduced different forms of voices to be eliminated with the use of filters. The study resulted in a model for an in-car control- and security system, that could be operated by speech recognition. While one could argue that the research topic of voice-controlled in-vehicle infotainment systems is a subject that has been implemented to vehicles by car manufacturers (Khan et al., 2017).

The APPSTACLE project itself had the primary task of providing an open-source solution for the issue of automotive software-intensive systems being developed separately in silos of each car manufacturer or by original equipment manufacturers (OEM). Even though the source code for the resulting artefact developed during this study is freely available, a requirement ensuring that the result is made open source was not defined. Even though Amazon provides open-source support, this decision was justified by the description of the use-case in the project: Demonstrating the utilization of third-party applications in the car-to-cloud platform.

7. Conclusion

The purpose of this thesis was to find out if tools and services utilizing modern technologies can be introduced to the infotainment system and its associated tasks within the in-vehicle environment. The research was conducted by using DSR methodology to test if the designed system can provide the estimated improvements as described in chapter 4.

The system was built by integrating the Amazon Alexa and its services with an android-based device simulating the infotainment system of a vehicle. The implementation of the system was based on requirements from both literature and the project environment this study was conducted in. Designing the system was conducted by following the steps of defining the system structure, defining the workflows and selecting the components and services suitable for the research. Finally, the software running the desired functionality was developed and the built system was evaluated to meet the requirements and fulfill the research task. Following these steps as described during this DSR, the first research question can be answered. The resulting system was able to prove that voice-based communication can be utilized to improve the handling of secondary tasks in the in-vehicle environment. These improvements and the challenges introduced by the demanding environment of the vehicle were reported in chapter 6, thus answering the research question 2.

The results of this research provide support for the process of introducing speech-driven communication capabilities to the vehicular system. This can be seen useful from both industrial- and academic viewpoints: The domain of speech-driven communication is still young and open for improvements and innovations. As the currently available commercial products already provide the basic means to integrate these two areas of voice-interaction and the in-vehicle environment, one could only imagine how well a voice-driven system specifically tailored for this purpose would do. As for the knowledge base, utilizing the distraction- and secondary task improvements that the voice-based navigation could provide could be a topic of research in areas other than the in-vehicle environment.

As for the knowledge base, the improvements in both lessening distraction and enabling secondary tasks could be an interesting research area to extend into. Within the vehicular environment, speech-based interaction is still in need of solutions to the challenges identified from prior literature and the results of the DSR conducted in this study. If the utilization of Amazon Alexa can help solve these challenges, this study verifies the feasibility of said utilization.

While the challenges of the in-vehicle environment were proven difficult to solve, the integration of tools and devices are expected to go through the same process to ensure safe and secure integration to the safety-critical environment that is the car. Further studies could extend the usability of the implemented voice-controlled interface to other services located in the vehicle. Additionally, constructing a setting that simulates the vehicular environment helps the evaluation process in areas such as the level of distraction and the robustness of the artefact.

8. References

Amazon. (2010a). *Alexa Voice Service.* Retrieved November 27, 2019 from https://developer.amazon.com/en-US/alexa/alexa-voice-service.

Amazon. (2010b). *What is Alexa?* Retrieved November 27, 2019 from https://developer.amazon.com/en-US/alexa.

Amazon. (2011). *About AWS*. Retrieved November 14, 2019 from https://aws.amazon.com/about-aws/.

Amazon. (2018). *Amazon Alexa Auto Software Development Kit*. Retrieved November 27, 2019 from https://developer.amazon.com/en-US/alexa/alexa-auto/sdk.

Amazon. (n.d.). *What are Alexa Skills?* Retrieved October 30, 2019 from https://www.amazon.com/gp/help/customer/display.html?nodeId=GG3RZLAA3RH83JAA.

ITEA3. (2016a). *APPSTACLE - open standard APplication Platform for carS and TrAnsportation vehiCLEs*. Retrieved December 23, 2019 from https://itea3.org/project/appstacle.html

ITEA3. (2016b) *Project profile*. Retrieved December 23, 2019 from https://itea3.org/project/result/download/7028/APPSTACLE%20Project%20leaflet.pdf

Barros, R. J., & Boucher, C. (1996). ITS navigation software. *Proceedings of Position, Location and Navigation Symposium - PLANS '96*, 422–425.

Chen, D., Johansson, R., Lönn, H., Papadopoulos, Y., Sandberg, A., Törner, F., & Törngren, M. (2008). Modelling support for design of safety-critical automotive embedded systems. In M. D. Harrison & M.-A. Sujan (eds), *Computer Safety, Reliability, and Security* (ss. 72–85). Springer.

Chung, H., Park, J., & Lee, S. (2017). Digital forensic approaches for Amazon Alexa ecosystem. *Digital Investigation*, 22, S15-S25.

Chung, H., Park, J. G., Lee, Y. K., & Chung, I. (2008). Fast speech recognition to access a very large list of items on embedded devices. *IEEE Transactions on Consumer Electronics*, 54(2), 803–807.

Ding, P., He, L., Yan, X., Zhao, R., & Hao, J. (2008). Robust mandarin speech recognition in car environments for embedded navigation system. *IEEE Transactions on Consumer Electronics*, 54(2), 584–590.

Dix, A. (2004). *Human-computer interaction (3rd ed.).* Pearson / Prentice Hall.

Eclipse. (2019). *About*. Retrieved December 23, 2019 from https://www.eclipse.org/kuksa/

Gao, Y., Pan, Z., Wang, H., & Chen, G. (2018). Alexa, my love: Analyzing reviews of amazon echo. *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing,* *Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI),* 372–380.

Hacioglu, K., Pradhan, S., Ward, W., Martin, J. H., & Jurafsky, D. (2004). Semantic role labeling by tagging syntactic chunks. *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, 110–113.

Sambells, J. (2010). *Decoding polylines from google maps direction API with java*. Retrieved October 24, 2019 from https://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java

Kinsella, B. (2019). *Google Assistant Actions Total 4,253 in January 2019, Up 2.5x in Past Year but 7.5% the Total Number Alexa Skills in U.S.* Retrieved November 27, 2019 from https://voicebot.ai/2019/02/15/google-assistant-actions-total-4253-in-january-2019-up-2-5x-in-past-year-but-7-5-the-total-number-alexa-skills-in-u-s/

Hansen, J. H. L., Kim, W., & Angkititrakul, P. (2008). Advances in human-machine systems for in-vehicle environments. *2008 Hands-Free Speech Communication and Microphone Arrays*, 128–131.

Hataoka, N., Manabu Araki, Takashi Matsuda, Masayuki Takahashi, Ryoichi Ohtaki, & Obuchi, Y. (2008). Evaluation of interface and in-car speech—Many undesirable utterances and sever noisy speech on car navigation application -. *2008 IEEE 10th Workshop on Multimedia Signal Processing*, 956–959.

Herbig, T., Gerl, F., & Minker, W. (2010a). Fast adaptation of speech and speaker characteristics for enhanced speech recognition in adverse intelligent environments. *2010 Sixth International Conference on Intelligent Environments*, 100–105.

Herbig, T., Gerl, F., & Minker, W. (2010b). Simultaneous speech recognition and speaker identification. *2010 IEEE Spoken Language Technology Workshop*, 218–222.

Hevner, A., & Chatterjee, S. (2010). Introduction to design science research. In *Design Research in Information Systems* (pp. 1-8). Springer, Boston, MA.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems,* 19(2), 4.

Hevner, March, Park, & Ram. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75.

Hong, L., Rosca, J., & Balan, R. (2004). Independent component analysis based single channel speech enhancement. *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No.03EX795)*, 522–525.

Hunt, M. J. (2002). An examination of three classes of ASR dialogue systems: PC-based dictation, in-car systems and automated directory assistance. *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2001. ASRU '01., 455–461.

Hüsson, D., & Holland, A. (2019). Intelligent personal assistants in business processes: Evaluation of a Prototype (V-ip-a). *Humanizing Technology for a Sustainable Society*, 1133–1145.

Iivari, Juhani (2007) "A Paradigmatic Analysis of Information Systems As a Design Science," *Scandinavian Journal of Information Systems*: Vol. 19 : Iss. 2 , Article 5.

Jeong, M., Kim, B., & Lee, G. G. (2003). Semantic-oriented error correction for spoken query processing. *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*, 156–161.

Kadambe, S. (2002). In-vehicle acoustic chamber ARMA modeling and classification. *Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop*, 56–61.

Khan, S., Akmal, H., Ali, I., & Naeem, N. (2017). Efficient and unique learning of in-car voice control for engineering education. *2017 International Multi-topic Conference (INMIC)*, 1–6.

Kirson, A. M. (1995). A compact driver interface for navigation and route guidance. *Pacific Rim TransTech Conference. 1995 Vehicle Navigation and Information Systems Conference Proceedings. 6th International VNIS. A Ride into the Future*, 61–66.

Kusuma, J., Rashmi, R., Sandhya, K., Tejashwini, S., & Vidyashree, V. Alexa based Weather Station. *International Journal of Research in Engineering, Science and Management*, 2(5), 406-408.

Moniri, M. M., Feld, M., & Müller, C. (2012). Personalized in-vehicle information systems: Building an application infrastructure for smart cars in smart spaces. *2012 Eighth International Conference on Intelligent Environments* (pp. 379-382). IEEE.

Patel, J., Ball, D. J., & Jones, H. (2008). Factors influencing subjective ranking of driver distractions. *Accident Analysis & Prevention*, 40(1), 392–395.

Qian, Y., Liu, J., & Johnson, M. T. (2009). Efficient embedded speech recognition for very large vocabulary Mandarin car-navigation systems. *IEEE Transactions on Consumer Electronics*, 55(3), 1496–1500.

Sandnes, F. E., Huang, Y.-P., & Huang, Y.-M. (2008). An eyes-free in-car user interface interaction style based on visual and textual mnemonics, chording and speech. 2008 *International Conference on Multimedia and Ubiquitous Engineering (mue 2008)*, 342–347.

Shozakai, M., Nakamura, S., & Shikano, K. (1998). Robust speech recognition in car environments. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, 1, 269–272.

Smith Jr, J. F. (2000). Eyes on the road, hands on the wheel: The OnStar approach to in-vehicle communication and safety. *Vital Speeches of the Day*, 67(3), 66.

Suchman, L. A. (1990). What is human-machine interaction. *Cognition, computing, and cooperation*, 25-55.

Tam, Y. C., Lei, Y., Zheng, J., & Wang, W. (2014). ASR error detection using recurrent neural network language model and complementary ASR. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2312-2316). IEEE.

Weng, F., Angkititrakul, P., Shriberg, E. E., Heck, L., Peters, S., & Hansen, J. H. L. (2016). *Conversational in-vehicle dialog systems: The past, present, and future. IEEE Signal Processing Magazine*, 33(6), 49–60.

Zheng, Y., Liu, Y., & Hansen, J. H. L. (2017a). Navigation-orientated natural spoken language understanding for intelligent vehicle dialogue. *2017 IEEE Intelligent Vehicles Symposium (IV)*, 559–564.

Zheng, Y., Liu, Y., & Hansen, J. H. L. (2017b). Intent detection and semantic parsing for navigation dialogue language processing. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6.

Appendix A: JSON files used in data exchange

```
"request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.1234",
    "timestamp": "2019-11-27T21:06:28Z",
    "locale": "en-US",
    "intent": {
      "name": "MapIntent",
      "slots": {
        "addressNumber": {
          "name": "addressNumber",
          "value": null,
          "confirmationStatus": "NONE",
          "encryptedValue": null
        },
        "knownLocation": {
          "name": "knownLocation",
          "value": null,
          "confirmationStatus": "NONE",
          "encryptedValue": null
        },
        "Location": {
          "name": "Location",
          "value": "Oulun yliopisto",
          "confirmationStatus": "NONE",
          "encryptedValue": null
        }
      }
    }
}
```

Figure A1. Information stored in the Alexa to Lambda transmission.

```
{
        results: [
        {
                address_components: [
                {
                        long_name: "Oulu",
                        short_name: "Oulu",
                        types: [
                                "locality",
                                "political"
                        ]
                },
                {
                        long_name: "Suomi",
                        short_name: "FI",
                        types: [
                                "country",
                                "political"
                        ]
                },
                {
                        long_name: "90014",
                        short_name: "90014",
                        types: [
                        "postal_code"
                        ]
                }
                ],
                formatted_address: "Pentti Kaiteran katu 1, 90014
                Oulu, Suomi",
                geometry: {
                        location: {
                                lat: 65.0593177,
                                lng: 25.4662935
                        },
                location_type: "GEOMETRIC_CENTER",
                viewport: {
                        northeast: {
                                lat: 65.0606666802915,
                                lng: 25.4676424802915
                        },
                        southwest: {
                                lat: 65.05796871970848,
                                lng: 25.4649445197085
                        }
                }
                },
                place_id: "ChIJs68F7g8tgEYRoeuhCGs_lug",
                plus_code: {
                        compound_code: "3F58+PG Linnanmaa, Oulu,
                        Suomi",
                        global_code: "9GQ73F58+PG"
                },
                types: [
                        "establishment",
                        "point_of_interest",
                        "university"
                ]
                }
        ],
        status: "OK"
}
```

Figure A2. The contents retrieved on a successful geocoding API request.

```
{
        results: [ ],
        status: "ZERO_RESULTS"
}
```

Figure A3. The contents retrieved on an unsuccessful geocoding API request.

```
{
  "version": "1.0",
  "sessionAttributes": {},
  "response": {
    "outputSpeech": {
      "type": "PlainText",
      "text": "Giving directions to: Pentti Kaiteran katu 1, 90014
Oulu, Finland"
    },
    "card": {
      "type": "Simple",
      "title": "Location response",
      "content": "Giving directions to: Pentti Kaiteran katu 1, 90014
Oulu, Finland, 65.0593177,25.4662935"
    },
    "reprompt": {
      "outputSpeech": {
        "type": "PlainText",
        "text": "Please provide a location"
      }
    },
    "shouldEndSession": false
  }
}
```

Figure A4. Speech response from Lambda function back to the Echo speaker.

```
{
    "geocoded_waypoints": [
       {
          "geocoder_status": "OK",
          "place_id": "ChIJqZ5loqUygEYR3mk0gq4HOyM",
          "types": [
             "establishment",
             "point_of_interest",
             "tourist_attraction"
          ]
       },
       {
          "geocoder_status": "OK",
          "place_id": "ChIJs68F7g8tgEYRoeuhCGs_lug",
          "types": [
             "establishment",
             "point_of_interest",
             "university"
          ]
       }
    ],
    "routes": [
       {
          "bounds": {
             "northeast": {
                "lat": 65.0557963,
                "lng": 25.4789578
             },
             "southwest": {
                "lat": 65.0131899,
                "lng": 25.4650866
             }
          },
          "copyrights": "Map data ©2019",
          "legs": [
             {
                "distance": {
                   "text": "5.6 km",
                   "value": 5630
                },
                "duration": {
                   "text": "10 mins",
                   "value": 604
                },
                "end_address": "Pentti Kaiteran katu 1, 90570 Oulu,
Finland",
                "end_location": {
                   "lat": 65.0556576,
                   "lng": 25.4664207
                },
                "start_address": "Kauppatori, 90100 Oulu, Finland",
                "start_location": {
                   "lat": 65.0131899,
                   "lng": 25.4650866
                },
                "steps": [
                   {
                      "distance": {
                         "text": "0.2 km",
                         "value": 216
                      },
                      "duration": {
                         "text": "1 min",
                         "value": 73
                      },
                      "end_location": {
                         "lat": 65.0148396,
                         "lng": 25.4675274
                      },
                      "html_instructions": "Head <b>northeast</b> on
<b>Rantakatu</b> toward
<b>Packhusgatan</b>/<wbr/><b>Pakkahuoneenkatu</b>",
                      "polyline": {
                         "points": "m{xkKyslzCwAcC{@yAGGMUYa@iB_D[i@"
                      },
                      "start_location": {
                         "lat": 65.0131899,
                         "lng": 25.4650866
                      },
                      "travel_mode": "DRIVING"
                   }
                ],
                "traffic_speed_entry": [],
                "via_waypoint": []
             }
          ],
          "overview_polyline": {
             "points":
"m{xkKyslzCiD{FcCaE[i@^sBb@iCwA}B_A{Ao@eAq@wAUWG?KFq@`BMPgC|CkBvBaAx@{
@\\wBz@m@Ps@FgACuFc@sCUgAM}@Kc@A{@Cw@Ga@Ga@[iAeAcCcBqIyGy@a@YMe@GuAZkC
dCiBnCaBjCeAzAcAfAcBfAqA^_AJO?cFMqPoAeBQaGa@eBQgB_@eB{@i@e@uB{BaAyA}@g
BkBsFwBoIWs@m@gBYy@a@}@sB_EcBqCMQMQ_CeCUSwA{AuAkAiAi@yAg@G?IDoAUw@MmAI
cAAuCXQDYDyAVGCmBZkDt@yFhBiD~@KNoAb@a@LqD~@]JMIuBh@kDb@_BB_BUKCAb@GzAE
dAGzCCnCBrE@h@@~@BvBLvFNxOD^B|FW@"
          },
          "summary": "Route 8156 and Alakyläntie",
          "warnings": [],
          "waypoint_order": []
       }
    ],
    "status": "OK"
 }
```

Figure A5. The contents of a successful Directions API request.

Appendix B: System diagrams

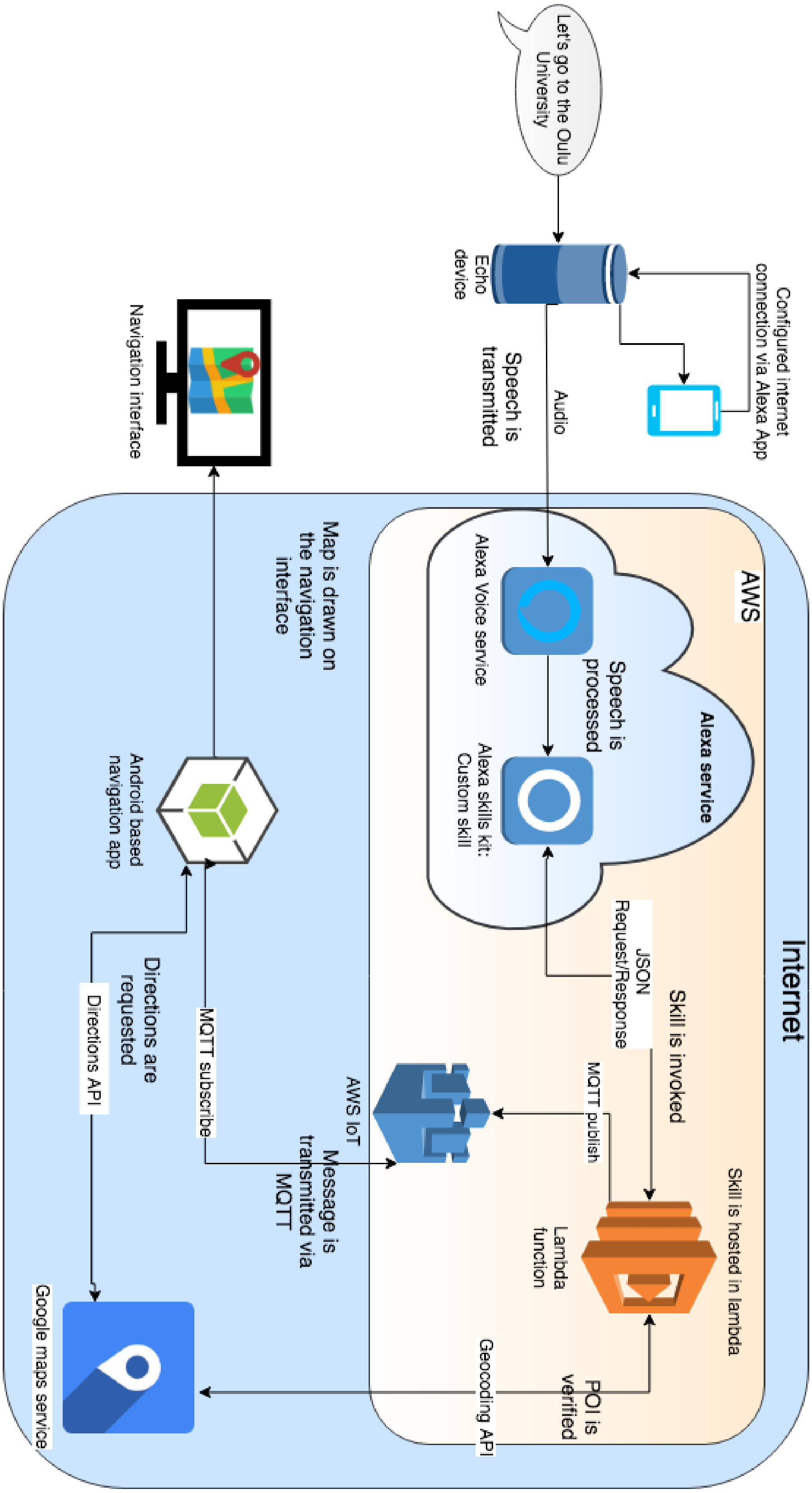


Figure B1. System Architecture diagram

Appendix C: System source code

```
function onIntent(intentRequest, session, callback) {

    var intent = intentRequest.intent
    var intentName = intentRequest.intent.name;
    if (intentName == "MapIntent") {
        handleMapIntentRequest(intent,session,callback);
    }
```

Figure C1. Directing the intent to its specified handler.

```
if (intent.slots.knownLocation.value === null ||
intent.slots.knownLocation.value === undefined) {
        if (intent.slots.Location.value === null ||
intent.slots.Location.value === undefined) {
            speechOutput = "Please provide a location";
            callback(session.attributes, buildSpeechletResponse(title,
speechOutput, rText,speechOutput, false))
        } else {
            command = intent.slots.Location.value.toLowerCase();
            //If slots.Location has value, addressumber could have
value
            //addressNumber slot, which we check and try to apply if
possible
            if(intent.slots.addressNumber.value !== undefined ||
intent.slots.addressNumber.value !== null) {
                command = command + " "+
intent.slots.addressNumber.value;
            }
        }
    } else {
             command = intent.slots.knownLocation.value.toLowerCase();
    }
var initPromise = getConfirmedLocation(command);
```

Figure C2. Handling the MapIntent -request and retrieving its associated values.

```
async function getConfirmedLocation(location) {
    return new Promise((resolve,reject) => {
        var escapedLocation = require('querystring').escape(location);
        const queryString = ("/maps/api/geocode/json?address=" +
escapedLocation + "&key={API_KEY}");
        const options = {
            hostname: "maps.googleapis.com",
            path: queryString,
            port: 443,
            method: 'GET',
            json: true
        };
        https.get(options, function (res) {
            var json = '';
            res.on('data', function (chunk) {
                json += chunk;
            });
            res.on('end', function () {
                if (res.statusCode === 200) {
                    try {
                        var data = JSON.parse(json);
                        console.log(data);
                        // data is available here:
                        resolve(data);
                    } catch (e) {
                        console.log('Error parsing JSON!');
                    }
                } else {
                    console.log('Status:', res.statusCode);
                }
            });
        }).on('error', function (err) {
            console.log('Error:', err);
        });

   });
}
```

Figure C3. Location confirmation handled by calling the Google geocoding API.

```
var initPromise = getConfirmedLocation(command);
    initPromise.then(function(result) {
    var obj = result;
    var coords;
    if (obj !== null) {
        if (obj.status == "ZERO_RESULTS") {
            speechOutput = "I couldnt find a location for " + command;
            callback(session.attributes, buildSpeechletResponse(title,
speechOutput, rText,speechOutput, false));
        }
        else {
            speechOutput = "Giving directions to: " +
obj.results[0].formatted_address;
            coords = (obj.results[0].geometry.location.lat + "," +
obj.results[0].geometry.location.lng);
            callback(session.attributes, buildSpeechletResponse(title,
speechOutput,rText,(speechOutput + ", " + coords), false));
            sendLocation(coords);
        }
    }
```

Figure C4. Handling the proceeding based on the contents of the JSON file.

```
var iotdata =  new AWS.IotData({endpoint:process.env.ENDPOINT});
function sendLocation(coords) {
    var params = {
        topic: "topic/loc1",
        payload: "GOTO: "+ coords,
        qos: 0
        };
        return iotdata.publish(params, function(err, data){
        if(err){
            console.log("Error occured : ",err);
        }
```

Figure C5. Setting up the message transmission function.

```
private static final String IOT_ENDPOINT = BuildConfig.ENDPOINT
mqttManager = new AWSIotMqttManager({client-id}, IOT_ENDPOINT);

public void subscribe(final View view) {
final Intent mapIntent = new Intent(this, MapsActivity.class);
final String topic = txtSubscribe.getText().toString();
try {
        mqttManager.subscribeToTopic(topic, AWSIotMqttQos.QOS0,
        new AWSIotMqttNewMessageCallback() {
                @Override
                public void onMessageArrived(final String topic, final
byte[] data) {
                runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                        try {
                        String message = new String(data, "UTF-8");
                        tvLastMessage.setText(message);
                        if (message.contains("GOTO:")) {
                        String retrievedLocation =
message.substring(message.indexOf("GOTO:") + 4, message.length());
                        if(!message.equals(currentLocation)){
mapIntent.putExtra("LOCATION_DATA", retrievedLocation);
                                currentLocation = message;
                                startActivity(mapIntent);
```

Figure C6. Subscription to the MQTT broker and handling the received messages.

```
location = getLocation();
lat = location.getLatitude();
lng = location.getLongitude();
mMap = googleMap;
dataDownload.execute();
LatLng curLoc = new LatLng(lat, lng);
if (mMarkerPoints.size() > 1){
    mMarkerPoints.clear();
    mMap.clear();
}
mMarkerPoints.add(curLoc);
mMap.addMarker(new MarkerOptions().position(curLoc).title("Current
location"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(curLoc));
mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(curLoc,12.0f));
```

Figure C7. Map generation in *MapsActivity* and assigning the current location.

```
try {
        response =
getDirections("https://maps.googleapis.com/maps/api/directions/json?or
igin="+ lat + "," + lng + "&destination="+location+"&key="+APIKEY);
        return new String[]{response};
} catch (Exception e) {
return new String[]{"error"};

public String getDirections(String requestURL) throws IOException {
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try {
            URL url = new URL(requestURL);
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setReadTimeout(3000);
            urlConnection.setConnectTimeout(3000);
            urlConnection.setRequestMethod("GET"); //http GET
            urlConnection.setUseCaches(false);
            urlConnection.setAllowUserInteraction(false);
            urlConnection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
            int responseCode = urlConnection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                // Reading data from url
                iStream = urlConnection.getInputStream();
                BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));
                StringBuffer sb = new StringBuffer();
                String line = "";
                while ((line = br.readLine()) != null) {
                    sb.append(line);
                }
            data = sb.toString();
            br.close();
            }else {
                data = "";
                Log.d("Response", "Responsecode was " + responseCode);
            }
        } catch (Exception e) {
            Log.d("Exception", e.toString());
        } finally {
            iStream.close();
            urlConnection.disconnect();
        }
        return data;
    }
```

Figure C8. Requesting the routing information with the *getDirections* -function

```
private List<LatLng> decodePoly(String encoded) {

            List<LatLng> poly = new ArrayList<>();
            int index = 0, len = encoded.length();
            int lat = 0, lng = 0;
            while (index < len) {
                int b, shift = 0, result = 0;
                do {
                    b = encoded.charAt(index++) - 63;
                    result |= (b & 0x1f) << shift;
                    shift += 5;
                } while (b >= 0x20);
                int dlat = ((result & 1) != 0 ? ~(result >> 1) :
(result >> 1));
                lat += dlat;
                shift = 0;
                result = 0;
                do {
                    b = encoded.charAt(index++) - 63;
                    result |= (b & 0x1f) << shift;
                    shift += 5;
                } while (b >= 0x20);
                int dlng = ((result & 1) != 0 ? ~(result >> 1) :
(result >> 1));
                lng += dlng;
                LatLng p = new LatLng((((double) lat / 1E5)),
                        (((double) lng / 1E5)));
                poly.add(p);
                }
        return poly;
```

Figure C9. The function that decodes the encoded polylines (Sambells, 2010).