An approach of Bluetooth performance evaluation in an Android automotive setting

ADAM BARVESTEN & OLA OLDE
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

An approach of Bluetooth performance evaluation in an Android automotive setting

Adam Barvesten
ad2626ba-s@student.lu.se
Ola Olde
elt13oo1@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Maria Kihl

Examiner: Christian Nyberg

June 15, 2022

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

In recent years cars are becoming more and more connected and it has become commonplace to connect one's phone to the car interface, allowing streaming of music, making calls and sending messages. The primary technology used for connection is Bluetooth, a short-ranged wireless communication technology that has been around since 1999 and improved and expanded upon ever since. It is of vital importance to evaluate the performance of Bluetooth in an in-car system to make sure the connection establishment and communication flow is without hindrance and to assess the Bluetooth functionality whenever there is a new software update within the infotainment system. In this thesis we investigate ways to develop a test for Bluetooth performance in an Android in-car infotainment system.

To simulate a scenario of a person entering their car, a test is designed to measure relevant Bluetooth profiles, the time it takes for the phone book to get downloaded to the in-car system, and the total time for the entire auto connection process to finish. The software of the test is developed in the source code of the infotainment unit to allow for evaluation in a lab environment as well as in cars. The test is run on two different phone models, to show how an comparison could be made with the test method, repeatedly in a lab to produce 10000 measurements. The results given are consistent measurements in a lab environment and potential for implementation in the real world with further improvements. The results from the test can be used to evaluate differences in connection times for various cars and phone models, or to continuously monitor the connection performance in a single setup.

Popular Science Summary

The car industry is in a constant state of improvement. In recent times there has been a surge to electrify the whole industry and with it new vehicles become more technologically advanced. The mobile phone is closely linked to this scenario, as its connection to the car allow users to navigate roads, stream music and make calls and messages from the car interface. The connection between car and phone is typically facilitated via Bluetooth communication. A reliable and quick connection is vital for the user experience of any car owner. As with any product development, testing is the first step in ensuring the result is satisfactory. This thesis takes a look into several ways of developing a test to enable performance evaluation of the Bluetooth connection in a car-to-phone system. The test was chosen to focus on relevance to the user experience and thus measuring the time it takes for establishing a connection.

There are many different parameters and methods one could use when developing a test for Bluetooth performance, and this thesis look into a number of them and assess their strengths and weaknesses. The final approach chosen for the design of the test developed was to measure connection times in the upper application layer.

More specifically, the test measures the automatic connection of phones, previously paired with the car's infotainment unit. During this automatic connection process several Bluetooth profiles are connected, which allow functions such as streaming music, making calls and sending messages. The test measures the connection time for each profile separately and stores it locally on the infotainment unit. The phone book, containing contacts and call history, is downloaded to the infotainment unit for each auto-connection process, and was thus chosen to be measured as well. A total time from the start of connection until the whole process is finished is also measured. The test software is integrated into the main source code of the infotainment unit, which allows test to be run both in labs as well as in cars.

The measurements in this thesis were collected in a lab environment, using an external infotainment unit, with the intention of future testing being easily integrated in cars out in the real world later on. Two different test phones were used to show how a comparison with the test could look like, each with phone

books containing 100 contacts. Connection time for the Bluetooth profiles were measured 10000 times and the phone book download times were measured 4000 times. Due to difficulties in automating a test for the total time, it was measured manually 20 times for each phone as a way of proving the feasibility of the method.

All in all, the test works reliably in a lab environment and should in theory work in cars, although this is yet to be attempted. With slight improvements it can be used to collect data from customer cars and development vehicles as well. The data collected from the test can be used for statistical analysis of the overall Bluetooth connection performance which can be useful for testing the performance of different phone models and monitoring the potential performance change from software updates.

Acknowledgements

We would like to express our gratitude to the following people:

- Our main supervisor at Volvo Car Corporation for giving us the opportunity and tools to complete our thesis:
  - **Karin Larsson**, *Team Manager, Infotainment department, Volvo Cars Corporation, Lund*
- The examiner and supervisor from Lund University, faculty of engineering (LTH) for the help with our thesis:
  - **Christian Nyberg**, *Associate Professor, Electrical and Information Technology, Faculty of Engineering, Lund University*
  - **Maria Kihl**, *Professor, Electrical and Information Technology, Faculty of Engineering, Lund University*
- The following employees at Volvo Car Corporation for excellent guidance and help throughout our project:
  - **Mats Fagerström**, *Software engineer, Infotainment department, Volvo Cars Corporation, Lund*
  - **Mia Månsson**, *Software engineer, Infotainment department, Volvo Cars Corporation, Lund*
  - **Yudi Hirata**, *Software engineer, Infotainment department, Volvo Cars Corporation, Lund*

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this section the background together with the thesis purpose and aim is presented.

1.1 Background

In recent years vehicles have become more and more technologically advanced and most newly manufactured cars today have the capability to establish connection with devices both through WLAN and Bluetooth [1]. Bluetooth was developed to replace physical cables for communication and has since then evolved into many use cases over the years, one of them being infotainment and in-car systems. An infotainment system typically acts as a combined headset, entertainment and navigation system by letting the user connect their device via Bluetooth to control media, phone calls, volume level etc. Different connectivity solutions e.g. integrated, embedded and tethered, are used by various systems to provide a pleasant user interaction with the vehicle. Bluetooth is an effective and useful technique but it is not impervious to interference and other issues, such as connection loss or lengthy waiting times for connection establishment. Adequate performance tests are a key to pinpoint where the user issues stem from. Enabling Bluetooth performance tracking over longer periods of time to monitor the effect of software updates and other system changes is an important aspect for development.

This thesis is a collaboration with Volvo Cars to develop tests and method to evaluate the performance of Bluetooth in the connection establishment to an infotainment system. Volvo Cars is an internationally recognized car manufacturer which is leading the global automotive industry in sustainability, autonomous drive and electrification as well as setting benchmarks in automotive safety and connectivity [2]. As the industry strives towards electrification and becoming more software oriented, Volvo Cars and their department of the Infotainment Platform are the first company to team up with Google to develop an Android based infotainment system. In this system, and area in general, it is of great interest to create an efficient way to automate the process of continuously measuring Bluetooth performance, to ensure that the connection quality is always satisfactory.

1.2 Thesis Purpose and Aim

The goal of this master's thesis is to investigate a method and develop a test for Bluetooth performance evaluation of the Android based infotainment platform when communicating with mobile phone devices. The purpose of the test is to be able to monitor and gather data on how well the Bluetooth connection performs, together with providing an insight into the root of any potential issue that causes an unsatisfactory performance.

Several Bluetooth parameters will be taken into account and be investigated to determine the most relevant ones for this scenario. Once a suitable parameter has been chosen, different ways of measuring it will be analyzed and an appropriate method will be chosen. During development of the test, it will be run in a lab environment to identify any potential problems that might occur and evaluate it's performance. Performance data from the test will be gathered for different phones and analyzed. This data will be shown as an example of what can be gathered from the test and how it can be visualized rather than an actual evaluation of the performance.

1.3 Related work

Related work on measurements of the performance of Bluetooth is presented in this section. The articles review the interference of WLAN on Bluetooth and study the limitation of Bluetooth in an indoor environment.

1.3.1 The effect on Bluetooth from WLAN in an automotive perspective

Current vehicles are getting more and more technologically refined, and the connected cars are getting more common than ever [1]. The increasing customer demand for connectivity, and the convenience of wireless connection makes the automotive market move more towards having both Bluetooth and WLAN integrated in their vehicles. More network activity and connected devices entails more occupancy in the frequency band, and with both WiFi and Bluetooth active on the unlicensed ISM band of 2.4 GHz [3][4] can contribute to interference.

A.Mourad et al. tested and discussed the WLAN and Bluetooth coexistence in their paper *On the performance of WLAN and Bluetooth for in-car infotainment systems* [5] where they concluded that WLAN using the non-overlapping channels 1,6 and 11 highly affects the Bluetooth performance, especially given high WLAN traffic load. They also noted that Bluetooth in one car may also be affected by surrounding cars when moving, especially at lower speed such as in a traffic jam. A suggested solution to solve this problem is having the integrated vehicle WLAN move to the 5GHz band, to off-load the traffic on the 2.4GHz band. Another suggestion to improve the Bluetooth performance is to have the WLAN standardised to only operate on channel 1,6 and 11, to at least leave a few Bluetooth channels free to be used in-between.

1.3.2 Bluetooth performance analysis in wireless personal area networks

The performance of Bluetooth is affected by distance and file size. Signal strength fades proportionally to the square of the distance and larger file sizes generate higher probabilities of errors. Obstacles can also affect performance. In a real world scenario these parameters are important factors. w J.Hipólito et al. discusses this and evaluates the performance of Bluetooth v2.0 + EDR (Enhanced Data Rate) in their paper [6]. The parameter used for performance analysis was the file time delay (FTD), the time passed from the start of transmission to the end of reception. FTD was measured for different ranges, with and without obstacles and with varying file sizes, simulating an indoor office environment. The experimental setup consisted of one controller node transmitting data and five responder nodes receiving. In addition two sniffers listened to the transmissions, one for Bluetooth and the other for WiFi, measuring FTD and errors in the data packages. The experiment showed that FTD increases with distance, obstacles and increases linearly with data size. The sniffing process of measuring FTD was done using a program called Colasoft Capsa, in this project customized tests will be written. This article is an insightful example of how to evaluate Bluetooth performance and gives inspiration as to what parameter one might use.

1.4 Previous Work

There is, at least to the authors knowledge, no previous published performance analysis or data collection of Bluetooth parameters in the automotive area. However, Yudi Hirata at Volvo Cars Corporation in Lund has provided a sample test, as an example which is depicted in Figure 1.1 below. The devices used for this test is the Volvo Infotainment Head Unit (IHU) and a Galaxy S9+ running on Bluetooth 5.0, Android 10.0 and with the chipset Octa-core (4x2.7 GHz Mongoose M3 & 4x1.8 GHz Cortex-A55) - EMEA. The graphs show the connection time and disconnection time for different Bluetooth profiles together with the CPU usage used during the test. Mr. Hirata's example illustrates how a Bluetooth parameters can be evaluated and displayed, which can give an insight on what the process may look like.

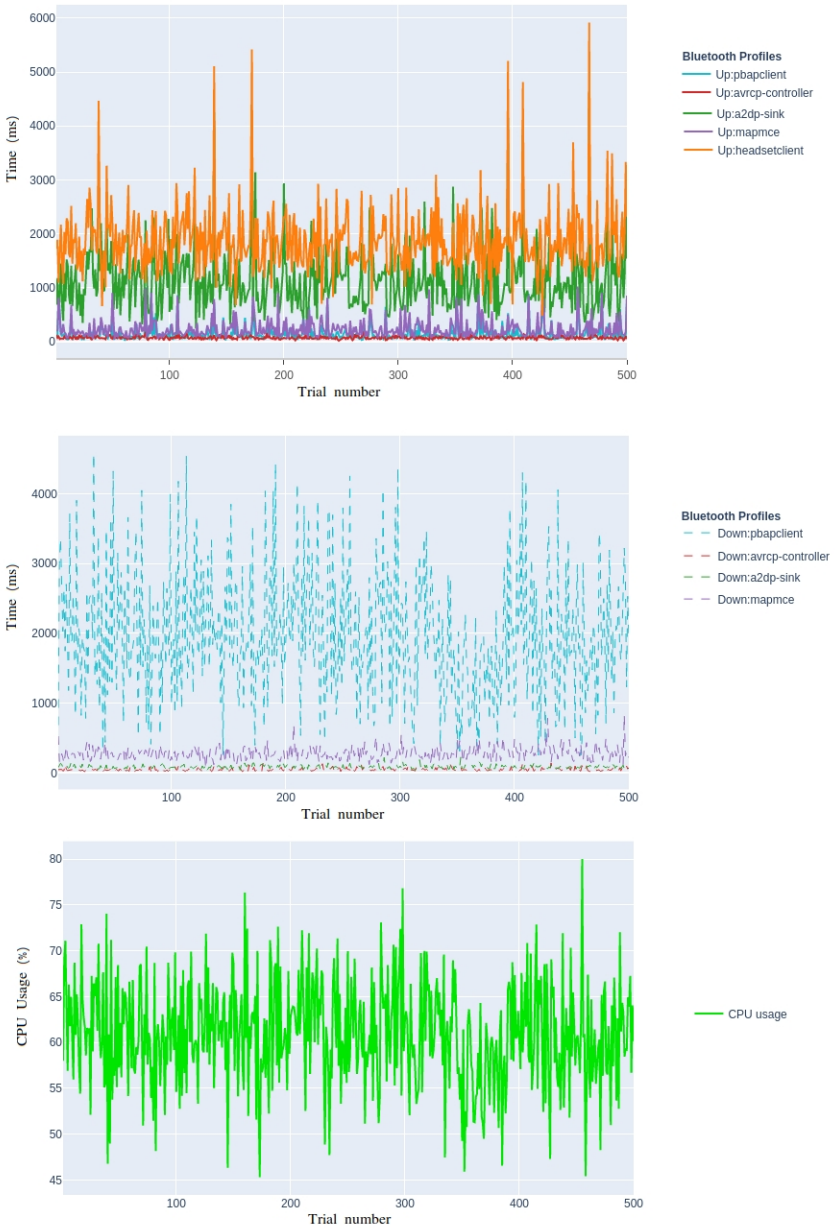


Figure 1.1: Sample of measurements by Yuri Hirata

1.5 Limitations

This master's thesis is mainly focused on developing a test of the performance of Bluetooth in the infotainment system of Volvo Cars. Due the complexity of having an all coverage system test, the test is rather to be seen as a proof of concept, not as a finished product but rather something to build upon in future development. While the test results are presented and discussed in later parts of this report, improving on the Bluetooth connection is not a goal.

1.6 Scientific contributions

The work in this master's thesis will contribute by providing an insight in how Bluetooth connectivity can be evaluated in an automotive setting together with the Android platform. It will provide information on how to identify hindrances in the connection and how to detect it on the platform. This will enable further work to focus on how to deal with each instance of hindrance to improve the overall performance of the Bluetooth connectivity, and overall in Android automotive infotainment units.

Chapter 2

Theory

There is no method of measuring that works best in all situations. One procedure might be beneficial in some cases and others not. It is important to take this into account when choosing the path forward, together with understanding chosen measuring parameters. In this section, different testing parameters and methods will be discussed along with a comprehensive background on Bluetooth and the aspects of relevance to this project.

2.1 Bluetooth

Bluetooth is a short-ranged wireless communication technology intended for cable replacement [7]. Bluetooth works on the unlicensed frequency band ISM, it spans 2.4GHz to 2.4835GHz. The frequency band is divided into 79 channels, each with a range of 1MHz. Bluetooth is managed by the Bluetooth Special Interest Group (SIG). There are two types of Bluetooth technologies, Bluetooth Classic and Bluetooth Low Energy (BLE). Bluetooth Classic is the focus of this report and, if not otherwise stated, all mentions of Bluetooth from here on refers to Bluetooth Classic v.4.2.

Bluetooth has two modes for data transfer, Basic Rate (BR) and Enhanced Data Rate (EDR). BR employs a technique called Gaussian frequency-shift keying (GFSK) and, in the optimal case, achieves a data rate of 1 Mbit/s. EDR uses differential quadrature phase-shift keying ($\pi/4$-DQPSK and 8-DQPSK) to achieve data rates of up to 2 and 3 Mbit/s.

The Bluetooth technology is arranged in a controller-responder architecture. In a typical setup there is one controller and up to seven responders, this network is known as a piconet. Adaptive Frequency Hopping (AFH) is used for communication and all devices in the piconet are synchronized using a clock given by the controller. The controller clock ticks every 312.5 microseconds and two ticks constitutes a time slot for sending and receiving. Once a controller and responder connection is set up, the controller sends a hopping sequence to the responder, with information on which channels to send and receive from.

2.1.1 Bluetooth Stack

In Figure 2.1 the structure of the Bluetooth Protocol Stack is shown [8]. The structure consists of a physical layer, a data link layer, a middleware layer and an application layer. The physical layer takes care of the modulation and demodulation of radio signals into data packages. The physical layer also focuses on the physical attributes of the transceiver[1] and defines connectionless and connection oriented links.

The base band link layer executes the establishment of connections within the piconet. The established links are then managed by the link manager on the data link layer which administers the security, encryption and authentication. The logical link control adaptation layer (L2CAP) is the interface between the upper and lower layers and handles multiplexing and segmentation. L2CAP converts data packages from the upper layers into a format the lower layers can process. Radio frequency communication (RFcomm) is a protocol used to emulate RS232[2] serial connections over L2CAP channels and has the capability to emulate several serial connections into a single data connection. The Telephony Control Protocol Specification Binary (TCS-BIN) is a protocol for telephony services. It handles various functions such as call notifications, audio connection establishment and call terminations. The Service Discovery Protocol (SDP) is responsible for keeping track of what services are available on other connected Bluetooth devices. The application layer is responsible for management and interactions with applications and predefined profiles.

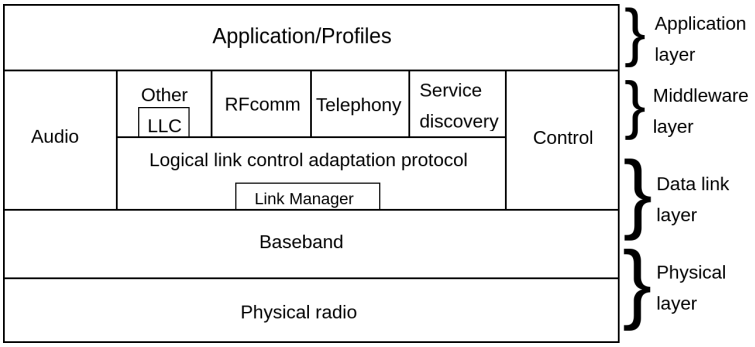


Figure 2.1: The Bluetooth Protocol Stack

2.1.2 Bluetooth packet

In Figure 2.2 a typical Bluetooth packet structure is shown [9]. The Channel Access Code is used to allow for transmission on a specific channel and to disregard packets sent on other channels close by. The Packet Header includes the logical transport address (LT_ADDR), which is utilized by all receiving devices to verify

[1] Transmitter and receiver

[2] Recommended Standard 232, a standard for serial communication transmission of data in telecommunication

which device the packet was intended for. EDR packages have a guard time and synchronization sequence used for changing the modulation type of the physical layer. The Payload Header includes a field for routing and one for the length of the payload. Some packages also contain a Cyclic Redundancy Check (CRC) at the end of the header to check for errors. The Payload body carries the actual data of the packet. Packages using AES-CCM, a block-chain encryption algorithm, contains a Message Integrity Check (MIC) before the CRC at the end.

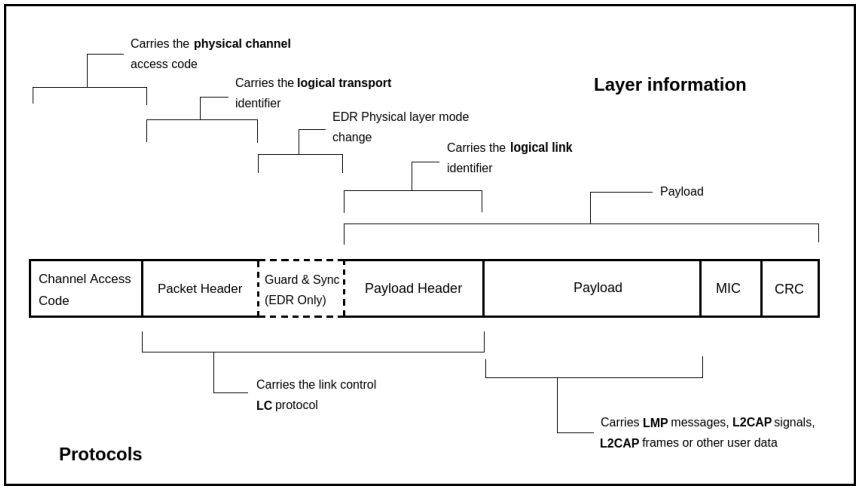


Figure 2.2: Bluetooth packet structure

2.2 Bluetooth profiles

In order for an application to communicate between two devices, Bluetooth profiles are needed [9]. If two profiles conform to the same profile they uphold certain requirements that allow them to cooperate. Depending on the desired functionality different profiles are needed. Bluetooth profiles specify the required function of each layer in the Bluetooth system, from the Physical Layer to the Logical Link Control and Adaptation Layer Protocol (L2CAP). The profile determines how the layers relate to each other and how they work between connected devices.

The profiles also define what type of data to be transferred and the functionality of the applications. All profiles specify the requirements for connection between devices. Many profiles depend on more fundamental ones, inheriting parts of the profiles rule-set.

2.2.1 GAP

All Bluetooth devices implement the Generic Access Profile (GAP)[9]. The profile specifies the basic requirements. For Bluetooth Classic this consist of a Radio, Baseband, Link Manager and L2CAP. GAP also defines the function of how devices discover and establish connection along with security and authentication methods.

2.2.2 A2DP v.1.2

Advanced Audio Distribution Profile (A2DP)[10] is a profile that defined the standard for high quality audio distribution in mono or stereo. It is used to distribute audio from a source to a sink. In a car related environment, a mobile phone would typically act as the source and an infotainment system as the sink. The source may also be a microphone located in the car, transferring audio to a mobile phone which then operates as a sink. The profile is dependent on the Generic Audio/Video Profile to set up audio streaming, the data is then compressed to use the limited bandwidth efficiently. A2DP does not handle any controls or functions of the audio, but is meant to be combined with a suitable control profile such as Audio/Video Remote Control Profile (AVRCP).

2.2.3 AVRCP v.1.2

Audio/Video Remote Control Profile (AVRCP)[11] defines a set of rules to be followed for a Bluetooth connection between a device working as an audio and video controller and its recipient. AVCRP is dependent on GAP.
In a typical connection there are two roles, the controller and target . Examples of controllers are PCs, TV-controllers, cell phones or tablets in a car infotainment system. Examples of targets are audio-players, TVs or headphones. The profile is divided into four categories. Category 1 defines the functions for a player or recorder. It works regardless of the media, audio or video. Category 2 describes the functionality of an audio amplifier or a video monitor. Category 3 specifies the operations of a video and audio tuner. The fourth category characterizes the basic operations of a menu. The way the menu data is displayed is not defined in the profile, this could be an external monitor, as in a car infotainment system, or a display integrated into the device.

2.2.4 HFP v.1.5.1

The set of rules defining a hands-free device connecting to a mobile phone is called Hands-Free Profile (HFP)[12], which is most commonly used in wireless headsets and car infotainment systems. Implementing the hands-free profile allows the device to connect wirelessly to a phone and to function as the phone's audio output and input without direct access to the phone. HFP is dependent on the Serial Port Profile (SPP) and GAP. SSP is used for emulating a cable connection between two devices.

In a typical connection there are two units, the Audio Gateway (AG) and the Hands-Free unit (HF). AG is the access-point of the audio in- and outputs, most commonly a cell phone. HF acts as the remote audio in- and output tool, such as a wireless headset or an infotainment system in a car. AG and HF have certain feature requirements in HFP. Some of the requirements are standard for both units, while some are optional for one while required by the other. For example AG and HF both need to be able to accept an incoming call, but placing a call using memory dialing is optional for HF while required for AG. This is quite reasonable as a normal wireless headsets lacks the ability to make a phone call, while in an infotainment system this feature is desirable.

2.2.5 MAP v.1.1

The Message Access Profile (MAP)[13] outlines the features and procedures to follow for devices sending and receiving messaging objects. The profile relies on a server-client model in which the client starts the interaction. MAP enables the client to send, receive, delete and generate messages and notifications from the server device. A major usage of MAP is in the car domain, providing users the ability to send text messages in a safe way with text to speech. MAP is dependent on the Generic Object Exchange Profile (GOEP), SPP and GAP. There are two roles defined for MAP, Message Server Equipment (MSE) and Message Client Equipment (MCE). MSE stores all messages and sends and receives them, it is also capable of providing the MCE with messages. MCE uses the messages stored in the MSE for browsing and display and can also create and upload messages to it. Typically in a vehicular scenario the car is the MCE and the cell phone is the MSE.

There are five main requirements on the MSE and MCE covered by this profile. The MSE must be able to notify the MCE of a message arrival. Messages stored on the MSE must be retrievable by the MCE. The MCE needs to be able to upload messages to the MSE. The MCE must be allowed to delete messages stored on the MSE. Lastly, the MCE must have the ability to send messages through the MSE.

2.2.6 PAN v.1.0

Public Area Networking Profile (PAN)[14] defines how two or more devices can create an ad-hoc network through a Bluetooth connection as well as how to connect via a network access point to a remote network. PAN also allows Bluetooth tethering between devices in the network, making internet access shareable between them. PAN is dependent on GAP and handles the profile roles: Group Ad-hoc Network, Network Access Point and Personal Area Network User.

Group Ad-hoc networking lets devices create networks (piconet) on their own without any additional infrastructure or hardware. It specifies that the piconet in turn contains a single controller which will communicate with 1 to 7 other responder devices. A Network Access Point is in turn a device in a network that act as a pathway for communication in a Bluetooth network to another network (e.g. internet). A Personal Area Network user (PANU) may also connect to another PANU directly, which is also a combination of a PAN in Bluetooth defined in the protocol.

2.2.7 PBAP v.1.2.3

Phone Book Access Profile (PBAP)[15] is a profile that sets the exchange protocol of phone book objects between devices. It handles the communication with a client-server interaction model as a base, and is created with the usage case of automotive hands-free devices, such as an infotainment unit, in mind.

It handles the procedure by letting the client (e.g. infotainment unit) or the server (e.g. mobile phone) initiate bonding. After a secure connection is established the client can download one or several phone book entries and access the

list of phone book entries, call history and subscriber number information[3] that is stored in the server. The profile does not allow any alteration of the content in the original phone book object, since the objects are read-only.

2.3 Real world vs. lab environment

There are essentially two different categories of tests when evaluating the performance of any market product, tests in a lab environment or in the real world. In this report the real world testing equates to measurements done in customer cars or cars that are a part of the Volvo co-development program. Co-dev cars are Volvo cars that can be leased by employees which are then run as test vehicles with the newest software upgrades to evaluate and collect data.

There are advantages and disadvantages to evaluating performance in a real world setting and in a lab. In a lab one can perform more kinds of tests than in a real world setting. For example, evaluating the lower layers in the Bluetooth stack is more easily done in a lab, but would be difficult in a customer car. Another advantage of the lab is that measurements can be done quicker and the data can be obtained directly, making it possible to quickly evaluate the results. The test environment in a lab is easily controlled and eliminating random variables such as interference and distance between Bluetooth devices are manageable. In contrast, measuring repeatedly with the same setup in a controlled environment is not indicative of a customer experience. Because of this, customer car testing is the best option in this regard.

An advantage of a customer car test is the large amount of data that can be collected. The data coverage is also very varied, due to the many different car models with different phones connected, resulting in a massive amount of valuable data. This could assist in finding and focusing on the troublesome scenarios, and analyze them in a lab environment.

Although real world testing would be of more interest, the focus in this study will be in a lab environment due to it being more suited to the scope and time frame of this master's thesis.

2.4 Application layer vs. Physical layer

Measuring on the application layer is beneficial when investigating the user experience due to it being on the top of the Bluetooth stack. Measurements like connection times on the application layer indicate the duration of the waiting time when a user establishes a connection or starts a service. This can assist in finding slow processes that are commonly used or user experience processes that are worsened with delays. This collected data will then include all delays and interruptions found in the Bluetooth stack, thus making changes in performance between updates easily detectable.

Connection time measurements on the physical layer will however not detect this since some delays may be in a higher layer than the isolated Bluetooth signal.

[3]The SIM's telephone number

Data from the physical layer may although be interesting as a substantial amount transfer time is spent in this layer. Depending on what parameter is of interest, different layers are more suitable for measurements. Packet loss and throughput are parameters that might be preferable to handle on the physical layer while latency and device discovery time may be preferable on the application layer. However, the undoubtedly superior option is to measure on all protocol levels from physical to application layer, compare values and thus find where the bottle neck is located.

2.5 Test parameters

Before a final testing parameter can be decided upon, different options need to be considered. Different perspectives have to be taken into account when designing a useful test. The goal for this thesis is to assist in improving the user experience of Volvo Cars infotainment unit. To achieve this, the parameter tested should be as relevant to the user experience as possible. The test should also be implemented in a way so it can be run repeatedly. In the following section the pros and cons of different testing parameters are reviewed.

2.5.1 Latency

Latency is the time delay of a system. In a Bluetooth setting this refers to the time for an input from one of the connected devices to affect the other. For example, when a customer makes a call from the IHU there is a slight time delay before the call is made from the phone, this is called latency. Latency is relevant in many situations in a vehicular environment, making it an applicable parameter for Bluetooth performance testing.

2.5.2 Throughput

The amount of data sent per time unit, typically bit/s, is called throughput. A high and steady throughput is particularly important in systems where plenty of data is being sent. This is not the case in a car environment. The highest data transfer is needed when streaming music. High resolution music streaming typically reach maximum bit rates of 320 kbit/s [16], which is only a fraction of the capacity of Bluetooth. For this reason testing throughput is not relevant for this thesis.

2.5.3 Device discovery

The first time a Bluetooth connection is made, the devices need to go through a device discovery process. This can sometimes be a lengthy procedure and from a customer perspective, it can be a big nuisance if it's too lengthy. On the other hand, pairing new devices is usually only done a couple times per customer, depending on the number of phones used. While discovery time is an interesting testing parameter, it is not the most crucial performance parameter due to its low occurring frequency and it is consequently ruled out as an option.

2.5.4 Connection time

Two previously paired Bluetooth devices do not need to go through the discovery process when connecting. However, each time a new connection is established, they need to connect through the relevant profiles. The connection time is usually fast, but in case of lost packages or other faults this time can significantly increase. Each time a customer enters their vehicle the phone automatically connects to the IHU, for an average user this occurs several times per day. Auto-connect works slightly different for fully electric vehicle compared to hybrid cars developed by Volvo Cars. In the electric vehicles a sensor is situated in the driver seat that sends a signal to the IHU to set up a connection as soon as someone gets seated. In the hybrid vehicles, the same signal is sent when the engine is started.

The combination of the potential for delays and the fact that connection establishment occurs frequently makes this a highly interesting parameter and the one chosen for testing in this thesis.

2.6 Measurement approach

In designing a test, the measurement approach must be considered. There are many tools and methods available for measurement of Bluetooth performance. An important component to examine is which parameters a given method can measure, and if those parameters are of relevance to the desired test. In the following sections the pros and cons of different approaches are examined.

2.6.1 Dumpsys

Dumpsys is an Android tool used to fetch diagnostic output and system information about devices through the Android Debug Bridge[17][18]. A plethora of information can be retrieved depending on the chosen command. To filter for data related to Bluetooth, the command `bluetooth_manager` is used. Dumpsys together with the command have the benefit of being informative and provide a substantial amount of information such as adapter properties, adapter state, Bluetooth profile data and profile states. The information is also presented in a readable string, and can be retrieved as a file on the local system.

However, Dumpsys is not ideal when doing continuous measurements due to the impact on system performance it has when run. It is preferably used in single instances such as when investigating outliers or debugging.

2.6.2 Perfetto

Perfetto is an open source tracing program made by Google, allowing users to pull system-wide performance traces from Android devices of many performance parameters [19]. Perfetto also provides a web based graphic interface, to visualize the tracked traces, see Figure 2.3. Obtainable data include CPU and memory usage, application threads, Bluetooth profile connections etc. Perfetto is available since Android 9 (P) and already enabled as default install since Android 11 (R), which makes it effortless to test out and start running on the IHU. It can collect

custom trace points and allows the use of triggers on threads to start and stop the tracing. Perfetto is beneficial for analysis of a single instance, offering plenty of information about the various running threads. However, for testing specific parameters repetitively it is not as convenient, and is in this case not used by us as a measurement method.

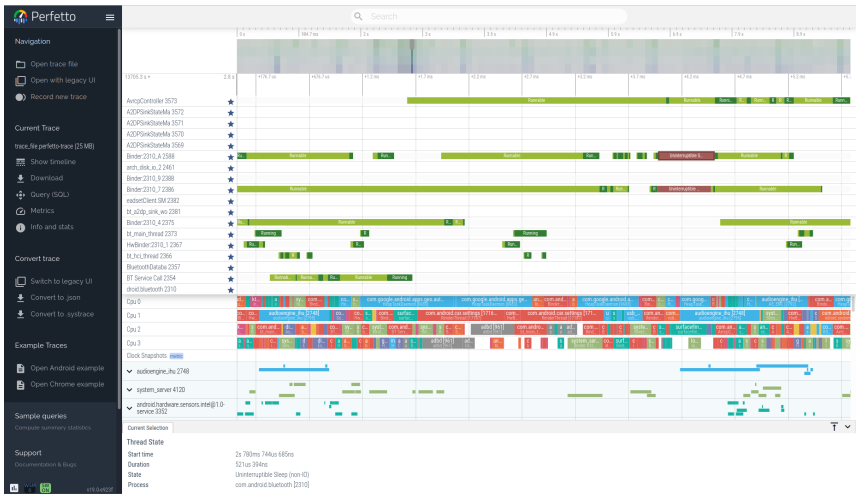


Figure 2.3: Perfetto Trace Viewer

2.6.3 Firebase

Firebase is a platform developed by Google, providing users with a variety of services and tools to assist in app development[20]. It is a Backend-as-a-Service[4] and a way for users to connect their apps to a backend cloud storage. It provides different services to build, release, monitor and keep the app active. Among these is Firebase Performance Monitoring which is a tool to check an apps performance issues. Firebase allows the user to set custom traces and metrics to monitor them, and keep track through the Firebase console. It can also trace application start-up time, http transactions, network information, crashes etc. The drawback with Firebase for Bluetooth performance analysis is that the tool is more oriented towards management of applications rather than system monitoring. It is also a part of Google Services, thus making it unavailable to monitor in some regions of the world.

2.6.4 Wireshark and BTmon

Wireshark is an open source tool used to fetch and analyse the details of network packages[21]. It provides filtering options and a byte interpreter which helps the user to understand the information given in each byte. Wireshark can open files given by other package sniffing program such as the Bluetooth monitor BTmon,

[4]A cloud service model in which the developer outsource all back-end work of a project

which is a part of the official Linux Bluetooth stack BlueZ. While the combination of BTmon and Wireshark results in a very capable analysis tool, it is not useful for tests designed to simulate user experience. This should be done by testing parameters in the application layer. Nonetheless, a comparison between a test in the application layer and an analysis in Wireshark could be a good way to evaluate the performance of the Bluetooth stack.

2.6.5 DID

An alternative approach to the aforementioned methods is to implement the test directly in the source code of the IHU. Volvo Cars use this approach in designing a variety of tests, a module called Data IDentifier (DID) is used for this purpose. The goal of DID is to implement tests in all compatible user vehicles.

Writing the test from scratch comes with a few upsides. Using pre-existing tools limits the test to the parameters measurable by the specific tool, this is not a concern with custom tests. Writing a custom test by adding functionality to the software of an application does however restrict the usage to the application layer, but this approach gives more freedom to create a specific test. A test implemented in DID could be run both in a lab environment and cars. It would also enable continuous measuring, allowing for tracking of performance changes between software updates.

For the reasons mentioned above, this method is used for designing the Bluetooth performance test presented in this thesis.

Chapter 3
Technology

This section describes the experimental system in which this project was implemented. The hardware and software structures of the project are presented along with some crucial parts of the Android framework. It also gives an overview of the technology of Bluetooth with descriptions of Bluetooth profiles relevant in a car scenario.

3.1 Hardware

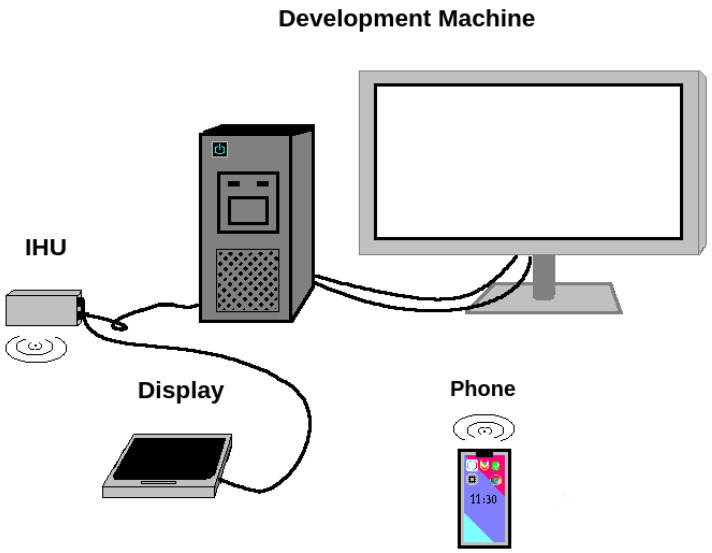


Figure 3.1: Structure of lab environment setup

Figure 3.1 shows how the rigs are set up in the lab environment. The Infotainment Head Unit (IHU) is at the core of the system hardware. It consists of two parts, a media processor and a vehicle interface processor. The vehicle interface processor handles signals and processes related to the driving and the

media processor controls all media processes and the Bluetooth connection. The media processor is a 2.4GHz Apollo Lake Premium with 4x1 GB RAM and 128 GB flash storage. Interaction with the IHU is done via the Center Stack Display (CSD). CSD is a touchscreen 9 inch display with one hardwired home button. It is connected to the IHU with LVDS cables. The IHU has an internal Bluetooth chip (CSR8311) used for connection.

3.2 Software

All applications along with the graphical user interface are written in Android and handled by the IHU. The foundation of the software system run on the IHU is developed by Google, Volvo Cars refine the code and make changes to fit their specific cars. The test is developed in Android Studio and integrated in the main code as a separate branch.

3.2.1 Broadcasts, Intents and Intent Filters

Broadcasts [22] can be sent and received by apps, services and the Android system. When certain events happen e.g. an established connection is lost, a broadcast is sent informing that the action has happened. Apps and services that have listeners set up to be informed about the specific action will then receive the broadcast and process the information in an appropriate manner.

An intent [23] is a message with instructions of an operation to be performed, or alternatively information of a transpired event. Intents have three basic functions: starting an app, communicating with a service or delivering a broadcast. For Bluetooth related events, intents are used to deliver broadcasts. In this case the intent contains information, rather than instructions, about the event that happened. Thus, when the state of a Bluetooth profile changes e.g. going from `disconnected` to `connecting`, a broadcast message wrapped in an intent is sent out to all listening services. The intent contains information in the form of a *component name*, *action*, *data*, *category*, *extra* and *flags*. The parts of interest in this thesis are action and extra. The action is a description of the event that has taken place in the form of a string. For the previously mentioned example, action is `android.bluetooth.headset.profile.action.CONNECTION\_STATE\_CHANGED` , given that it concerns the HFP (or headset) profile. Information about which type of connection state change has occurred can be extracted from *extra* part of the intent.

Services listening for broadcast intents are called broadcast receivers. To make sure the service receives the desired broadcast, an intent filter is used [23]. The intent filter tells the receiver which type of broadcasts to listen to.

3.2.2 Android Debug Bridge

Android Debug Bridge [18] or ADB is a very useful tool executed from the command line. It facilitates communication between devices and apps. Among other options, ADB enables the installation and debugging of apps on a device, enables access to files stored on a device and allow for printing of console logs.

ADB is a client-server based program with three components, a client, server and deamon. The client runs on the workstation, sending out commands. The deamon is tasked with running the commands as a background process running on the given device. The server conveys information between the deamon and server, running as a background process on the workstation.

Chapter 4
Methodology

The chosen method was to implement a program running locally on the IHU as a part of the main source code on the unit. The test is designed to simulate the auto-connection process occurring when a driver enters the car with a previously paired Bluetooth device. The time for the whole scenario is measured, i.e. from the time the auto-connection initiates until all the connections are established and all needed data is downloaded.

Each process is measured separately and recorded to be able to identify any deviations. The processes include the download of the phone book data, including contacts and photos attached to them, together with the connection establishment with the Bluetooth profiles PBAP, HFP, A2DP, MAP and AVRCP. This is also combined with a measurement of the entire process, from start until everything is finished.

4.1 Initial test

The first thing that was set up was a small initial test, which recorded and collected the connection establishment times for the HFP, A2DP and PBAP profiles. This was done with a Java class in the IHU source code which inherits BroadcastReceiver[24] and uses an IntentFilter[23] to listen to Android's broadcast intents. When a state change occurs the intents are sent out and announce the information of a change in a profile, which then the filter is configured to catch. The four different states that were chosen to be listened to were *connected*, *connecting*, *disconnecting* and *disconnected*. As soon as a broadcast intent announcing a change in state, the previous and current state is inspected to determine if the connection process is initialized or finished. If e.g. the intent provides the current state as *disconnected*, the previous state defines if it was a successful disconnection i.e. going from *disconnecting* to *disconnected*, or if it was a connection establishment failure, going from *connecting* to *disconnected*. When determined, a timestamp is taken when connection is started and another when the connection process is finished with the difference between the timestamps constituting the connection time.

Each time a connection measurement is produced, it is added to a JSON array in a JSON file with a structure for each profile, see Figure 4.1 and section 4.4 for more details.

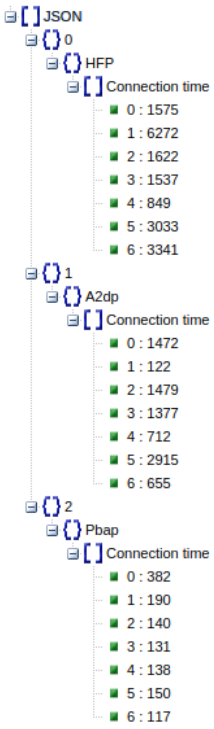


Figure 4.1: Initial test JSON file structure of the output

4.2 Profile addition

In addition of the profiles in the initial test program, the MAP and AVRCP profiles were added to expand the program. Collectively compiled together creating the list seen in table 4.1 over all the recorded profiles. The times for each profile is taken simultaneously for each profile when establishing a connection after the IHU starts its auto-connection process.

The PAN profile is rarely used by users. In the vehicles of Volvo Cars an internal internet connection is included. Not many customers choose to connect to the internet via their phone, for this reason the profile was excluded from the test.

Bluetooth profiles
A2DP
AVRCP
HFP
MAP
PBAP

Table 4.1: List of recorded Bluetooth profiles

4.3 Mean and standard deviation

To keep track on how different devices compare on an average to each other the mean and standard deviation were added as a recording function. The mean value and standard deviation of the connection times are automatically and continuously updated as more measurements are collected. Each time a new value is added to the array, the mean value is calculated as:

$$\mu_N = \frac{\mu_{N-1} * (N-1) + x_N}{N}, \ \mu_1 = x_1 \tag{4.1}$$

Where x_N is the N:th data-point and N is the number of data-points and the first mean is equal to the first data-point.

Similarly to the mean, the standard deviation is updated for each newly received data-point with the following formula:

$$\sigma_N = \sqrt{\frac{((N-2) * \sigma_{N-1}^2 + (x_N - \mu_N)(x_N - \mu_{N-1})}{N-1}}, \ \sigma_1 = 0 \tag{4.2}$$

The first standard deviation σ_1 is set to zero, as there is no variance given one data-point.

4.4 File structure

The data is saved in JSON format. An outer JSON object contains one JSON object for each profile and one to catch if the received action intent is null. Each profile JSON object contains one JSON array containing connection times, and calculations of the mean and standard deviation of the connection times. The connection times are indexed to easily see which times belong to which connection establishment. The purpose of using this format is to allow for easy expansion of other categories connected to the profiles, such as phone book download times. All measurements are then recorded, updated and saved locally on the IHU each time a new auto-connection is finished, see Figure 4.2.

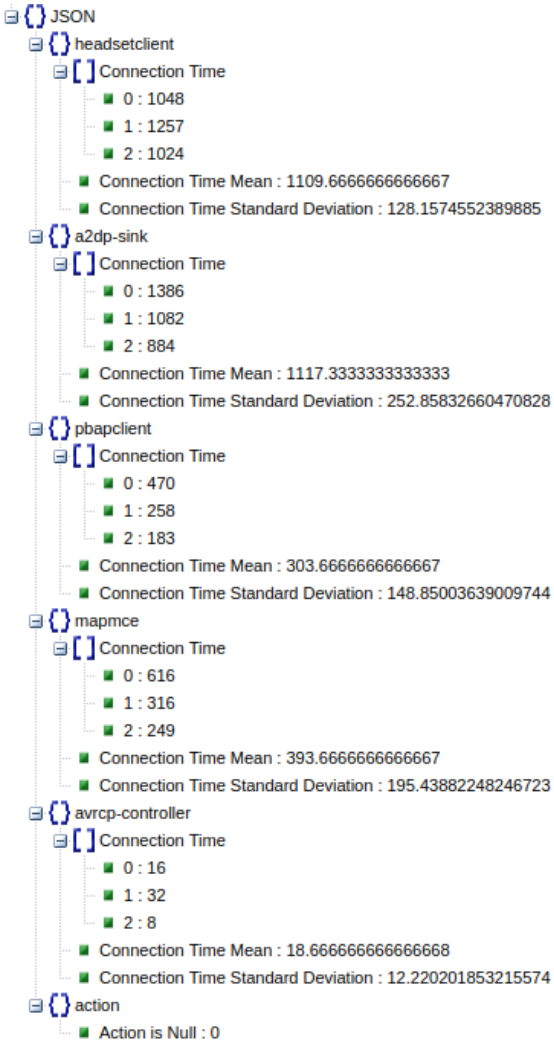


Figure 4.2: JSON file format of the output

4.5 Phone book download time

Due to the phone book information being the main data mass to transfer via Bluetooth when auto-connecting, it was chosen to be recorded in addition to the profiles. The phone book data is collected from the contact provider [25] which gives access to a database containing the phone book where each entry in the database contains a contact. This database is the same as the one used by the dialer app, the application handling phone calls and messages, for access to the contacts. By querying the database, information can be extracted about each contact such as the name, number of the contact, the time of download and whether or not the contact contains a photo. Each time a new contact is downloaded to the IHU the JSON-file is updated with a new entry.

The IHU starts downloading the phone book data when the PBAP Bluetooth profile is finished connecting, thus the start time for the phone book download time is set when PBAP is in the *connected* state. The end time is taken as the last download time of the last contact to be downloaded. To ensure the phone book is completely downloaded, the database is queried multiple times pausing the thread in between. Only when there is no change between consecutive queries, the downloading is deemed done. The phone book download time is then calculated as the time difference between start and end time.

Along with the download time, the number of contacts and number of photos in the phone book is saved for each measurement. The number of rows in the database is counted to get the number of contacts and the number of rows containing a photo is counted to get the number of photos. This is done for two purposes, to make sure the entire phone book has been downloaded and to allow for comparison between download speed for different numbers of contacts and photos. Figure 4.3 shows the structure of phone book download time and info in the larger JSON file.

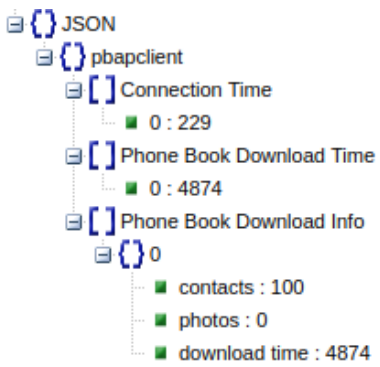


Figure 4.3: JSON format of the phone book download

4.6 Total time measurement

The vehicles produced by Volvo Cars can be categorized into two types, BEV and Non-BEV cars. BEV stands for *Battery Electric Vehicle* and are fully electric cars while Non-BEV cars are the hybrid models and car models propelled by liquid fuel. Apart from differing in propulsion, BEV and non-BEV also differ in functionality modes. These modes are called usage modes for BEV cars and ignition modes for non-BEV models. The usage and ignition modes each consist of three different modes. Usage modes are *Passive*, *Comfort* and *Drive* and ignition modes are *0*, *I* and *II*. The usage and ignition modes have similar functionalities and for the purpose of this thesis they can be thought of as equivalent. The transition from the first mode, *Passive* or *0*, to any of the other modes triggers the Bluetooth adapter to turn on. In BEV models there is a sensor in the driver seat activating this mode transition when someone sits down. In non-BEV cars there is no such sensor, instead the start knob is turned clockwise to change ignition modes.

The total time measurement measures the time it takes from a person sitting down in the driver's seat, or alternatively (for non-BEVs) when turning the start knob, until all the profiles are connected and the phone book is fully downloaded to the IHU. The starting timestamp is taken when the usage mode change from *Passive* to *Comfort* or *Drive* or alternatively when the ignition mode changes from or *0* to or *I* or *II*. The last timestamp is taken when the connection is established and the phone book is downloaded, allowing the user to make a call to a contact in the phone book. The total time measurement is then taken as the difference between the start and end timestamp in milliseconds. Figure 4.4 shows the JSON structure of the total time measurements.

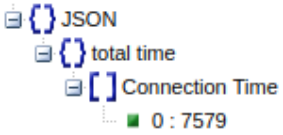


Figure 4.4: JSON format of the total time

4.7 Concurrency issues

When a larger quantity of data is collected in conjunction with the data being time sensitive the measurements need to be calculated in parallel to not interfere with each other. Due to the phone book being the largest and most time consuming process, and to ensure the profile measurements are not affected by it, the phone book download time is measured in a separate thread working to collect data from the contact provider. In parallel, the start and end timestamps for each profile are recorded and registered. When the Bluetooth adapter is turned off, constituting the end of a test run, the saving procedure starts by taking the stored values and posting them to a separate thread which in turn handles the saving to file locally on the IHU. In the same manner, the loading is done in parallel when a device is connected or the file needs to be loaded before saving. This is likewise to

ensure the loading and saving does not interfere with timed procedures. Because of concurrency issues and the complex setup needed when recording and continuously updating the standard deviation and mean value for each profile at each run, the previous solution discussed in section 4.3 had to be omitted. It was replaced with a script written in MATLAB using all data points collected from each test session. The script calculates the mean through the standard formula of arithmetic mean and standard deviation.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

4.8 Broadcast reception

In the source code running the IHU, three other services catching broadcast intent for connecting Bluetooth profiles are found. To investigate the consistency of when the broadcast intents are received, a test is made. Logs for when the intents are caught are printed, utilizing logcat in ADB. All logs contain timestamps, these times are recorded for the profiles A2DP, HFP, MAP and PBAP. Twenty different connections are made where timestamps for connecting and connected are recorded. The difference in milliseconds, between connecting and connected, is calculated constituting the connecting times. The connection times are compared to find the maximum difference between the four services for each measurement attempt. All these recordings and calculations are done in Excel and then exported to MATLAB to visualize the results in Figure 5.6.

4.9 Data collection

The data collection is done in a lab environment on the IHU rig utilizing ADB to automatically turn on and off the Bluetooth adapter with a set delay of 15 second in-between. Two phone models with different operating systems, named "Phone A" and "Phone B", both having 100 contacts each with no photos, are run separately with identical IHU units. When the Bluetooth adapter is turned on, the devices auto-connect to the IHU and the connection establishment and the data from the phone book commences. When the set delay time is reached and the Bluetooth adapter is turned off, the data is stored and saved on the IHU. If a profile does not have time to finish its connection establishment, the data is left out for that profile and the JSON file gets a placeholder value of -1 instead. The phone book data is also left out and replaced with a placeholder value in case it doesn't download all the needed data. The collections is typically done over the weekend, gathering thousands of data points.

In doing this the total time measurement is omitted as there is no change in usage mode or ignition mode when turning on and off the adapter.

As previously stated, to include the total time in the data collection, one needs to change the usage/ignition mode, as this is the trigger for the start time. In the

lab environment this is not as easy as one might expect. Simply restarting the IHU will not change the usage/ignition mode. It is however possible to change the mode of the rig manually using a software called Carsim. Setting up an automatic data collection using Carsim is not easily done and is deemed outside of the scope for this project. The total time measurements are done manually, changing the usage/ignition mode and turning on the Bluetooth adapter directly after. After 30 seconds the Bluetooth adapter is turned off, and process is repeated 20 times for the two phone models.

4.10 Presentation of data

The presentation of the collected data is done in MATLAB. The JSON file, containing all measurements, is exported from the IHU using ADB. A script written in MATLAB then imports the data into a struct, using functions reading JSON. The connection times of each profile together with the phone book download times are extracted into separate arrays. The arrays are then used to graph the measurement, with the measured time on the y-axis and attempt number on the x-axis.

Included in the script are options to remove outliers and failed measurements, represented in the JSON file as *-1*. Datapoints deviating more than two standard deviations from the median are classified as outliers. Around 95% of all values are within two standard deviations of the mean in a normal distribution. Almost all deviating data points from the measurements produced by the project are higher than the mean, this indicates that the measurements are not normally distributed. The median is used instead of the mean to remove data points that are deviating most from the bulk of the measurements.

Chapter 5
Results

In this section the results from the test are presented with graphs and tables over connection times together with mean and standard deviation values. Firstly results from the initial test are shown, then the final test result are shown after concurrency issues are resolved. The measurements for the phone book download time are also presented, followed by the total time results. Analysing measurements of the broadcast reception is also given.

5.1 Method results

The method approach presented in this thesis have two main benefits when measuring the data shown in section 5.3 - 5.5. The first being the number of devices necessary is low since the only needed hardware is an IHU, with the test software downloaded to it, together with one or more mobile phone devices, depending on what kind of investigation is in process, to run a test session. The second reason is it being a low maintenance test due to it only needs an initial set-up to start and run. When in the lab environment, a PC connected to the IHU automatically runs a script turning on and off the Bluetooth adapter, which means there is not any need for active control as long as the auto-connection is enabled on the mobile phone device and the script actively running. The same argument would apply if it was installed in a vehicle, however it would then automatically record the measurements whenever the infotainment system is started and the auto-connection process runs.

The encountered failed runs were in general only due to a mobile device had discharged, or an automatic software update during the testing session. The data points which were collected during the session were however still available and saved locally in the IHU.

The results the method provided are described in to two parts, the initial demo in section 5.2 and the final version in section 5.3. The results shown in section 5.2 are before concurrency issues were handled and because of this may occur a bit deviating. These results show what data can be collected, how the data can be compiled and presented.

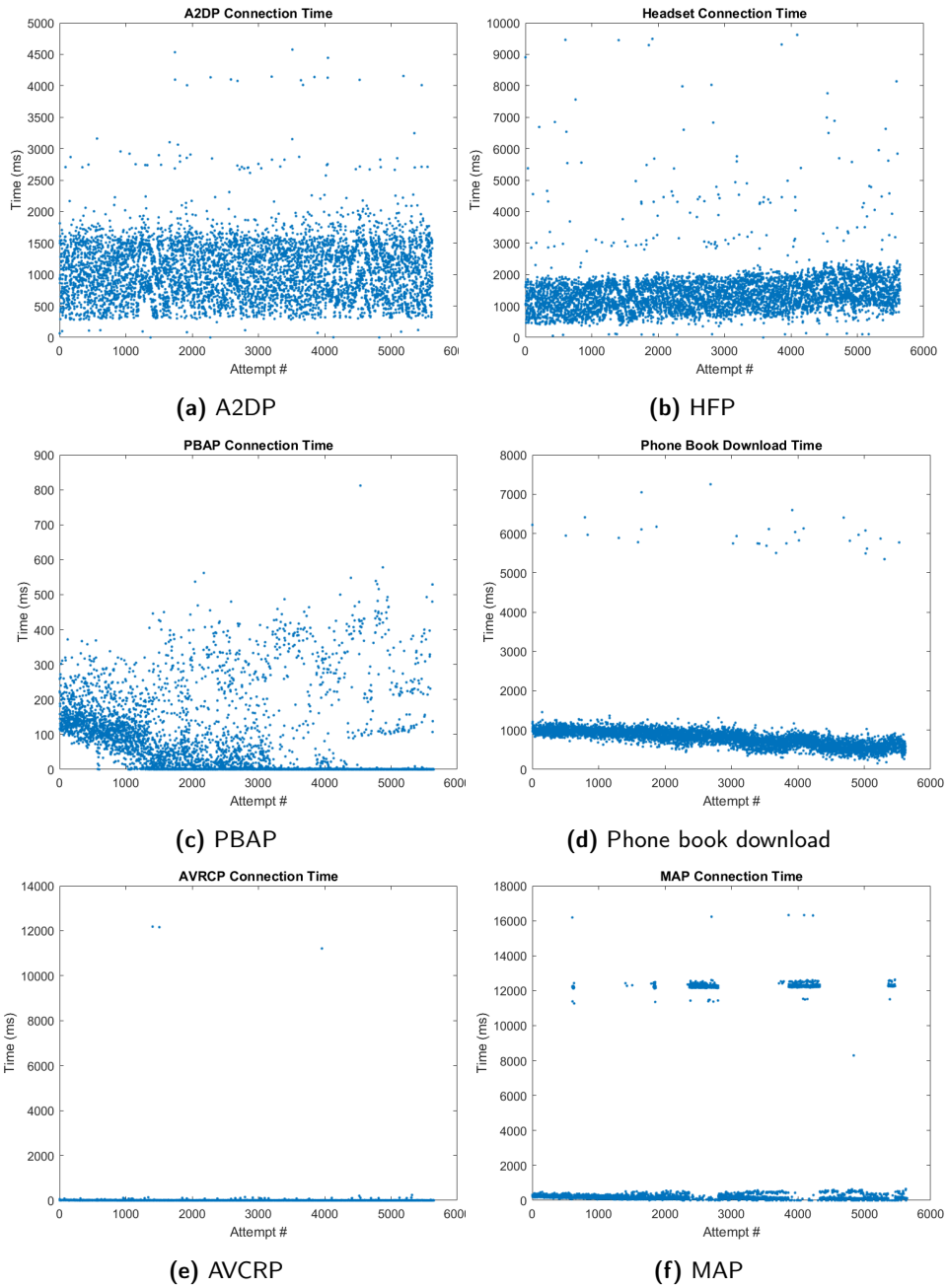
5.2 Initial test results

The measurement results from the initial test can be seen in Figure 5.1 with the mean value and standard deviation in table 5.1. As seen in the figure, the different profiles behave differently over time. The A2DP is fairly consistent, with most measurements within the range 500 to 1500ms. HFP is similar to A2DP and stays around 500 to 2000ms, however HFP seems to rise slightly after around 4000 measurements and on-wards.

The opposite goes for PBAP which has a quite a substantial downwards trend and the phone book download time which has a slight decrease over time. PBAP starts out at around 200ms but after around 1000 measurements the connection times deviates wildly and the majority measures at 0ms. This was likely caused by concurrency issues. Other connection time measurements were likely blocking the start timer of PBAP, making the timer start too late and thus measured incorrectly. For the higher valued outlier the stop timer was likely blocked. The phone book download time measurements are very concentrated, apart from the odd outlier being 6-7 times as high as the mean. AVRCP is the quickest, having a mean value of 9.3ms, and very consistent except a few extreme outliers around 12000ms. MAP seems to be gathering around a few levels which are at 16000ms, 12000ms and under 1000ms. These strange behaviours were also concluded to be an effect from concurrency issues as presented in section 4.7.

Measurement	Mean (ms)	Standard Deviation (ms)
PBAP	68.9	103.9
A2DP	1064.1	464.4
HFP	1391.6	695.7
MAP	856.9	4669
AVRCP	9.3	273.4
Phone Book Download	818.8	432.3

Table 5.1: Statistics from the initial test



(a) A2DP **(b)** HFP

(c) PBAP **(d)** Phone book download

(e) AVCRP **(f)** MAP

Figure 5.1: Measurement from the initial test using Phone B

5.3 Profile results

The measurements of profile connection times from the final test, after concurrency issues were resolved, are shown in figures 5.2 and 5.3 and table 5.2. The results from the phone book download time measurements are shown in Figure 5.4 and table 5.3.

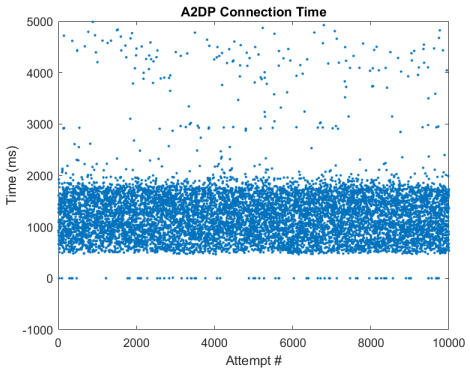
There is no tendency for increase or decrease in mean value over time for any profile connection for either Phone A or Phone B. The mean values, standard deviations and fail percentage differ a bit between the two phone models. In table 5.2 the differences are clear. Phone B is quicker to connect for profiles PBAP, A2DP and AVRCP and slower for MAP and HFP. The most significant difference in mean value between the two models comes from MAP, where Phone B takes approximately 42% longer to connect. The standard deviation is higher for Phone B compared to Phone A for all profiles except AVRCP. The differences are fairly small however for A2DP, HFP and AVRCP. Once again MAP constitutes the biggest difference with almost double standard deviation for Phone B compared to Phone A. The fail percentages for the profiles are in general low, all profiles for both phone models have fail percentages lower than 1% when trying to establish a connection. Phone A has lower fail percentages compared to Phone B for all profiles. A2DP has the highest fail percentage for both models, with 0.7% for Phone B and 0.64% for Phone A.

For both test phones AVRCP is a lot faster to connected compared to the rest of the profiles, with most values ranging from 1 to 20 ms. MAP and PBAP behave quite similarly in their connection times, the vast majority of measurements are in the range 100 to 300 ms. HFP is the slowest profile to connect for both phones. A2DP is a bit quicker to connected compared to HFP but behave similarly apart from HFP having more extreme outliers.

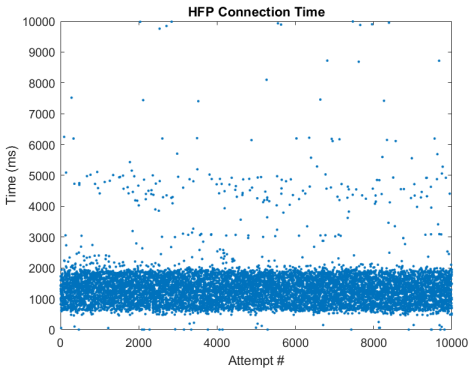
The measurements on Phone B and the MAP profile showed some outliers measuring above 20 seconds, these were removed from the results as these measurements should be viewed as faulty. Since the Bluetooth adapter turns on and off every 15 seconds, the measurements should not exceed this mark and not be included in the mean and standard deviation of connection times.

Device	Measurement	Mean (ms)	Standard Deviation (ms)	Fail Percentage (%)
Phone A	PBAP	174.15	46.54	0.16
	A2DP	1237.7	535.34	0.64
	HFP	1351.2	678.96	0.16
	MAP	152.33	34.76	0.16
	AVRCP	8.69	5.42	0.16
Phone B	PBAP	141.1	70.48	0.28
	A2DP	1223.7	579.46	0.7
	HFP	1410.4	754.08	0.28
	MAP	217.61	67.1	0.28
	AVRCP	6.673	3.36	0.28

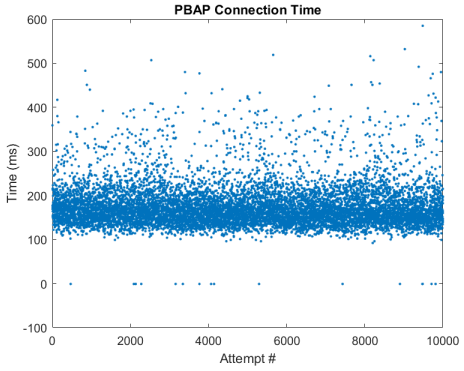
Table 5.2: Statistics of profile connection times from the final test



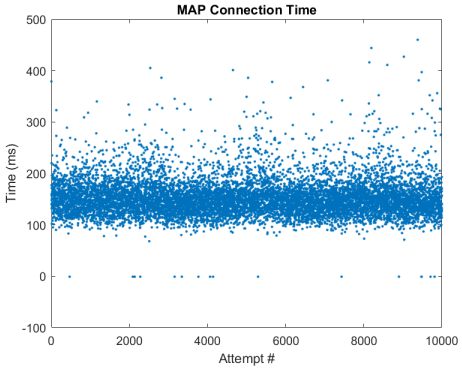
(a) A2DP



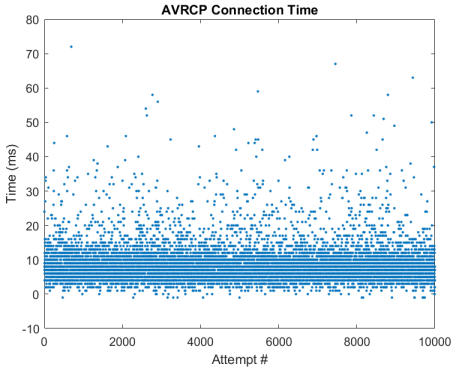
(b) HFP



(c) PBAP

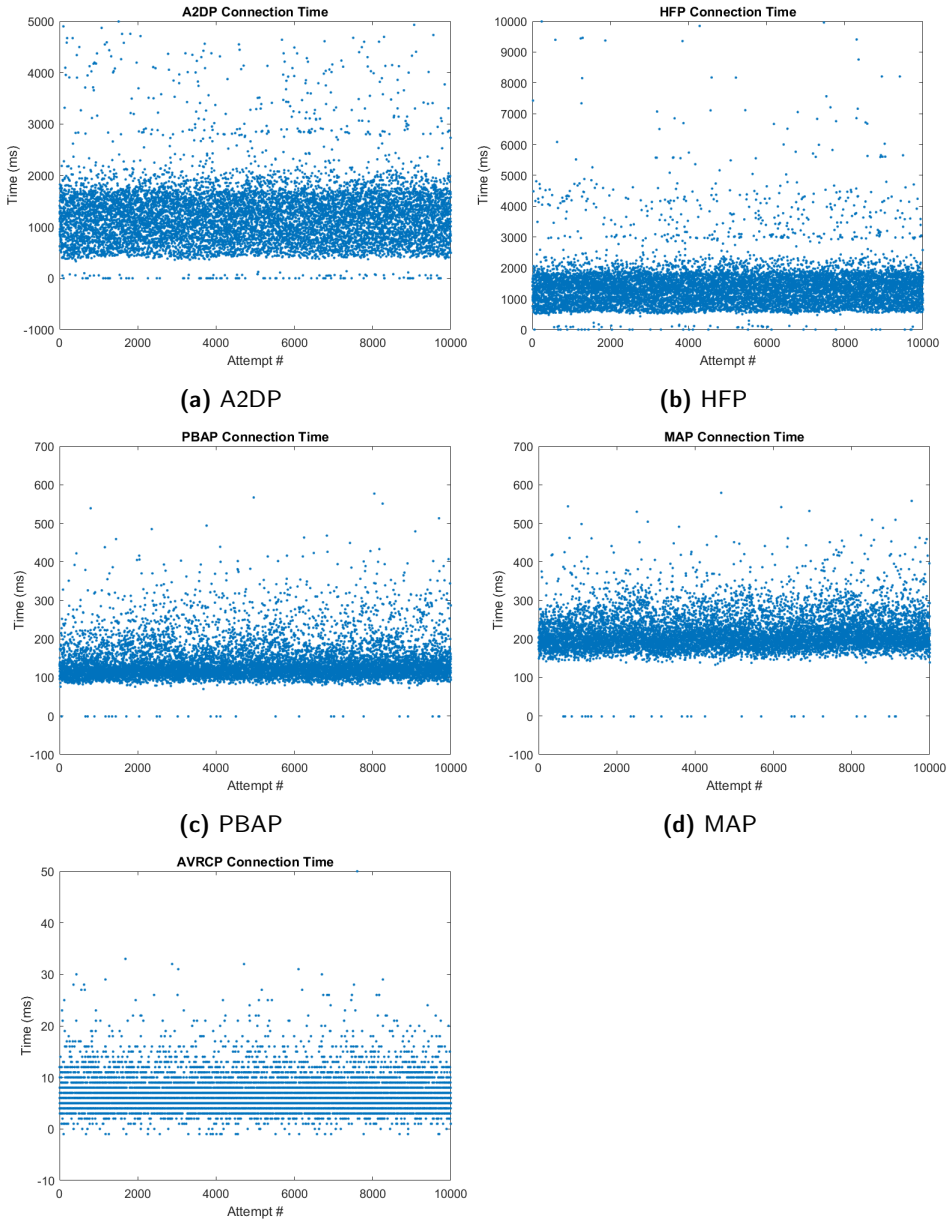


(d) MAP



(e) AVRCP

Figure 5.2: Measurement from final test with Phone A



(a) A2DP

(b) HFP

(c) PBAP

(d) MAP

(e) AVRCP

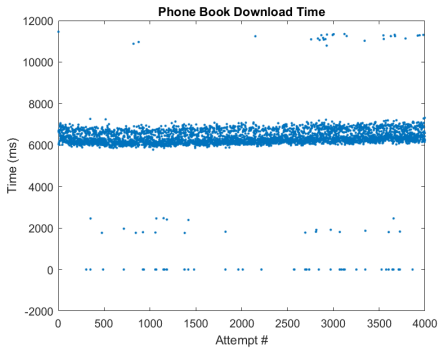
Figure 5.3: Measurement from final test with Phone B

5.4 Phone book results

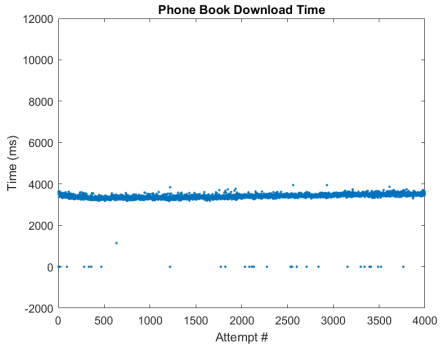
The phone book download time varies more between the two phone models. The mean value of Phone B is 6378.1 ms, almost doubling that of Phone A. The standard deviation of Phone B is 585.52, this is almost 6 times the amount of Phone A. The difference in standard deviation is easily seen in Figure 5.4. The graphs shows that the measurements for Phone B contain outliers, both lower and higher than the mean, not present in the measurement from Phone A. In addition to the outliers, the values close to the mean are more spread out for Phone B compared to Phone A. The fail percentage is also higher for Phone B, although the difference is not as significant as for the other metrics. Both devices have fairly low fail percentage, with Phone B at 1.23% and Phone A at 0.9%.

Device	Measurement	Mean (ms)	Standard Deviation (ms)	Fail Percentage (%)
Phone A	Phone Book Download	3410.8	98.472	0.9
Phone B	Phone Book Download	6378.1	585.52	1.23

Table 5.3: Phone book download times from the final test



(a) Phone B

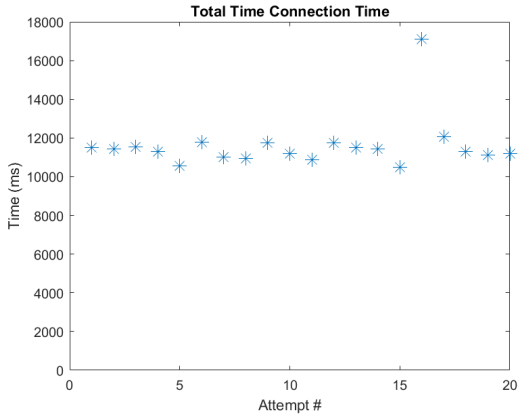


(b) Phone A

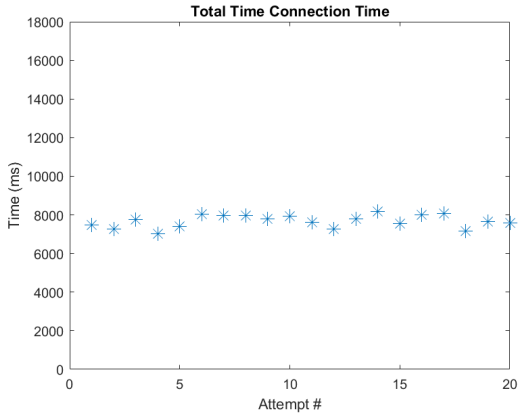
Figure 5.4: Measurements from final test of phone book download time comparing Phone A and Phone B

5.5 Total time results

The total time measures, on average, slightly under 8000 ms for Phone A and slightly under 12000 ms for Phone B, with the exception of one outlier of around 17000 ms for Phone B, see Figure 5.5. This time difference of around 4000 ms matches the average difference in the phone book download time between the two models quite well.



(a) Phone B



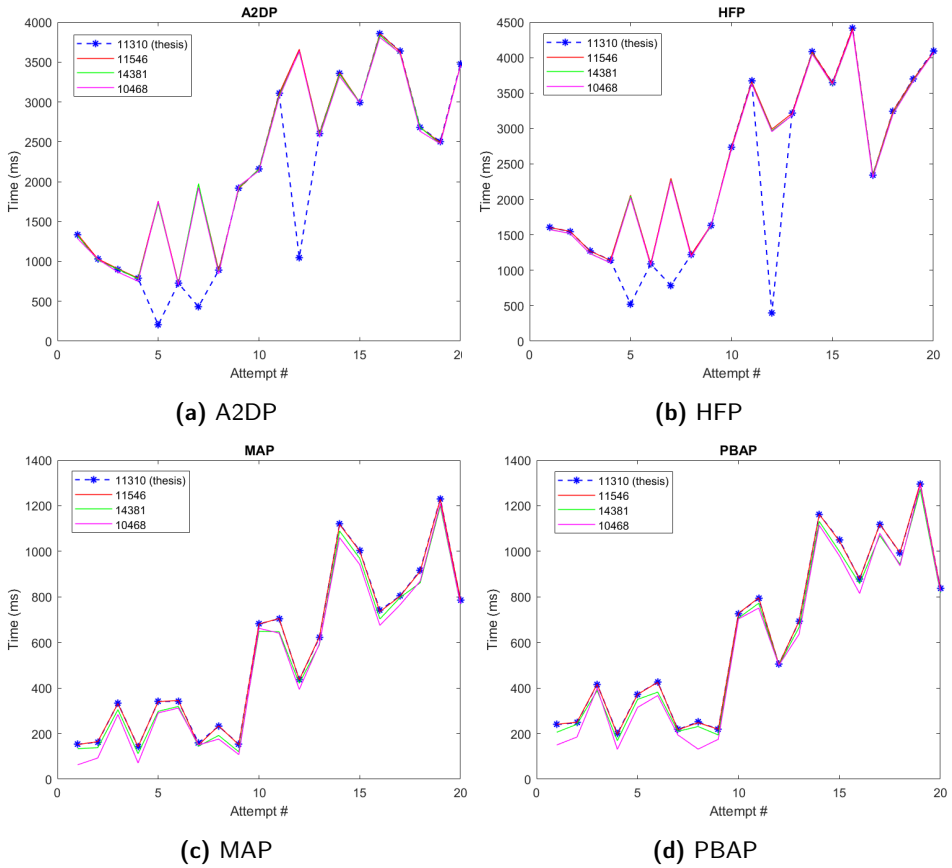
(b) Phone A

Figure 5.5: Measurements of total time comparing Phone A and Phone B

5.6 Broadcast reception

The results from the broadcast reception test can be seen in Figure 5.6 and table 5.4. The largest deviation is seen at 5th, 7th and 12th measurement for A2DP and HFP. These are substantial differences, in all three cases the reception of the broadcast intent, showing the profiles connecting, are caught by our process several seconds later compared to the three other processes that are listening to the same intents in the system. Figure 5.7 illustrates this, showing a cut-out of logs from the 12th measurement. The logs from our process are surrounded by the green box and newProfileState 1 means that the state of the profiles have changed to connecting. HFP and A2DP are caught more than 2 seconds later for our process compared to the others.

For the rest of the measurements, of HFP and A2DP, the reception is quite consistent for the different processes and for PBAP and MAP the processes behave similarly for all measurements.



(a) A2DP **(b)** HFP

(c) MAP **(d)** PBAP

Figure 5.6: Measurement from broadcast reception test

Measurement #	A2DP	HFP	MAP	PBAP
1	55	31	90	91
2	17	33	72	65
3	42	40	49	28
4	40	38	71	69
5	1553	1534	50	58
6	23	31	33	59
7	1543	1514	14	24
8	47	30	58	119
9	30	13	45	44
10	24	32	32	25
11	53	41	64	45
12	2612	2591	42	4
13	40	31	35	57
14	49	31	60	48
15	4	32	61	69
16	49	37	66	64
17	33	34	39	50
18	51	43	54	57
19	31	37	32	27
20	19	26	22	19

Table 5.4: The maximum difference of connecting time of the four different broadcast receivers in the system.

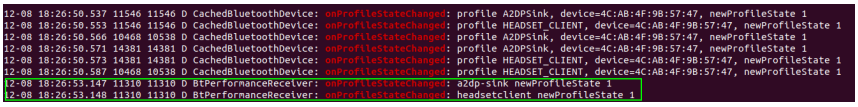


Figure 5.7: Cut out of the logs from the 12th measurement of the broadcast reception test

Chapter 6
Discussion

In this section the different parts of the project are discussed. The choice of method and testing parameter is explored and the sources of error for the final test are discussed. The reasons why the results vary for the initial compared to the final test is examined.

6.1 Test design

In the initial phase of the project, different ways of designing the test were considered. Several testing parameters and methods were examined and in the end the test was developed in DID, measuring connection times utilizing broadcast receivers.

6.1.1 Parameter

There are a few reasons for picking connection time over any of the other testing parameters discussed in this thesis. The focus of the test was the user perspective, and thus the connection time was chosen since it represents the waiting time a user has to experience when connecting a device to the IHU.

One could argue that throughput, latency and device discovery all are interesting user experience perspective, which they are. However, throughput was ruled out due to the capacity of Bluetooth throughput being more than enough for streaming music, with music being the function with the highest demand for in a car environment. It would perhaps become of greater value to measure throughput if Bluetooth is used for a more demanding transfer usage in the future.

Device discovery was omitted because of the rarity it has in comparison to the other parameters. A normal use case would only discover a new device once in a while, while an auto connection of an already paired device occurs far more often.

Latency and connection times were both interesting testing parameters. But ultimately connection time was chosen due to the convenience of implementation with broadcast receivers to set timestamps.

6.1.2 Method

The method of implementing the test in DID was chosen for a couple of reasons, it was a reliable way to move forward due to the amount of control over the test you have by setting your own timestamps, storing the data and choosing your output format. There is also a big potential for expansion with this method. As discussed in section 8 there is a lot of additions which can be done, and when run in vehicles, a large amount of data to be collected and analysed.

The method of using Dumpsys for this test was not chosen due to the performance impact it would have on the test. Dumpsys provide a lot of system information and diagnostic which, when collected, would slow down the system if repeatedly called to record the connection times. Perfetto was a very useful tool, providing a substantial amount of detailed information, but was more aimed to analyse single recordings, than handling larger amount of data which was the aim in this thesis.

Firebase was better suited for this, but did however have an approach more focused on the Android applications than the embedded side of the operating system. It may be possible to collect recordings, but in the end it was omitted due to complexity. Another complex method would be to implement a method with Wireshark and BTmon. It would provide valuable information on the physical signal, but has the downside of not having a good way of extracting the data and interpreting the communication, e.g. if it's regarding HFP, MAP or PBAP in the Bluetooth packets.

6.2 Sources of error

As table 5.4 shows, the time of reception of broadcast intents can vary quite a lot. The whole test is based on measuring time from broadcast intents, hence this is a big source of error. However, while these times differ substantially, the overall trends and behaviours of the connection times are caught, especially when running the test multiple times. While a single measurement is untrustworthy, the mean of hundreds of measurements is a lot more useful.

In section 5.3 outliers in the MAP measurements of Phone B were mentioned. These outliers all measured 20 seconds longer than the rest of the measurements. The cause is most likely that the broadcast intent for *connected* wasn't caught until the next measurement which is an unexpected result. While these results are a cause for concern and deserves looking into in future development, this is a not major issue as it did not occur for any other profile nor any profile in the Phone A measurements. Furthermore the outliers can easily be removed from the results and should not be an issue for measurement in cars since there will not be any specific time limit in this case.

The results presented in this thesis should not be viewed as a representation of connection times in a car, but rather a simulation of it. There are more uncertainties and possible sources of interference present in a real world scenario compared to the lab environment. The measurements made in this report was done in the same controlled fashion each time, there is a high likelihood that measurements made in a car would differ from the results in this report.

6.3 Test improvements

The results from the initial test compared to the final are significantly different. When comparing the graphs from the two tests, it is clear that the results from the final test are more consistent, contain less outliers and have no drift in mean value. This improvement is due to major changes done to the test. In the initial test the phone book download time started as soon as PBAP was connected. When waiting for the phone book to finish it's download, it blocks the profiles timestamps to get registered. This means that any profile connecting slower than PBAP will not be registered as connected until the phone book download is finished, and subsequently the entire delay of the thread will be added to the connection time of the profile. In the final test this problem is handled by starting the phone book thread after all profiles have been registered in a separate thread.

Another potential cause for the inadequate results of the initial test is the process of how the measurements were saved. In the initial test, as soon as a new connection time was measured, the stored JSON file was loaded, updated and saved. This means that during one measurement attempt the file was updated six times. As long as the stored file is small, this is not a problem, as loading and saving is a quick process, but as the file grows larger this takes longer and longer. In the long run this delay has the risk of affecting measurement times. To eliminate this problem, the file is updated for all measurements after the Bluetooth adapter has been turned off, together with having the saving and loading done in a isolated thread.

Another major difference between the initial and final test is the implementation of adding the placeholder value of -1 as a measurement for failed connection attempts. This did not only add the possibility of measuring fail percentage, an important statistic for Bluetooth performance evaluation, it also made sure all profiles had the same number of measurements, this is important for evaluating how one profile might affect another. It also ensured there were no left over timestamps, saved from the previous measurement, affecting new measurements. In the initial test if the Bluetooth adapter was restarted before a profile had time to connect but the timestamp for connecting was saved, the next measurement would not be accurate as the time difference calculated as the intent for connected is caught will be the summation of two measurement attempts.

The way the test deems a profile connection a failure can be viewed as questionable. The time between turning the Bluetooth adapter on and off is effectively the maximum time a profile has to connected before it is determined as a failed connection. In a lab this is not a problem, as this waiting time can be changed to one's liking. Using 15 seconds for our test was a trade-off between collecting as much data in a short time period as possible while catching the vast majority of profile connections.

6.4 Measurement results

In general both phones measured in this thesis behaved pretty similarly, their fail percentage were very comparable, as well as all mean values. Their graphs also seem to behave in the same way, with no big deviations or anomalies. The phone book download time did however differ quite substantially, with a average quicker time for Phone A of approximately 3 seconds. The slower phone book download time also contributed to a slower total time for Phone B.

Chapter 7
Conclusion

The primary objective of this thesis is to bring forward a method approach together with a test to evaluate the Bluetooth performance when communicating with the Android infotainment unit from Volvo Cars. In the thesis several methods and parameters are presented and discussed to conclude what approach is the most suitable for the data and information wanted.

It was determined the best method was to implement a test locally on the Infotainment head unit which recorded the time measurements. The final test measures the connection times for a selection of profiles, the time it takes to download the phone book and the total time for the phone to establish the connection with the vehicle. Connection times and download time was chosen due to it was a good representation of the user experience when connecting to the vehicle via Bluetooth. To implement the test locally on the infotainment head unit was mainly because of two reasons, the first being the amount of control it gives over data format and timing measurements, secondly due to the possibility of further expansion and additions.

The test is designed to be able to gather data from customers for long periods of time. This enables Volvo to see if there is any difference in performance over time or if changes such as software updates affect the performance. The test is not designed to get any deeper information about Bluetooth and what might be going wrong during a connection. However, if there is a connection problem, analysis of the collected data can give a first clue into what to look for. It should be seen as a tool to collect data for statistical analysis, which by extension gives information on the Bluetooth performance.

Tests were run in a lab environment where two different phone devices were tested, in this thesis called "Phone A" and "Phone B". The tests concluded that all the measured profiles, A2DP, HFP, PBAP, MAP and AVRCP performed similarly. Furthermore, the error rate of failed connection establishments and graphs of the data was comparable. However, the phone book download time was approximately 3s quicker on average with Phone A. This also made Phone A have a quicker total time measurement.

There are some sources of error such as inconsistencies in the reception of broadcast intents, which are presented and discussed. However the results presented in this thesis should not be viewed as a representation of connection times in a vehicle, but rather a simulation of it.

In conclusion, the test is an example of a tool that collects data on Bluetooth

performance in a car environment. The data is useful in evaluating how well the connection establishment works from phone to car. It can be used to evaluate differences between car and phone models. While a single measurement is not necessarily reliable, the mean of several measurements can be trusted and used for statistical analysis. The test has only been used in a lab environment and needs further testing in cars before it can be implemented in customer vehicles.

Chapter 8
Future Work

This section discusses future improvements, additions and further work that could be done with the method described in this thesis. It covers how test could be set up to provide more informative data both on the software and the hardware level.

8.1 Method

There are several areas of interest the method described in this thesis could be expanded to, one being to make a systematic test comparing different Android and iOS versions on a device. It would be very interesting to run tests with two identical devices, but with different operating system versions. If repeated with several kind of cell phone distributors running Android such as Samsung, Sony and Nokia, it will thus eliminate the hardware factor in the test. With this kind of testing it would give an insight in how much the actual OS version affect the Bluetooth performance, and if e.g. Android 10 has better performance in comparison to Android 7. This would determine if outdated operating systems are more prone to delays in connection establishment, which could create a new group of devices to focus the testing further on.

Another similar approach is to set up several Android devices and compare them with iOS devices and run tests systematically to compare the operating systems. This setup would naturally need to compare similar versions of the operating systems to be comparable. This would show if there is any significant difference in Bluetooth performance between the operating systems together with giving an indication if there are any behavioural differences. With this kind of setup it could be investigated if embedded Android infotainment systems have a greater Bluetooth performance together with Android devices or not.

Another interesting test would be to see what impact contacts and contact photos have on the phone book download time. A systematic test of this could show if the relation of the download time and the amount of contacts is linear, or if it has another behaviour. In the same manner, it would be of great value to understand if the photos have a substantial impact on the time it takes to download the entire phone book.

The method presented could also be used for automatic testing when the software of the IHU is updated, which could provide feedback to see if the new software update is affecting the Bluetooth performance in any way or not at all.

8.2 Hardware

The method could be adapted to other hardware configurations. It could be an addition of devices or using vehicles to test how different values are affected by different hardware factors.

8.2.1 Devices

Future work with different hardware combined with the method in this thesis would be to start comparing different cell phone vendors. This could for instance be a setup to compare models from each vendor, and do a comparison to identify if any specific vendor has a poor Bluetooth performance compared to others. This could either be made with a mix of models within the cell phone vendor, or with each vendor's flagship model, but would require a larger amount of devices to conclude what the difference between the vendors are.

With a similar approach another extension of this method would be to compare models within each individual vendor. This could answer if there is any significant difference between newer devices or outdated ones. It may be the case that there is no difference due to Bluetooth being a long time established technology, or it might be that outdated devices are the most prone to create extensive delays when connecting to the infotainment unit.

With this kind of testing, in combination with the knowledge of the most used devices, the focus could be shifted to the device or devices that have the worst Bluetooth performance and are common among users.

8.2.2 Vehicles

To further work on the method provided in this thesis and to add data to the one collected, testing in real world vehicles would give valuable information on what the actual connection establishment times are. This would provide a more accurate model on what the user experience is when using Bluetooth with the infotainment unit. This could initially be done in the co-dev cars to collect data, and further expanded to customer vehicle to amass a large amount of info from different parts of the world

Comparing BEV-cars with Non-BEV cars would give an insight in how the two differs when trying to auto connect with the phone, as well as give a metric on how large the time difference is. If combined with an estimated time it takes to reach the correct usage mode for the vehicle, it would give an indication on how much the sensor in the seat of the BEV-cars changes the experienced time it takes when getting in the driver's seat.

8.2.3 Analyzing equipment

A different method approach which could complement this work, would be to have a set up which can analyse the connection establishment traffic on the physical layer. To be able to listen and analyze the Bluetooth traffic a Bluetooth analyzer, would be of great help. One example of this kind of equipment would be a Frontline® Sodera® Wideband Bluetooth® Protocol Analyzer [26]

This kind of equipment would make it able to analyse and measure the time the connection process spends in the physical layer. With that, the amount of time spent above the physical layer in the connection stack, could also be calculated. This would provide another piece in the puzzle in figuring out all the steps for the whole timeline when connecting a device to the IHU.

References

[1] SBD, "2025 every car connected: Forecasting the growth and opportunity," GSMA mAutomotive, pp. 1–28, February 2012, (Date last accessed 20-September-2021). [Online]. Available: https://www.gsma.com/iot/wp-content/uploads/2012/03/gsma2025everycarconnected.pdf

[2] "Our story vcc," Volvo Cars, 2022. [Online]. Available: https://www.volvocars.com/intl/v/our-story

[3] Bluetooth®SIG, "Bluetooth technology overview," (Date last accessed 15-July-2014). [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[4] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pp. 1–1076, 2007, table 18-9.

[5] A. Mourad, S. Muhammad, M. O. Al Kalaa, H. H. Refai, and P. A. Hoeher, "On the performance of wlan and bluetooth for in-car infotainment systems," *Vehicular Communications*, vol. 10, pp. 1–12, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209617300669

[6] J. I. N. Hipolito, N. C. Arballo, J. A. Michel-Macarty, and E. J. Garcia, "Bluetooth performance analysis in wireless personal area networks," in *2009 Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, 2009, pp. 38–43.

[7] C. Bisdikian, "An overview of the bluetooth wireless technology," *IEEE Communications Magazine*, vol. 39, no. 12, pp. 86–94, 2001.

[8] D. Kammer, G. McNutt, B. Senese, and J. Bray, "Chapter 2 - exploring the foundations of bluetooth," in *Bluetooth Application Developer's Guide*. Burlington: Syngress, 2002, pp. 69–102. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781928994428500057

[9] "Core specification 4.2 vol 1," Bluetooth®SIG, pp. 97–99, 43–44, December 2014. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-4-2/

[10] "Advanced audio distribution profile 1.2," Bluetooth®SIG, April 2007. [Online]. Available: https://www.bluetooth.com/specifications/specs/a2dp-advanced-audi%e2%80%8b%e2%80%8bo-distribution-profile-1-2/

[11] "A/v remote control profile 1.5," Bluetooth®SIG, July 2012. [Online]. Available: https://www.bluetooth.com/specifications/specs/avrcp-a-v-remote-control-profile-1-5/

[12] "Hands-free profile 1.5.1," Bluetooth®SIG, January 2019. [Online]. Available: https://www.bluetooth.com/specifications/specs/hands-free-profile-1-5-1/

[13] "Message access profile 1.1," Bluetooth®SIG, July 2013. [Online]. Available: https://www.bluetooth.com/specifications/specs/message-access-profile-1-1/

[14] "Personal area networking profile 1.0," Bluetooth®SIG, February 2003. [Online]. Available: https://www.bluetooth.com/specifications/specs/personal-area-networking-profile-1-0/

[15] "Phone book access profile 1.2.3," Bluetooth®SIG, January 2019. [Online]. Available: https://www.bluetooth.com/specifications/specs/phone-book-access-profile-1-2-3/

[16] "Audio quality," Spotify. [Online]. Available: https://support.spotify.com/us/article/audio-quality/

[17] "Dumpsys," Google. [Online]. Available: https://developer.android.com/studio/command-line/dumpsys

[18] "Android debug bridge," Google. [Online]. Available: https://developer.android.com/studio/command-line/adb

[19] "Perfetto - system profiling, app tracing and trace analysis," Google. [Online]. Available: https://perfetto.dev/docs/

[20] "Firebase," Google. [Online]. Available: https://firebase.google.com/

[21] G. B. Ulf Lamping, "Wireshark developer's guide," Wireshark. [Online]. Available: https://www.wireshark.org/docs/wsdg_html_chunked/

[22] "Broadcast," Google. [Online]. Available: https://developer.android.com/guide/components/broadcasts

[23] "Intents and intent filters," Google. [Online]. Available: https://developer.android.com/guide/components/intents-filters

[24] "Broadcastreceiver," Google, 2021-06-09. [Online]. Available: https://developer.android.com/reference/android/content/BroadcastReceiver

[25] "Contact provider," Google. [Online]. Available: https://developer.android.com/guide/topics/providers/contacts-provider

[26] "Frontline® sodera™ wideband bluetooth® protocol analyzer," Teledyne LeCroy, 2022. [Online]. Available: https://fte.com/products/sodera-protocols.aspx

Appendix A
Glossary

ADB - Android Debug Bridge
AFH - Adaptive Frequency Hopping
A2DP - Advanced Audio Distribution Profile
AES-CCM - Advanced Encryption Standard-Counter with CBC-MAC
AG - Audio Gateway
AVRCP - Audio/Video Control Profile
BEV - Battery Electric Vehicle BR - Basic Rate
BLE - Bluetooth Low Energy
BTmon - Bluetooth Monitor
CSD - Center Stack Display
Co-Dev - Co Developer
CRC - Cyclic Redundancy Check
DID - Data IDentifier
DQPSK - Differential Quadrature Phase-Shift Keying
EDR - Enhanced Data Rate
FTD - File Time Delay
GFSK - Gaussian Frequency-Shift Keying
GAP - Generic Access Profile
HFP - Hands-Free Profile
HF - Hands Free unit
ISM - Industrial, Scientific and Medical
IHU - Infotainment Head Unit
L2CAP - Logical Link Control Adaptation Layer
LT_ADDR - Logical Transport Address
LVDS - Low-voltage Differential Signalling
MAP - Message Access Profile
MCE - Message Client Equipment
MIC - Message Integrity Check
MSE - Message Server Equipment
PANU - Personal Area Network User
PBAP - Phone Book Access Profile
PAN - Public Area Networking Profile
SPP - Serial Port Profile
SIG - Special Interest Group
WLAN - Wireless Local Area Network

Printed by Tryckeriet i E-huset, Lund 2022

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-875
http://www.eit.lth.se