CHALMERS
UNIVERSITY OF TECHNOLOGY

Design of a seat-based wireless ticketing system for an efficient passenger control process

Bachelor's thesis in Computer Science and Engineering

DENNIS BROQI, HELÉNE JARL, ANDREAS JOHANSSON,
CHRISTIAN KRIŽAN, HAMODI MANSOUR,
LUKAS ÖGNELOD

CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Department of Computer Science and Engineering
Gothenburg, Sweden, June 2016

"Ticketing" by *Sara Hindawi* (2016)

Design of a seat-based wireless ticketing system
for an efficient passenger control process

Dennis Broqi
Heléne Jarl
Andreas Johansson
Christian Križan
Hamodi Mansour
Lukas Ögnelod

Supervisor: Lennart Hansson, Research engineer [(1)]
Examiner: Arne Linde, Associate professor [(1)]

[(1)] Computer Engineering division, department of Computer Science and Engineering

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

Front-page illustration:
"Ticketing" by Sara Hindawi, 2016

Typeset in LaTeX
Gothenburg, Sweden 2016

Abstract

Design of a seat-based wireless ticketing system for an efficient passenger control process

Innovative Projects Sweden AB has commissioned the development of a prototype in order to investigate if there is a market for a new kind of ticketing system. A train-based ticketing system, where each seat handles its own passenger check-ins. The aim of this Bachelor's thesis is to create a prototype of the system, which could improve the process of validating tickets on trains by reducing train attendant workload and by focusing attendant work tasks on illegal passengers. This report shows how this prototype for validating tickets is developed. The prototype is built with components from Arduino, Adafruit and Digi. A hand-held device in the form of an Android smartphone is used as an interface for the train attendants to allow for an efficient and easy way to monitor the train and the passengers. The prototype communicates in two ways, firstly via the Zigbee protocol and then via Bluetooth to the smartphone. This thesis results in a prototype that meets the goals of the project. In conclusion, a system like this could give train attendants the added functionality that is requested by the thesis. The system also provides added security when validating passenger ticket credentials.

Dennis Broqi, Heléne Jarl, Andreas Johansson,
Christian Križan, Hamodi Mansour, Lukas Ögnelod

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Gothenburg, Sweden, June 2016

Keywords: ticketing, public transport, Zigbee, XBee, NFC, Bluetooth, seat unit, Android, Arduino

Sammanfattning

På uppdrag av Innovative Projects Sweden AB kommer detta kandidatarbete att behandla utvecklingen av en prototyp för ett nytt biljettsystem specifierat för tåg. Kandidatarbets syfte är att skapa en prototyp som förenklar och effektiviserar verifieringen av tågbiljetter. Rapporten kommer att gå igenom utvecklingen av ett sådant system, där kompontenter från Arduino, Adafruit och Digi används för realisera prototypen. Till systemet ingår även en applikation för smarta mobiltelefoner, vars syfte är att möjliggöra ett effektivt och smidigt övervakningsmedel för tågvärdar. Prototypen kommunicerar på två olika sätt, dels via Zigbeeprotokoll och sedan via Bluetooth till mobilapplikationen. Rapporten resulterar i en prototyp som uppfyller målen för projektet. Sammanfattningsvis ger det utvecklade systemet tågvärdar de medel som krävs för att underlätta biljettverifiering på tåg, vilket i sin tur leder till en ökad säkerhet i processen.

Dennis Broqi, Heléne Jarl, Andreas Johansson,
Christian Križan, Hamodi Mansour, Lukas Ögnelod

Chalmers Tekniska Högskola
Göteborgs Universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, juni 2016

Nyckelord: Biljettsystem, kollektivtrafik, Zigbee, XBee, NFC, Bluetooth, sätesenhet, Android, Arduino

Acknowledgements

We would like to give our special thanks to Chalmers, our supervisor Lennart Hansson, examiner Arne Linde and Innovative Projects Sweden AB for giving us the opportunity to realise this project.

This page intentionally left blank

Contents

Glossary

Table 1: Technical glossary for terms and acronyms used in this report

Term	**Written out**	**Brief explanation**
Android		Operating system mainly used in mobile devices
Arduino		Open-source microcontroller device, company and project. All aimed at providing a simple hardware development process
AVR	Advanced Virtual RISC	Series of microprocessors manufactured by Atmel
Bluetooth		Wireless communication standard
DBMS	Database management system	Computer software application for creating and managing databases
ECMA	European Computer Manufacturers Association	Organisation behind the ECMA-340 standard used by the NFC readers in this project
EEPROM	Electrically Erasable Programmable Read-Only Memory	Non-volatile memory storage type
ER-diagram	Entity-Relationship-diagram	An overview of the relationships in the database
Firmware		Embedded control program within a device
GPIO	General-purpose input/output	Generic pin on an integrated circuit
IDE	Integrated development environment	Software used to assist in software development
IEEE	Institute of Electrical and Electronics Engineers	Organisation which maintains the Bluetooth standard and the Zigbee protocol relevant for this project
IoT	Internet of Things	Term used to describe uplinked device interconnectivity

Table 2: Continued technical glossary

Term	**Written out**	**Brief explanation**
ISO	International Organization for Standardization	Organisation which maintains the standards ISO 18092 and ISO 14443 which describes the functionality of the NFC readers used in this project
Java		Object-oriented programming language
MariaDB		A database management system
Mockup		Draft of a prototype
MySQL		A database management system similar to MariaDB
NFC	Near field communication	Set of multiple protocols for wireless data transfer between two devices within a few centimetres
PAN	Personal area network	A network with limited range
RF	Radio frequency	Emitted electron oscillations per unit of time
Shield		Hardware peripheral designed to fit the Arduino printed circuit board form factor
Smart card		Used in this report to denote contactless cards which include circuitry and memory and supports the NFC standard
SPI	Serial Peripheral Interface (bus)	Synchronous serial communication bus used in embedded applications
TFT	Thin-film transistor	Used in this report to denote the liquid-crystal display technology based on thin-film transistors
TTL	Transistor-transistor logic	A typeset of digital circuitry, logic is done by bipolar transistors

Table 3: Continued technical glossary

Term	**Written out**	**Brief explanation**
UART	Universal asynchronous receiver/transmitter	Used in this report to denote a circuit for serial communication
(U)UID	(Universally) unique identifier	Identifier standard used in software construction
XBee		Zigbee RF module by Digi Inc.
Zigbee		Wireless communications protocol mainly used for interconnecting devices in a personal area network

Term	**Explanation**
Seat unit	Machine placed at every seat. Used by passengers to check in at their seats
Radio unit	Interface unit that links the train attendant's smartphone to the personal area network via Bluetooth
Database access unit (DAU)	Unit which acts as the network coordinator and butler for all database access requests from the seat units and attendant smartphones
Database	Not to be confused with the *Database access unit*, which is the device used for accessing the database content from the Zigbee personal area network

Table 4: Important project-specific glossary. These terms constitute our own system vocabulary and are not necessarily generally accepted abbreviations.

Chapter 1

Introduction

Trains have been one of the most common means for transportation since industrialisation. Throughout years of technological development, trains have undergone major improvements. Yet the tools for validating passengers have not changed substantially. In many trains, travellers are still supervised and validated physically by train attendants in an old fashioned manner. It can be assumed that this method of validating passengers may result in errors as the train attendants need to memorise which passengers have been validated or not.

Innovative Projects Sweden AB has commissioned this project to investigate if there is a market for a product that aids train attendants and passengers in public transportation by mandating the development of a ticketing system prototype. The prototype will demonstrate the concept of a system where passengers check in at a seat unit and where a smartphone application will monitor each train seat in order to let train attendants control the seat activity.

The project's prototype development will take place in two main development cycles which results in two prototype version iterations. The first iteration, the *In-development* prototype, will consist of a prototype where the most rudimentary system functions will be achieved. Then, a revised *alpha prototype* will be devised in order to achieve a near-finished state of system functionality but without any final designs in place. Such designs can differ broadly depending on an actual commercial implementation of the system and will be left for eventual future work. These two stages will be referred to as the in-development prototype and the alpha prototype respectively.

1.1 Purpose of the project

The purpose of this project is to develop and demonstrate a prototype of an integrated ticketing system for trains. The expected outcome is a system that improves the work environment for the train attendants and provides a possible reduction in stowaways. This includes a scenario analysis on how such a system combats logistical conflicts.

1.2 Goals with the project

The goal is to create a prototype which includes a system of units that could be embedded in a seat of a train car or placed elsewhere in the train car. A smartphone application for train attendants will also be developed, along with a database that contains all information regarding travelling passengers on the train's current route. The system of units consists of a seat check-in unit, a radio unit and a database access unit. The functions of the individual units will further be explained in *2.1 System specification* and in *4 Resulting system*.

1.3 Limitations

Limitations have been made in order to make the project feasible. These limitations are focused on leaving technical implementations open and have been set in order to focus on the ticketing system. Some limitations have been set to focus on the scenario analysis deemed necessary to fulfil the purpose of the project, as well as the problem description.

1.3.1 There will only be one database

The prototype of the ticketing system uses a local database only applicable on the train containing the system. As seen in chapter *5 Discussion*, an actual implementation of this system could include several local databases, one on every train using the system. Because of this, a central database would be required to keep the overall system synchronised. This in turn is beyond scope of this project, and a multi-database synchronisation system is thus omitted as a limitation.

1.3.2 The database will not cross-check NFC card security

As seen in *2.3 Near field communication peripheral: Adafruit PN532*, the alpha prototype ticketing system validates cards using the identity serial. This allows for card duplicates since this number can be pseudo-randomly generated. As this project will be limited to a local database only i.e. one train, there will be no detailed treatment on how to develop a larger cross-checking network in order to detect duplicate or forged NFC cards in the network. This in turn means that every compatible near field communication (NFC) card will be treated as a legitimate ticket unless explicitly treated otherwise. In practice, an NFC compatible system is recommended to include said cross-checking system in order to detect counterfeit duplicates of NFC cards, see *5 Discussion.*

1.3.3 There will be no separate checkout unit

Following the early results of the problem analysis, it was deemed that a system where a passenger is required to perform a check out when reaching their destination is a necessity. This however brings forth a scenario where a passenger has omitted a check out. In order to let the passenger perform a check out in case this happens, thus ending their trip, it was deemed fair to suggest a checkout unit separated from the on-board train system with the sole purpose of signalling to the system that the passenger's trip has ended. This project will however not realise such a unit as the main focus of the project is to relieve workload from the train attendant. The separate checkout unit has thus been omitted as a suggestion for a commercial system implementation, see *5 Discussion.*

1.3.4 The project will not treat a crowded train scenario

This project will not aim to develop the specific functionality required for the seating units in order to process standing passengers. As the system is considered a relieving tool and not an automated replacement for the train attendant, this person is considered available to process standing passengers in a crowded train scenario.

1.4 Related work

A lot of studies regarding the field of ticketing systems have been made. Some of the key words for related work are *transport, NFC, smart cards* and *safety.* The following section will cover a brief summary of four studies regarding ticketing, ticketing data collection, NFC and smart cards.

The report *Near Field Communication: A study on the NFC technology's possible field of application and its development within social media,* by Lisa Hamberg and Christine Hube, discusses some advantages and disadvantages with wireless communication over far distances and short distances [1]. For instance: RFID, Bluetooth, Xbee, QR, contactless smart cards and of course NFC. The report found that NFC can effectively be used in fields such as ticketing systems, and expects the technology to find wide use in the future of wireless communication technology. In 2010, Kerem Ok et al. published the paper *Current Benefits and Future Directions of NFC Services*[2]. The authors discuss the benefits and future usage of the three operating modes for NFC applications: the read or write mode, the card emulating mode and the peer-to-peer mode. According to them, near field communication is an emerging and promising technology that in the future will find use in a vast assortment of areas.

In 2006, Gražvydas Jakubauskas wrote the paper *Improvement of urban passenger transport ticketing system by developing intelligent transport system* [3]. In this paper, Jakubauskas discusses benefits and drawbacks about paper tickets. It is claimed that even though paper tickets fulfil the purpose of validating a paid trip, it is not able to assist in the optimisation of routes. Another big drawback with the paper ticket is that it is not multifunctional. An alternative to paper ticketing can be found in smart card ticketing. Although the concept with smart cards is expensive to install and start to use, there are some benefits to take into consideration. The smart card can in a convenient manner keep track of the passengers boarding and getting off at stations. This information can help with optimisation of routes by using information about the passengers' daily turn. This smart card could also be used for several different tasks, such as the library or parking.

An article that further emphasises the subject of data gathering and multifunctional smart cards was published by M. Bagchi and P.R. White in *Transport Policy*, 2005. In this article, named *The potential of public transport smart card data*, the authors analyses the potential role of data collected by smart card systems as a means of analysing travel behaviour [4]. Just as with the case provided by G. Jakubauskas, the data that can be collected from smart card travelling is useful from a statistical point of view. M. Bagchi and P.R. White explains that a smart card implementation can for instance show how many trips are performed using each card, and thus such factors such as card turnover rate can be calculated. A point is made that a limiting factor to the data collection quality is that trip length is not recorded on buses that only validate cards on entry. Due to the design of our system as seen in *chapter 4 Resulting system* where a passenger must perform a check out to end their trip, it is possible to log trip length and thus satisfy such a parameter should a train operator wish to analyse data recorded from their systems. This, in combination to aforementioned articles further indicates that the usage of NFC technology in the system would be suitable.

According to OK et al. and Hamberg & Hube, NFC as a technology has a bright future ahead of it. Jakubauskas, Bagchi and White emphasises that the usage of smart cards in ticketing systems contribute to a wider data collection used to optimise the transport system and analysis of travel behaviour. A multifunctional smart card technology could be a good choice of ticketing since the biggest drawback with a paper ticket is that it cannot be used as easily for optimising routes and that it is not multifunctional.

However as pointed out by Jakubauskas, the traditional paper ticket can still be used to validate travelling. In conclusion, it can be derived that a future ticketing system should include both support for traditional ticketing as well as smart card ticketing.

Chapter 2

Technical background and component details

This chapter is devoted to providing the reader with a sufficient technical background in order to understand the report. In order to provide this background, the chapter will start by laying out the system specification put forward to treat the project description.

The system specification will consist of a set of units in a network, the construction of which will require a set of evaluated hardware components. These will be explained in detail respectively as the system units are analysed.

2.1 System specification

The following hardware has been evaluated and chosen to constitute the parts needed for the ticketing system, starting with the central building platform for each unit in the system and working down to the details.

To manage the task of making each seat its own check-in handler, a seat unit was developed to be placed at each seat. Given that a train has a relatively high amount of seats; it was deemed necessary to develop a database that can handle high amounts of information regarding every single seat unit. Having only a database by itself would require each seat unit to access the database on lookup and registration of passengers, thus a database access unit was deemed necessary in order to handle a great magnitude of seat units making requests to the database, a butler of sorts.

A way to communicate between the seat units and the database access unit could be by wireless transmission. This enables greater placement freedom and omits potentially expensive installation of communication wiring. Thus, the seat units as well as the database access unit contain XBee modules configured to create a personal area network (PAN) on the train. Given that the database needs to communicate with every single seat unit, a star network topology was deemed as a suitable structure. The central hub would then be set as the PAN coordinator and the seat units as end devices. The XBee modules communicate using Zigbee protocol which means that the system would not be compatible with existing train WiFi-systems.

However being compatible with WiFi poses a security risk, for instance as any WiFi-compatible laptop running a security-oriented Linux distribution could be a potential threat. The Zigbee protocol is more focused on IoT embedded systems, and does not accept communication from a unit that is not part of the PAN setup. More information on the XBee radio modules can be found in *2.6 XBee radio module*, additional setup information can be found in *2.6.3 Connectivity.*

In order to let the train attendant have a good overview over the train seats, a smartphone-based application was developed. The application shows every seat unit status, including vacancy, name of traveller if available, ticket type etc. Most smartphones, however, are not able to communicate via the Zigbee network protocol, thus unable to interface with the PAN. Therefore, a radio unit was developed in order to act as a link between the XBee network and the attendant's smartphone. This is done by repeating relevant information from the XBee network via Bluetooth connection. That way, the application can send and receive information through a Bluetooth connection in order to communicate to the rest of the network. This Bluetooth connection would either require one transmitter capable of reaching the entire train, or several transmitters where the connection is "spread out" across each wagon. The latter option however could require that the train attendant pairs the hand-held device with each transmitter depending on the implementation. In either case, achieving Bluetooth coverage in the entire train is technically feasible. In summary, the whole ticketing system is divided in four parts: the seat unit, a database with an added access unit, a radio unit and an application. A detailed component specification of the parts used in the system will follow in table 2.1.

In summary, the whole ticketing system is divided in four parts: the seat unit, a database with an added access unit, a radio unit and an application. A detailed system component specification of the parts used in the system will follow in *table 2.1.*

Used in unit	**Name**	**Component type**	**Component specifications**	**Purpose**	**Explained in section**
Seat unit	Arduino Mega 2560	Microcontroller	Prototype board based on the AVR ATmega2560-16AU microprocessor, 54 pin GPIO, 16 MHz, USB, SPI, I^{2}C, linearly regulated, 4 kB EEPROM, 256 kB Flash memory	Main component of the seat unit, system specifications aimed at complex tasks and multiple peripherals	2.2
	Adafruit PN532	NFC/RFID reader	10 cm range, 13.56 MHz quadratic antenna , I^{2}C or SPI, Arduino R3 form factor with overlap, ISO 14443-compliant	Enables the seat unit to read contactless smart cards	2.3
	Arduino TFT display	TFT LCD display	160 x 128 pixel RGB display, 262 thousand colours, 42.72 x 60.26 mm	Displays data and interface prompts to passengers	2.4
	Green, yellow and red indicator LEDs	LED	Through-hole, 20 mA	Indicates seat status, for instance *checked in, reserved* or *error*	2.7.1
	Arduino membrane 3x4 keypad	Keypad	Membrane type, bridging connector interface	Used to acquire keypad input from passengers, for instance ticket ID's	2.7.2
	Brother standard-issue piezo buzzer	Buzzer	Piezoelectric, 5 V, through-hole, 4 kHz optimised	Provides audio feedback	2.7.3
	Connector for seat detector	Pin header	300 mil female header	Provides connection for the auxiliary seat detector	2.7.5
Seat unit, DAU	SparkFun XBee Explorer Regulated WRL-11373	TTL to 3.3 V logic level converter	100 mil, 28 pin full pinout from an XBee radio device, linearly regulated	Protects the XBee unit from damage caused by higher signal level voltage from the microcontroller	2.7.4
Radio unit, DAU	Arduino Uno R3	Microcontroller	Prototype board based on the AVR ATmega328P-PU microprocessor, 14 pin GPIO, 16 MHz, USB, SPI, 1kB EEPROM memory, 32 kB Flash memory	Main component of the DAU and radio unit, system specifications aimed at simpler tasks and fewer peripherals	2.2
Radio unit	Funduino XBee shield V03 B	TTL to 3.3 V logic level converter and XBee unit reset circuit	Arduino R3 form factor, XBee reset function	Protects the XBee unit from damage caused by higher signal level voltage from the microcontroller and allows for hard reset of connected XBee unit	(2.7.4)
	Guangzhou HC IT HC-06 Bluetooth module	Bluetooth radio to UART converter	Up to 1.38 Mbaud Bluetooth transmission, 3.1 - 4.8 V, UART, over 40 m urban range	Links the radio unit to the train attendant's smartphone	2.5
All	Digi XBee XBP24 S1	Zigbee radio unit	60 mW, UART, 3.1-4.8 V, 1600 m range (100 m urban environment)	Connects the system units to the personal area network	2.6

Table 2.1: Components used in the ticketing system. Please observe column *Explained in section*: each component is explained further in this chapter.

2.2 Microcontroller platforms: Arduino Mega 2560 and Uno

Arduino is an advanced virtual RISC (AVR)-based microcontroller breakout platform that can control electronic components connected to the board [5]. The behaviour of the Arduino can be modified by programming the board using the affiliated Arduino software, the Arduino IDE [6]. A program written using the software can be uploaded to the board from the computer through USB connection [7], the programming session of which is done by universal asynchronous receiver/transmitter (UART) communication.

The microcontroller on the board consists of a microprocessor containing flash memory for holding a program, random access memory (RAM) for storing data and a set of input/output pins which link the microcontroller to the rest of the electronic components [5]. The precise memory capacity varies between different types of Arduino boards since there are different kinds of AVR microcontrollers [8].

There are both analogue and digital inputs and outputs on the Arduino [5]. The Arduino GPIO pins communicate using transistor-transistor logic (TTL) voltage levels although the analogue pins can vary between 0 V and 5 V [5]. The Arduino has sufficient power output to for instance power LEDs [8].

This project utilises the Arduino Uno and Mega 2560 boards. Both of which have the same basic features but mainly differ in size [9]. The Mega is a bigger variant and features more pins and memory than the Uno. This allows for more electronic connections and larger programs [5]. The Mega features a surface mounted AVR ATmega2560 as the microprocessor. Unlike the Uno, the Arduino Mega's microprocessor cannot easily be replaced [5] since the microprocessor is soldered onto the board. In contrast, the Uno utilises a connector for the microprocessor.

2.3 Near field communication peripheral: Adafruit PN532

The project will utilise a near field communication (NFC) peripheral consisting of a Philips PN532-based printed circuit board implementation made by Adafruit Industries [10]. The unit communicates with a frequency of 13.56 MHz using established protocols specified on top of the ISO 14443 specification by the NFC forum [11][12].

The function of the peripheral follows closely that of the standard specified in ECMA-340 and in ISO 18092. This enables an early prototype of the ticketing system to interface with a broad selection of NFC-compatible smart cards such as the MIFARE card. According to NXP Semiconductors, the trademark owner of MIFARE, 80% of all contactless ticketing credentials utilises MIFARE technology [13]. This should in theory allow for a wide level of contactless smart card support for the alpha seat unit prototype.

In order to comply with the standard, MIFARE compatible smart cards and tags are expected to present a unique identification (UID) number when activated. This number is usually synonymous with a unique card serial number of sorts, and is burned into the card's EEPROM during formatting of said card memory [14]. The early seat unit alpha prototype utilises the UID as a serial number to identify the passenger, although this is not recommended. Firstly, a MIFARE classic card has a UID of four bytes which may be determined from a broad selection of methods, ranging from predetermined manufacturing codes to pseudorandom sequence generation [14]. Since there exist pseudorandom methods of UID generation, there is a slim chance that for a given card there may exist another card bearing the same UID. However the chance of a collision when using single size UIDs is at most one in 2^{32}, or about one in four billion. There is UID variant that utilises a larger UID size, called "double length UID" [14]. MIFARE cards bearing double length UID instead present a seven byte long UID upon activation, making it even less likely to find a duplicate (about one in 70 quadrillion, $1/2^{56}$). The probability of a collision lessens somewhat when accounting for the manufacture codes i.e. that manufacturers may not manufacture duplicate cards. Single length MIFARE cards however do not have manufacturer codes [14].

Although methods exist where a simple device may clone any MIFARE card provided one memory block is obtained, as for instance shown by Flavio D. Garcia et al. in the computer security report chapter *Dismantling MIFARE Classic* [15]. Meaning that a commercial implementation of the system should include database cross checking as well as an internal deciphering system of actual card data. Unless suitable methods are taken to cross correlate the cards in a wider database system, nefarious technology aware travellers may use cloned cards for travelling on another traveller's credit. As for the concept prototype however, a formatted and thus usable NFC-compatible card (or tag) can hereby be seen as a unique passenger card.

The seat unit is compatible with single- and double length UIDs and can detect cards from a distance of about 10 cm. In our own experiments, the unit can scan a passenger card within less than half of a second upon detection since the UID is stored in the very first read memory block on the card [14].

2.4 Arduino TFT display

The alpha prototype utilises a thin-film transistor (TFT) liquid crystal display unit to interface with passengers and relay important information. The display is very easy to work with as it accepts very rudimentary functions such as lines, rectangles and text [16]. The display is a very basic implementation consisting of an on-board controller IC which relies on a selection of colour and position graphics. The available palette consists of a three byte RGB selection, meaning 262 thousand colours available for 160x128 pixels [16].

The display utilises a character map based on a proprietary implementation which does not follow ordinary Unicode notations. This poses a problem in case a system developer wishes to print other than standard alphanumeric ASCII-characters.

The choice of this exact display model was made in part due to the fact that it is a very basic display implementation where graphics are drawn using swatch settings, text, line commands and rectangle placement. The module features an available option to load bitmaps from a provided SD-card unit should said simple graphics controller not suffice. One could imagine that a commercial implementation of the ticketing system prototype might utilise more advanced graphics in order to appeal to device aesthetics. However a basic display can be deemed acceptable for displaying passenger information in a more conceptual manner.

2.5 HC-06 Bluetooth module

Bluetooth is a wireless technology standard that has enabled billions of devices to communicate with each other [17], the standard of which is currently maintained by the Bluetooth Special Interest Group.

The radio unit prototype uses an HC-06-based Bluetooth module to communicate with the attendant's smartphone. The HC-06 Bluetooth module offers a complete solution for Bluetooth communication on a single board [18] by offering regular UART for communication in order to interface to microcontrollers. It is also able to run at voltages of 3.1 V to 4.8 V, thus it is possible to power the module directly from the selected Arduino microcontrollers.

2.6 XBee radio module

XBee is an RF module ideal for low-power and low-cost applications [19]. According to the manufacturers of the XBee radio module, the XBee is an embedded solution that provides cost-effective wireless connectivity to electronic devices [19]. The project is using XBee-Pro Series 1 devices, which according to the manufacturer is a power-amplified version for extended-range applications. The XBee modules are of version XBP24-ACI-001 which operates at a voltage of 3.3 V at 215 mA [19]. According to the XBee 802.15.4 RF module documentation from the manufacturer, the XBP24 has a 1600 meters line-of-sight range and 100 meters urban range. The XBP24 uses the IEEE 802.15.4 networking protocol, which is designed for high-throughput applications [19].

2.6.1 802.15.4 Networking Protocol

The 802.15.4 networking protocol for XBee-Pro Series 1 has two different operation modes [20][21], one in which the RF module acts like a coordinator. The coordinator has control over all devices connected to it and can identify other devices connected in its Personal Area Network (PAN)[20]. The other operation mode allows the RF module to act as an end-device [20].

2.6.2 Network operation modes

As aforementioned, the XBee-Pro Series 1-module is compatible with two different network operation modes: AT mode and API mode. This subsection is devoted to explaining both types of operation modes.

AT mode (Transparent mode)

According to the XBee module manufacturer Digi, the transparent mode allows the module to act as a serial line replacement. In AT mode, the received data through the serial input is immediately transmitted onto the network [22]. Moreover, Digi states that received data by one module is exactly what is sent from another module. To run modules in transparent mode, every XBee module must be part of the same network. This is achieved by configuring the modules to bear the same PAN ID and channel parameters [22]. In transparent mode, all modules read the sent data on the serial and is therefore the ideal method when facilitating communication between two modules.

API mode (Application Programming Interface)

API mode is a frame-based method for sending and receiving data to and from a radio's serial UART [23]. According to the manufacturer Digi, API mode allows the programmer to receive packet delivery confirmation on every transmitted packet. The data that is sent must be formatted in frames with the destination address and payload. The common way of using API mode is in larger networks involving end devices talking to multiple targets.

2.6.3 Connectivity

The network topologies for XBee series 1 are either point-to-point or point-to-multipoint which can be implemented as a star topology [21]. In the star topology implementation, each end device is directly connected to the coordinator as demonstrated in *figure 2.1.*

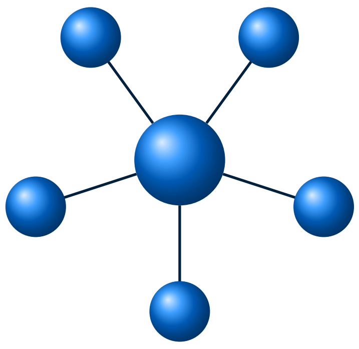


Figure 2.1: Demonstration of a star network topology. Observe that no end-node is communicating with another end-node, only via the central hub of the network.

2.7 Auxiliary hardware

This section briefly describes the technical specifications of auxiliary hardware that is not necessary for achieving rudimentary functionality of the system.

2.7.1 LEDs

Due to the effects of electroluminescence, LEDs light up when voltage is applied across its leads. The seat unit has three LEDs in order to further indicate the current seat status beyond the displayed information. One green, used for indicating a valid check in; one yellow, to indicate whether the seat is reserved further onto the trip (flashing) or reserved at the current station (lit); and one red LED, used to indicate errors. Every LED is current-limited using resistors in order not to burn out.

2.7.2 Keypad

The keypad utilises a very standard method of bridging rows and columns. There are seven pins, four of which correspond to the rows and three correspond to the columns. When button 1 is pressed, the pin indicating row 1 gets bridged with the pin indicating column 1. The firmware then detects that pins "row1" and "column1" are bridging. The same functionality can be applied to all keys on the keypad.

The addition of the keypad to the seat unit allows for accepting ticketing information beyond NFC smart cards. This information can come in the form of, for instance, SMS ticketing or paper tickets.

2.7.3 Buzzer

The seat unit is equipped with a piezoelectric buzzer in order to provide audio feedback to the user. The buzzer is active at a wide selection of scenarios, for instance when a button is pressed or to indicate that the seat unit has encountered an error. Every frequency in use by the buzzer of the seat unit corresponds to actual frequencies of piano notes in order to limit possible annoyance by regularly harassing passengers with beeping seat units. To further lessen possible irritation from the seat units, the buzzer audio intensity is limited using a resistor.

2.7.4 XBee signal level converter

The Arduino platform communicates using a TTL logic level of 5 V, which means that the logic voltages are too intense for the XBee unit [19]. Thus, a level converter is used in the seat unit and the DAU to limit the TTL logic voltages to a suitable level for the XBee-unit. The signal level converter used in the project is an "Xbee Explorer Regulated" from SparkFun.

2.7.5 Seat detector

The seat unit supports an external seat sensor via a connector at the bottom of the seat unit. The expected interface is a simple bridging, either by button or more sophisticated seat electronics. This is useful for detecting when a person is currently sitting on the seat in question. Should the seat sensor be installed and active without the seat being checked in, the seat unit will for instance stop registering new passengers at the seat and alert the train attendant that there is a passenger at this seat who has not paid for the trip. This shows up as a red indication on the application, a red light on the seat unit and a message on the display urging the passenger to pay for their trip.

2.8 Software

In order to achieve adequate hardware function, writing firmware and other system software is a necessity. The following sections will discuss the tools that will be used to develop the system application. The main stages of software development consisted of developing the firmware, the personal area network and finally the application.

2.8.1 Arduino integrated development environment

The Atmel microprocessors on the Arduino Uno and Mega development boards are programmed using the stock Arduino integrated development environment (IDE) which is a derivative of Processing-C. In addition to the

IDE, the system contains classes written in C++. The IDE offers the possibility to compile and upload the code from the same application [6].

2.8.2 Android Studio

When developing the Android application, a choice was made on whether to use Android Studio or Eclipse and its Android plug-ins. Eclipse is a multi-language integrated development environment, Android Studio is the official IDE for Android development and is developed and maintained by Google. Both IDEs are free to acquire and use, however the final choice fell on Android Studio mainly due to it providing an emulator with real life testing. This in theory eases on development time.

2.8.3 XCTU

XCTU is the main software used for configuring the XBee modules and is provided by Digi, the manufacturers of said modules [24]. The software provides all necessary configurations needed to create a personal area network in the shape of a star topology [24] as demonstrated in *figure 2.1.* In order to configure the device, an XBee explorer USB connector is used to connect the XBee to a personal computer for configuration.

Chapter 3

System development methodology

This chapter describes the procedure for developing the prototype. Both hardware and software were considered with regards to their cost and access for further development. The requirements were that the system should be implemented with tools that are well-documented, user-friendly and with development environments that are suitable for the project.

3.1 How the system specification was made from the problem description

In the initial stages of the project, discussions were made regarding the problem description specified in *1 Introduction*. This resulted in a rough overview on how the system could be designed as well as which peripheral units were needed in order to fulfil each functionality of the system. A decision was made that a seat unit would be needed in order to enable each seat to check in and out passengers. A smartphone application was deemed as a suitable solution to let the train attendant stay in control over each seat. A database (and thus an access unit) were deemed suitable to store the passenger information for the system, as this would not tax the seat units with unnecessary storing functionality. A flow chart was made for the check-in process in order to make sure everyone involved were on the same page of the project. System specifications were then made to fit the problem description as specified in *1 Introduction* and in the aforementioned flow chart (a derivation of which can be found in *figure 4.3* found on page 34).

3.2 Development of the seat unit

The following section will focus on the development process of the seat unit as visible in the non-faded part of the generalised system as a whole in *figure 3.1*.

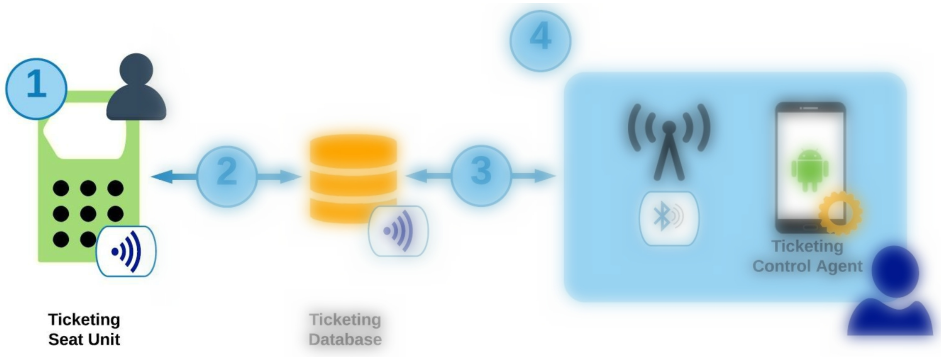


Figure 3.1: The seat unit of the system in focus (1) as contrasted to the rest of the system.

The seat unit development cycle consisted of a thorough investigation of which functions were needed in order to fulfil the project's purpose. In order to keep compatibility with existing systems of dynamic travelling, i.e. commuting using contactless smart cards, it was deemed necessary to include near field communication (NFC) functionality into the unit. However, a lot of travel methods omit the usage of a smart-card, for instance text message ticketing where the passenger is given a travelling code. The unit keeps compatibility with such forms of ticketing by including a keypad for entering a travel number.

In order to determine the placement of the NFC reader on the device, a very limited set of four students were asked to place a card onto the reader in order to check in. All of the subjects placed the card onto the keypad in order to perform a check in, thus it was determined to place the reading portion of the alpha prototype seat unit behind said keypad.

In contrast to the simple interface posed by the keypad, the NFC peripheral posed a challenge as its serial peripheral interface (SPI) mode requires communication in little-endian. It was decided to interface the NFC peripheral using the I^2C protocol in order to avoid the endianness differences of the integrated SPI components. The main reason for trying to implement SPI primarily was made because the display peripheral's embedded SD card reader is interfaced using SPI. In theory, having every peripheral communicating using the SPI bus would allow for a multi-slave SPI setup overall between the unit peripherals.

3.3 Development of the radio unit

The following section will focus on the development process of the radio unit as visible in the non-faded part of the generalised system as a whole in *figure 3.2.*

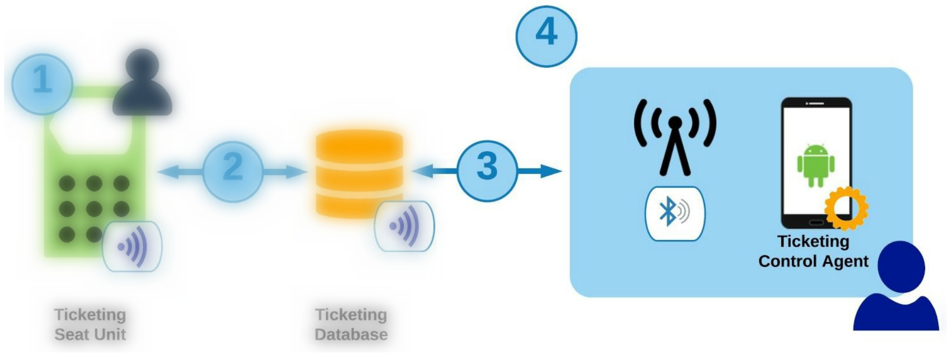


Figure 3.2: The radio unit of the system in focus (4) as contrasted to the rest of the system. The blue box symbolises the separation of the application as seen by the rest of the network which only communicates with the radio unit via (3). Only the radio unit can communicate to the application.

Since the radio unit acts as a link between Bluetooth and XBee, it needs to be able to read and send data to both ends. As previously mentioned in section *2.1 System specification*, an Arduino Uno was used to forward data between the two networks. However to handle communication from multiple sources, multiple serial ports on the Arduino were needed. In this case, two serial links would be required but the Arduino Uno only features one hardware serial port. In order to get more serial ports on the Arduino Uno, the library SoftwareSerial was used. This enabled serial communication using digital pins on the Arduino different from those corresponding to the hardware serial.

The XBee module was connected to the Arduino hardware serial on pin 0 and 1 and the Bluetooth module was connected to pin 10 and 11 by using a software serial port. In actuality it does not matter which device is connected to which port since the digital pins 10 and 11 will now equally support serial communication as experienced on pins 0 and 1.

The unit features a continuous system loop that runs until content is available on one of the two serial port buffers. This usually occurs when data has been received. The data will be treated differently depending on whether the data is received from the Bluetooth or the XBee module. The application on the Bluetooth-connected smartphone does not require all of the communication taking place on the XBee network, thus only messages that regards the application is forwarded via the Bluetooth serial port. All other messages are deleted from the serial buffer. However, any message received on the Bluetooth serial port is forwarded to the XBee network, as the application might want to instantiate communication to either the database access unit or the seat unit.

It is important not to overload the serial buffer on the Arduino, otherwise received messages may be lost. The buffer becomes overloaded when the transmitting unit sequentially sends data too fast for the receiving end to handle. It is therefore important that the Bluetooth and XBee devices use some delay between each transmission. That way the radio unit has time to read from the buffer and forward messages without overflowing the buffer. The radio unit should also use delay between each transmission to prevent overflow at the receiving ends. The system was thoroughly tested in order to find a good delay time. The test was conducted using the function *overflow* in the SoftwareSerial library, which returns a boolean if the buffer is overloaded. Ultimately, a 125 milliseconds delay seemed to work well which is considered a good time as it does not affect the total communication time notably.

3.4 Development of the database

The database started out as an Entity-Relationship (ER) diagram in order to easily overview all tables and how the tables are connected. This is done to detect problems or flaws at an early stage before programming the database.

The first database management system (DBMS) used in this project was MySQL, which has a lot of documentation available on the Internet. Computer software problems during initial stages of creating the database in MySQL caused problems and the DBMS was changed to MariaDB. The database was created following the final ER-Diagram. When some changes had to be made, it was found to be harder to recreate the database with said changes. Instead, direct manipulation in the table was more doable if this table was linked to another table. The final ER-diagram is visible in *figure 3.3*.

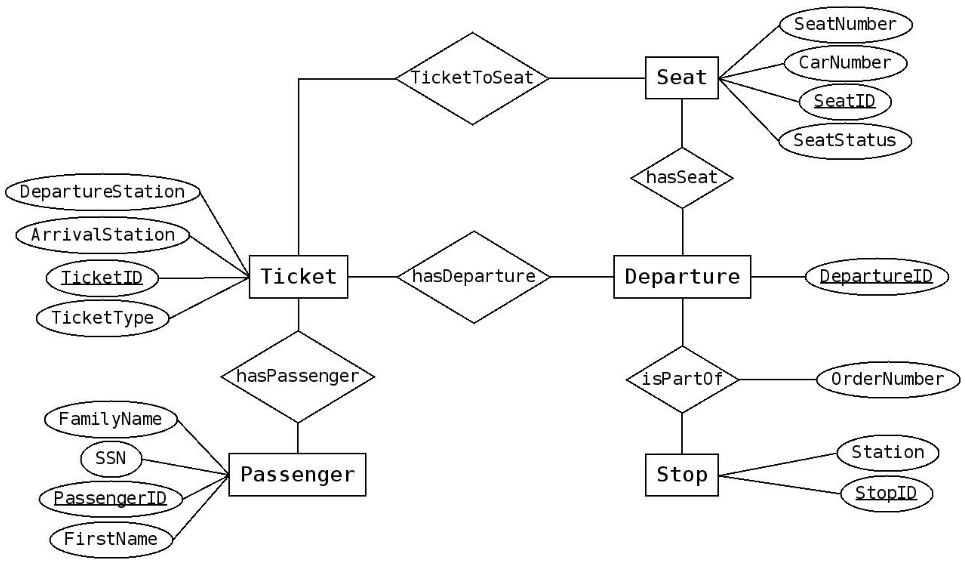


Figure 3.3: Final ER-diagram of the database.

3.5 Development of the database access unit

The database access unit (DAU) was developed in two phases. First in the in-development prototype phase and then in the alpha prototype phase, the reasoning behind this was that the in-development prototype was created in order to detect communication issues between the system units and the alpha prototype would serve as an actual unit in the system. The DAU is an addition to the database which serves as the link from the network onto the database, and is presented as (2) in *figure 3.4*.

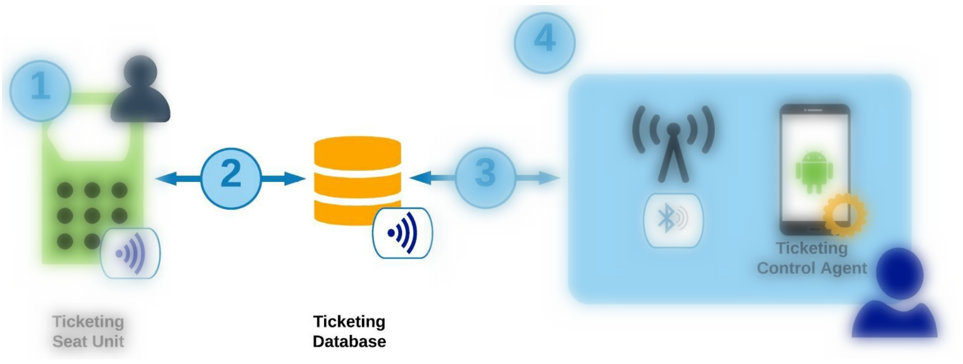


Figure 3.4: The DAU of the system in focus as contrasted to the rest of the system. As the DAU acts as the coordinator of the network, all communication has to pass through it (2 and 3).

3.5.1 In-development prototype

The DAU started as an Arduino Uno with an Ethernet shield. We connected the Arduino to the database using the MySQL Connector Arduino package created by Dr. Charles A. Bell. Using this package, search queries for the database were stored in arrays of strings which require a lot of memory space on the ATmega328 microprocessor on the Arduino. As the MariaDB database access functionality requirements eventually grew past the maximum flash storage space of the ATmega328 microprocessor, an μSD-card was added as a storage method for search queries. The late realisation of a lack of storage space and the addition of the shield when the project was in its final phase of development proved not up-to-par with regards to the project planning and the in-development database prototype was terminated.

3.5.2 Alpha prototype

The DAU was developed further to act as a full substitute to the database. The main idea was to have a continuously running master application which processes and possesses information about every seat in the train. In addition to this, hard-coded bookings for a specific train journey was added. The DAU had to communicate with the seat units and radio units in order to exchange information about bookings, check-ins, checkouts, reservations, conflicts and current station status.

The development began with the implementation of the master application in Java Eclipse. The used programming development methodology is known as XP (Extreme-Programming), which is useful when project requirements rapidly change [25]. One may not say that the requirements rapidly changed, but the alpha project phase required a DAU without the MariaDB database and thus parts of the principles of XP were used. Said principles used in development were simplicity, pair-coding and unit testing when bugs are detected [25].

The application was designed to loop continuously during a train's journey. As the train passes different stations during a journey, the application had to be dynamic in order to change stations when needed. This was made possible by using a string scanner in the application where the master could type in the current station to change it. As the application had to possess and process information about the bookings and seat units, all travel bookings were fixed in a text file with either six or seven columns. Each row corresponds to one booking, whereas the first column was the ticket id followed by ticket type, first name, family name, start-station, end-station and car-and-seat number (featured only for reservations).

Since the data of the bookings was to be processed and stored, an easy-accessible way of stashing was preferable. The abstract data structure hash table was used to store data, since it has a constant time of inserting and look-up. Furthermore, each row of the booking was put in the hash table with the ticket id as a key and the rest of the row as a value. The rows with seven columns were put in a hash table for reservations only.

Since the passenger checks in and out by typing or scanning a ticket ID at the seat units, the master application utilised the existing DAU hardware to communicate with the seat units wirelessly. This was done by setting up a serial processing between the Java application and the DAU's Arduino. A platform referenced library JSerialComm was used in order to allow inter-communication between the application and the Arduino over the serial link [26]. The Arduino board was programmed using the Arduino IDE with an XBee module connected to it, whereas the XBee was configured to be the network coordinator in the XCTU software.

3.6 Development of a smartphone application

In the process of relaying information to the train attendant in a comfortable and efficient manner, a hand-held device that synchronizes wirelessly to the ticketing system and the train database was suggested as a possible solution. The application as part of the ticketing system as a whole is visible in *figure 3.5.*

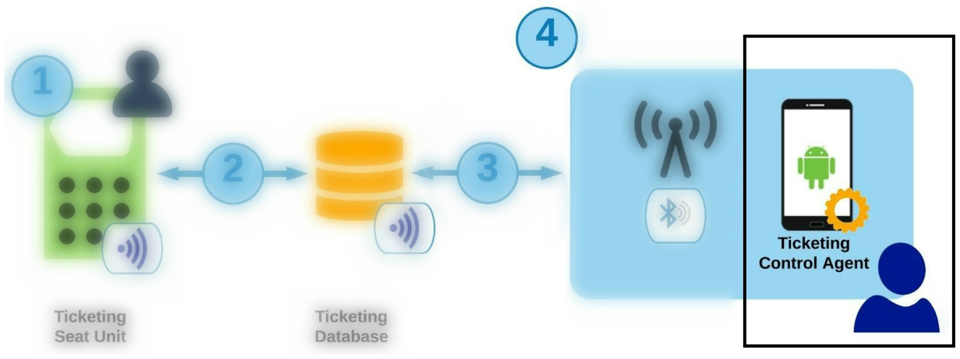


Figure 3.5: The application part of the system in focus as contrasted to the rest of the system. As the DAU acts as the coordinator of the network, all communication has to pass through it (2 and 3).

By having a wireless device, the train attendant is free to roam the train as they decide. Choosing a smartphone to act as the train attendant's front-end device is a good choice since a smartphone already has all of the needed sensors and wireless systems that are needed in order to interact with the ticketing system. Constructing and programming a dedicated device is both expensive and time-consuming thus deemed off-scope for this project.

The major brands of smartphone producers today build smartphones for a very small number of operating systems. The operating systems iOS and Android are practically the two leading forces in the world as of now with some minor competitors. Any real difference between the operating systems however is too small to be of any importance for the ticketing system.

There are a few key reasons as to why we choose to develop for Android: development and prototyping tools for Android applications are currently free to acquire and use, also the amount and diversity of Android devices is very large. According to IDC's report in August 2015, Android held over 80% of the market during the second quarter of 2015 [27]. Applications for Android are written in Java, an object-oriented programming language that the project group is well-versed in. Java also has a very large user base and a suitable amount of documentation. Conceptually however, a similar application can be developed for iOS.

A simple model of a human-computer interaction design dictates that to be able to design a good user interface you have to first establish the requirements and needs the users have.

- Who are the possible users?
- What needs do they have?
- In what context will the users use the interface?

The system is to be placed in a train and the application is entirely meant for the train attendant. This means that all possible users would be limited to users hired by the train company, who should be familiar with the company's ticketing policies and ticketing system. The train attendants will use the application while on the train both while on the go and at stops.

The train attendants want to be able to see all seats on the train, as well as the state of each seat. Train attendants might also want to be able to see all passengers associated with the train and what seat and type of ticket they have. This means that all the information that the system has about the passengers and seats on the train will have to be transferred to the application at regular intervals. With these guidelines it is possible to make a fairly decent prototype.

Since Java is an object-oriented programming language, it is a good idea to start with dividing the application into classes and objects. At its core, the application is going to handle seats. Making a class that describes a seat means that the following simplifications can be made in the code. As the number of seats in a train rarely changes it is possible to set the number of seats to a constant number. The seat's physical identifier can be used as an identifier in the collection of objects in the application. The fact that all elements of the interface are centred around the seats gives more freedom when indexing the instances of the class.

Every seat will need to have a couple of attributes, fields and methods for it to use. By making the application only handle the seats on the train, the need for handling and storing all possible reservations and possible passengers are eliminated. The extraordinary case of having a ticket on the train that belong to another train is not considered.

Allowing Bluetooth compatibility on Android is possible by using the available classes BluetoothAdapter, BluetoothDevice and BluetoothSocket. They are, in sequence, responsible for describing the physical Bluetooth on the device, describing the physical Bluetooth on the remote device, and finally describing the actual connection for the two devices.

Trying to establish a connection via Bluetooth in the UI thread on Android is however not encouraged by neither the hardware nor any programming conventions. Therefore, the connection has to be given its own thread to run in. In this new thread, ConnectThread, the BluetoothSocket is declared and initiated with the method *remoteBluetoothDevice.createRfcomm SocketToServiceRecord(MY_ UUID).* The MY_UUID portion is the universally unique identifier (UUID) of the local device and is used to authenticate the request to open the connection. Once the socket has been initiated, the connection is enabled by calling the method *BluetoothSocket.connect().*

Once the socket has connected, it is passed back to the main thread where a handler receives a message containing a status integer and a reference to the socket. The socket is then passed to another thread that handles the actual read and writes. The ConnectedThread takes two streams and attaches them to the socket, one InputStream and one OutputStream. These streams own their respective methods *read()* and *write()*, which are used to get bytes from the remote service, via a buffer, and send bytes to the remote device.

3.7 Development of the personal area network

The system units which feature XBee modules are the DAU, the seat units and the radio units. The XBee featured on the DAU acts as the coordinator in the network topology seen in *figure 2.1 on page 15.* While on the seat and radio units, the XBees will be configured to act as end devices. The XBee modules were configured using the software XCTU by connecting the respective modules to a computer via an XBee explorer USB connector.

The XBee acting as a coordinator was simply set as *coordinator* in the function set, and all the other XBees where set to *end device.* The personal area network (PAN) ID needs to be the same for all the XBees in the network in order to communicate with each other. We selected "1111" as our PAN ID, however the exact selection of PAN ID number does not affect the system in any noticeable way. The AT mode-setting was set to "enable" for all XBees. On the end devices, the destination address was set to 0 (low), which means that the XBees only communicate with the coordinator.

In summary of the PAN setup process, the database will now act as a coordinator in the star network topology, meaning all information on the network will go through the DAU. All the seat and radio units will be end devices which communicate directly to the DAU.

3.8 Merging the components into a prototype

The system was interconnected in two attempts spanning two development stages. One attempt was made during the in-development prototype stage and the other during the alpha prototype stage.

3.8.1 In-development prototype

For the in-development prototype we used one keypad, one Arduino Mega, two XBee modules and one Arduino Uno with an Ethernet shield mounted. The DAU was connected to the database using the Ethernet shield on the Arduino. Furthermore the seat unit was connected to the DAU with the serial port transmitted by an XBee module. After the addition with a μSD-card to the DAU the in-development prototype stage was closed and the alpha prototype stage opened.

3.8.2 Alpha prototype

The merging of the alpha prototype is similar to the in development prototype. The several units of the seat unit such as the keypad, NFC, TFT-display, Arduino Mega and XBee radio frequency (RF) module were put together and connected in a 3D-printed shell designed in SolidWorks software. The DAU was connected to an Arduino with an XBee RF module plugged in.

Chapter 4

Resulting system

This chapter focuses on providing the reader with a concatenated view on how the final alpha prototype system works. The chapter will initiate with a section describing the design of the main unit interfaced by passengers (the seat unit) and continue describing the resulting system as a whole. The last section focuses on the application and ends with a demonstration of the results following a scenario analysis on how potential issues with the prototype plays out.

4.1 Design of the seat unit

The seat unit is designed as a simple mockup that can double as an enclosure for all the hardware during the development cycle and also as a complete demonstration tool. The case was drawn in SolidWorks and then printed out using a 3D-printer. The plastics used are based on bio-degradable compounds extracted from agricultural processes.

The size of the enclosure is, at this stage, only big enough to accommodate all of the hardware and can change according to future needs.

The actual enclosure is shown in *figure 4.2*, together with the layout inside the enclosure, *figure 4.1*.

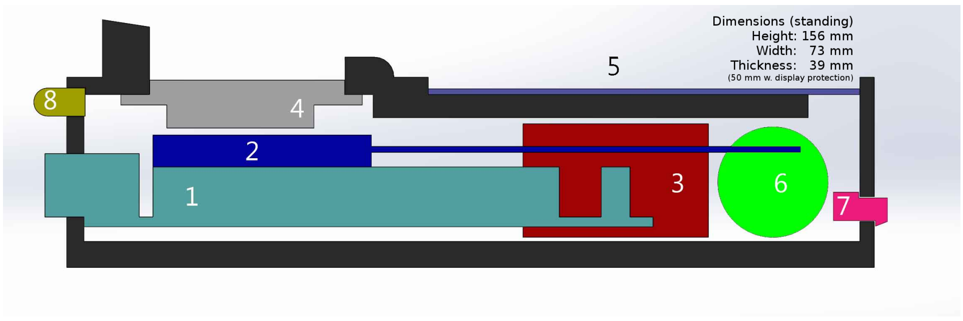


Figure 4.1: Seat unit hardware layout seen in a cut-through manner. The bottom of the unit is positioned to the right in the image; the front of the seat unit is oriented to the top in the image. 1 (cyan) is the Arduino microcontroller board. 2 (blue) is the NFC-peripheral. 3 (red) is the XBee radio peripheral with the signal level converter attached. 4 (light grey) is the TFT display. 5 (light purple line) is the keypad. 6 (green) is the buzzer. 7 (pink) is the connection for external seat sensor. 8 (yellow) represent the three LEDs in a row. The LED placement is better illustrated in *figure 4.2*.

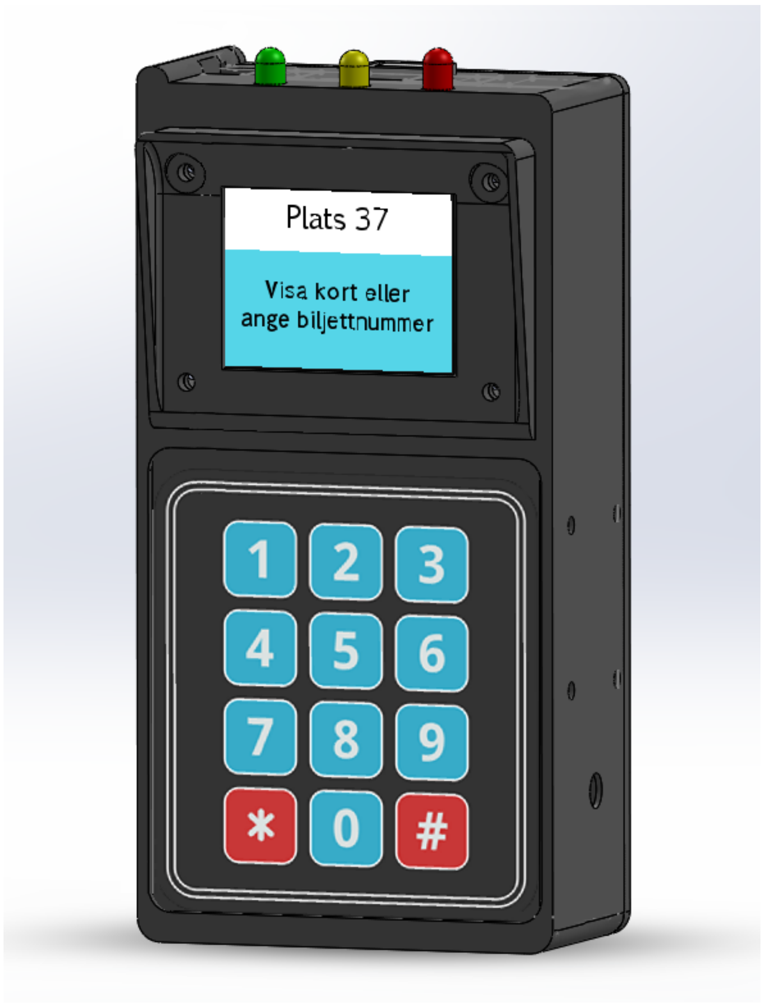


Figure 4.2: 3D-rendering of the seat unit alpha prototype. Notice the display featuring a protective screen which adds to the total thickness of the unit as seen in *figure 4.1*. The holes on the right side of the unit make up an air hole for the buzzer (large hole) and mounting holes for the XBee signal converter (four smaller holes).

4.2 How the alpha prototype works

When a valid ticket ID is detected, either by keypad or NFC, the unit sends out a call to the network addressed to the database access unit. The call-out consists of a radio string message, which in turn contains data regarding what car number the seat unit is in; what actual unit number the unit corresponds to in the network; a command that the unit intends to perform a check in/out; and also the ID in question should this be a check-in request. The process of a check-in progress is illustrated in *figure 4.3*.

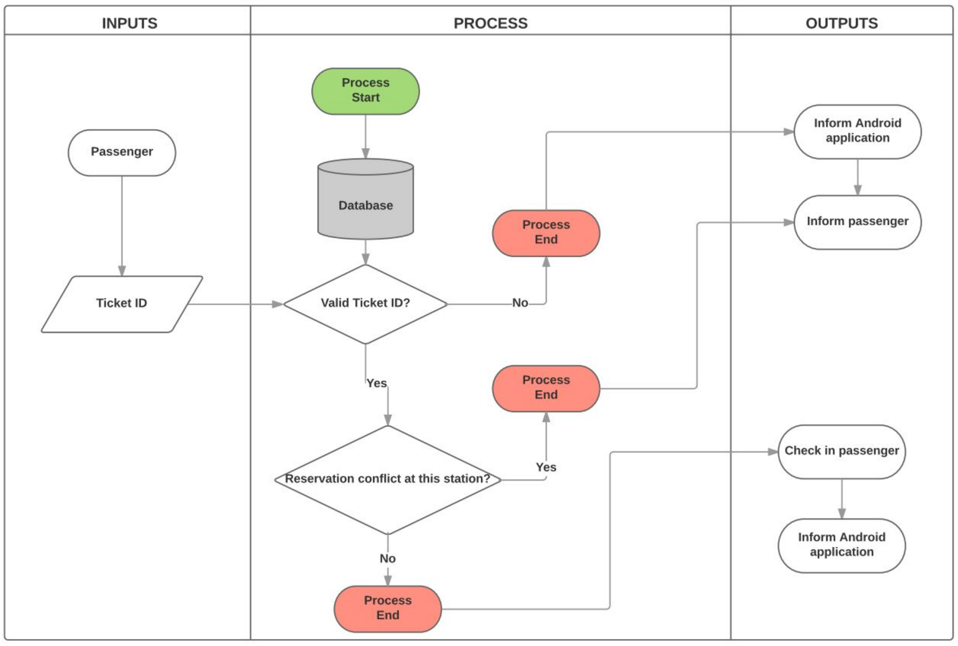


Figure 4.3: Flowchart for a simplified check-in process.

The DAU accesses the database in order to make a decision on whether or not the check in/out is valid. Should for instance the acquired data correspond to a ticket number that was given to a particular passenger which has reserved this seat, then this is probably an attempt to claim his/her booking and thus the suitable response is a radio check-in acknowledgement. This can consist of an acceptance of the booking, or a deferral i.e. this is an illegal booking attempt. The seat unit runs on an integer timer in order to process a response from the DAU as quickly as possible. In case no response is given by the database, the unit throws an error and displays a message to contact the staff.

The DAU also transmits the new seat update to the train attendant's application. In case there is an error, the seat unit itself sends out a masked message portraying to be the DAU, which the DAU directly relays onto the application via the radio unit. The message in question is a simple seat alert, which the application treats as displaying the seat in question as a red colour. The system consists of seat units, a DAU, a radio unit and a hand-held device, running Android. When a passenger checks in at a seat unit, the input is controlled by the DAU. The DAU then responds directly to the seat unit and forwards the messages to the radio unit. The radio unit acts as a communication link between the DAU and the Android application. *Figure 4.4* illustrates the communication links within the system.

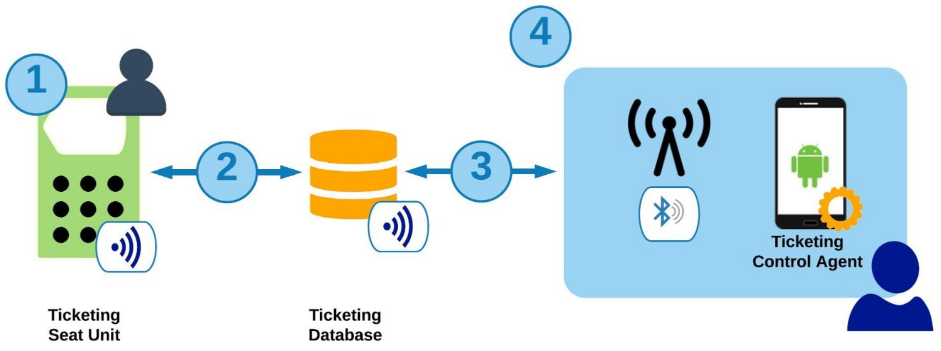
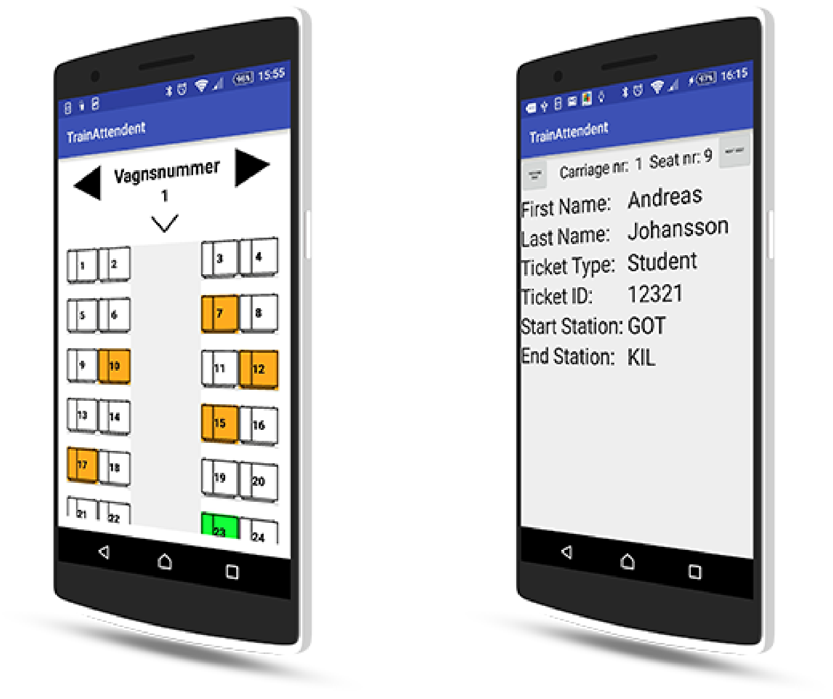


Figure 4.4: A simplified overview of the communication process within the ticketing system. (1) represents the ticketing seat unit, (2) represents the two-way communication from the seat unit to the database, (3) represents the two-way communication from the database to the application and (4) represents the train attendant with his/her smartphone application.

A passenger checks in at the ticketing seat unit (1) which transfers data via a wireless radio interface (2) to the DAU which is directly linked to a local database of passengers. The unit decides on how to treat the attempted check in, and responds accordingly back to the seat unit (2). The database then relays the updated seat information to the radio unit (3). The radio unit in turn relays the received message via the wireless interface Bluetooth (4) to the attendant's hand-held device, which updates the corresponding seat, as shown in *figure 4.5*. In special occasions, such as an error alert, the seat unit may completely omit the database request section seen in step (2) and (3). This is done to differentiate the purpose of the database access from the process of error reporting and illegal travelling attempts, thus the link is established directly to the attendant via the DAU.

The seat unit has three main methods for acquiring data. The first method is by entering a number on the keypad. The current settings of the keypad accepts any integer series of up to five integers but this can easily be



Carriage nr: 1 Seat nr: 9

First Name:	Andreas
Last Name:	Johansson
Ticket Type:	Student
Ticket ID:	12321
Start Station:	GOT
End Station:	KIL

Figure 4.5: The Android application as shown on the attendant's device. Note the coloured seats in the left view: these correspond to seats with a non-vacant status. Pressing a coloured seat results in the right view, where the application displays relevant data for the seat in question.

adjusted in firmware depending on need. The second method is by holding an NFC card to the reader. This initiates a readout of the stored card data. The alpha prototype of the seat unit collects only the card's serial identification number. The third method is via direct data injection. This is done either by entering appropriate commands into the smartphone application or using the database.

4.3 How the system handles specific scenarios

- If a passenger takes a seat without checking in at the seat unit, a seat sensor will be triggered. This sends out a warning to the train attendant's hand-held device via the DAU, and the specific seat will be marked red in the application. Red seats are visible among the other seats in the application's ordinary wagon view. The seat unit will also signal its red LED and display a message urging the passenger to pay for their trip.

- A passenger with a reservation may only check in at the reserved seat in question, provided the current (or recently left) station is the station from which the reservation is active. However, should the passenger for any reason fail to check into their seat before reaching the next station, the reserved seat will cancel its reservation and is now a regular seat. However the passenger's reservation still exists in the system and the passenger may now choose any other eligible seat and check in there for as long as the reservation is valid.

- Each seat unit informs whether it is reserved and how many stations are left until the reservation is in effect. Passengers may still accept that the seat will be taken by another passenger and check into the seat if it is currently vacant.

- It is not possible for a person to reserve a seat if that seat is already occupied for that travelling section.

- A passenger that decides to end their journey and leave the train before the ticket's final destination will be checked-out at their end station. If this station is undefined, as could be the case with commuting cards, then the default end station is assumed to be the end station should the passenger not check out their ticket earlier.

Chapter 5

Discussion

As the finished alpha prototype ticketing system may lack in distinct areas which makes it unsuitable for a commercial implementation, this chapter aims to compare the finished alpha prototype with our suggestions for a commercial implementation.

This chapter will also look back on the purpose of this project and analyse to what extent it has been fulfilled by the prototype. In order to further emphasise the analysis, this chapter will include a section devoted to a cross analysis of how a different methodology might have changed the outcome. For instance by looking at the chosen methodology and prototype development in a similar project and applying it to this one.

5.1 Regarding the alpha prototype

The main topic of this discussion will be focused on the system itself rather than nearby topics such as the choice of design and hardware. These topics are in its absoluteness important, however we as developers of this prototype believe that our thoughts will have a bigger impact regarding terms of improvement. Whereas commercial design and final hardware is a matter of embedded design and engineering taste.

Abandoning of the first DAU

We decided to abandon the first DAU when we realised that the amount of memory on the Arduino was insufficient since all search queries for the database were stored in the Arduino program as arrays of strings and we did not find any other way to store them.

Could the used database handling method be made in a better way?

The implemented DAU in Java and Arduino lacks in design and trust because it lacks well-developed error handling. Since it was a substitute to the MariaDB database and had to be implemented as simple as possible and fast due to lack of time, parts of the software development methodology Extreme Programming were used. This caused us to develop a program that works with the system specifications and ignore threats such as bugs and errors in the program.

Although the DAU is working, its implementation can be made in a better fashion. The DAU uses serial processing when communicating between the Java application and the Arduino application. It is therefore possible to write a script that loads the tables from the MariaDB database and processes it in the Java application. If implemented this way, the aforementioned lack of memory would not be a problem since a personal computer is better equipped to handle the memory taxing operations.

The implementation has a lot of potential to be developed further since the code is implemented in a simple way with classes and methods. Java and the library jSerialComm is well-documented and well-known, which gives possibilities for re-writing or cleaning up the program. Possibly even merging it together with the MariaDB database.

The personal area network

The initial PAN development stages were done using the XBee modules' AT mode since the modules came preconfigured in said mode. This turned out to be a good way to initiate the PAN development cycle since there was no complicated communication addressing needed in order to create the star network topology. As developers we knew in advance that AT mode broadcasts reaches every end device within range, which becomes problematic when intending to send messages to a specific unit. This was not considered critical for the development cycle since we were only implementing a prototype of the system with few network end devices and not an actual implementation. However this brings forward a prototype-only issue where incoming messages have to be filtered as seen in *3.3 Development of the radio unit*.

An actual commercial AT mode implementation could bring forward a scenario where 100+ seat units receive the same exact DAU message even though it might be intended for a specific seat unit. To avoid this, the XBee modules can be configured in API mode instead which allows for stock communication addressing. This in turn allows for single receiver communication since the destination address is included in the broadcasted message.

Android application

Building the application in Android Studio turned out to be a good choice. Android Studio has superb documentation and many great tools for compiling fast and easily transferring the code to a smartphone for debugging purposes. The development process was a steady process since Java documentation is readily available and abundant. Adding to this, there are many users on the Internet who willingly share their thoughts and experiences on questions about programming for Android.

The application has not been put to any tests yet when it comes to user experience or usability. The basic principles of human-computer-interaction have been followed while developing the application, but no analyses have been made to evaluate this claim.

The areas of the application where improvements can be made are mainly in the user interface. Different XML-files for layout options could be added, giving the customer the options of designing train-specific layouts to show for the user. This can also be used to alter other UI elements like language, colours, styles and sizes of UI elements. In the underlying code, more information can be shown for the user as well as higher degrees of automation, automated alarms, checking in and out passengers from the application, smarter handling of cars and which car to show, in the application.

The ticketing system's functionality

A seat-based ticket agent could work flawlessly with a train attendant since the prototype gives the train attendant the possibilities of prioritising where ticket validations are required. This could work very well since there can be possible stowaways in a certain car of the train. If the activity of stowaways is well-distributed over all cars, our solution still gives the train attendant information of which seats are in need of validation. This would not be possible in a ticketing system where the only tool is a smartphone application. That is, no seat unit included in the system for which the train attendant also has to mark passengers that have checked-in. The train attendant would still have to control every passenger and also update the application when needed. Therefore, the original problem of checking every passenger would remain as described in the introduction.

From the passenger's point of view, the seat unit could be a very handy ticketing agent since said passenger could easily find free seats. Furthermore the passenger could get direct feedback from the seat unit, getting updated and told if there are any reservations on the seat. This information also allows passengers with no seat booking to find a seat where no reservation conflict will occur. That is, a conflict where a reserved seat holder shows up later.

As for the system's compatibility with different ticket types such as NFC smart cards, SMS-tickets, electronic tickets and physical paper tickets; having a keypad and an NFC reader allows for a broad variety of ticketing.

A system without the keypad would require every passenger to carry an NFC-compatible ticket. In case a train operator would for instance decide to implement SMS-ticketing where the only needed information is a numeric ticket travelling code, then lacking a keypad would require the system backend to implement cross-system functionality in order to link travelling codes to NFC-tickets. This in turn could for instance decrease the amount of paper tickets, but a question of data gathering and privacy is spawned out of the fact that the train operator requires both the NFC information as well as the telephone number correlation. A solution to all this could in turn be to develop an NFC-compatible smartphone application, however this instead alienates users without smartphones of which the keypad solution does not.

A system with no NFC compatibility would prohibit passengers who for instance commute by smart card tickets or by using their phone as the NFC host. An NFC-less system would in turn mean that in order to allow for dynamic travelling, every passenger needs to travel by reservation ticket only or for instance by printing out a travelling code on a paper. This in turn can be seen as an environmental problem since the smart cards are multi-use in contrast to the paper tickets.

Even though the previous section has discussed specific parts of the system, there may still be specific scenarios for which the expected outcome is unclear. The next section aims to clarify the system outcome of a selected assortment of scenarios.

5.2 Analysis of scenarios

There are some imagined scenarios that could have an improved outcome.

The first scenario regards the risk of stowaways taking a passenger's seat while the passenger is visiting the train's facilities. A solution to this can be found in the suggested seat sensor. This sensor can also be used to indicate when someone has *left* the seat which in turn can be used to check out said passenger. Further, the hand-held device could indicate the seat as both red and green, a status which could imply that the checked-in passenger is no longer at the seat.

A scenario where a passenger forgets to check out before leaving could be solved in several ways. For instance by including separate checkout units at the train station, or even by implementing a check out-function as a smartphone application where the passenger now has the possibility of checking out while not on the train.

If a smartphone application on the passengers' side was to be developed, it would also give possibilities of displaying current seat activity in the cars of the train. Conflicts due to reservations could be solved in a convenient manner, and such an application would also provide an opportunity of faster navigation to a free seat.

5.3 Regarding the limitations of this project

All limitations were set due to not complying with the aim of the project and therefore would not contribute to the final system. However, every limitation touches on functions that we imagine could be featured in a commercial system.

There will only be one database

A commercial system would use several databases. One local database for each train, possibly one regional database for each province or at the very minimum a global database that handles all local databases. The first limitation was to use only one database. It influenced the work by letting us concentrate on the aim of the project, since the prototype only needs a single database to achieve the project goals.

The database will not cross-check NFC card security

A commercial system with several databases would need an implemented functionality for cross-checking NFC cards in order to limit the risk for duplicated or forged NFC cards. There are two reasons as to why this was decided not to be implemented; the first reason was that our prototype did not affect any real passengers and therefore was not needed; the second reason was that this functionality might have been vastly different from system to system, thus giving a prototype sketch of this functionality would only succeed in wasting our development time.

There will be no separated checkout unit

If a passenger chooses to get off the train earlier than expected, her seat would be occupied until her ticket's end station. This problem could be solved by installing separate checkout units beyond the seat units, both on and off the train, allowing passengers to relinquish their seat at all times. Such a solution gives flexibility to the off-getting passenger as well as for the commuter.

The project will not treat a crowded train scenario

It is impossible for our system while on the alpha prototype stage to handle passengers standing up. The current system solution is that standing passengers' tickets are still checked by a train attendant. Since our goal is not to replace train attendants, but instead ease their workload, train attendants were still expected to have to check passengers standing up. This only slightly influenced the work process as the units are already wireless and can be placed at standing locations as well. However needed code alterations in order to cope with such a scenario was never devised nor developed.

5.4 Social, economic and environmental factors of this technology

Our expectation for this system is that it will lower train operator loss due to non-paying travellers since the system specifically alerts train attendants when somebody is joyriding. This could result in increased train company profit which could be used to improve the commuting system. An improved commuting system could in turn bring more departures and a wider range as the possible number of users could increase. Giving more people access to a public transportation system allows for a greater people circulation in society. As a public transport system can be seen as an environmentally friendly alternative to people movement done by for instance cars, this technology also allows for environmental benefits. The environmental drawback with this seat-based ticketing system is that it requires a great increase in necessary electronics production per train, the manufacture of which in a commercial system might require the use of conflict minerals or environmentally-unfriendly metals. The decrease in traditional paper tickets may in turn be seen as beneficial for the environment.

Another economic aspect of this system is that as it becomes much harder to joyride, people who politically incline with the notion that public transport should be free will face an unwelcome economic setback or may start using other means of transportation.

An implementation of this system may very well lead to an improvement in the work environment for the train attendants. However if the system turns out to be effective to the extent that fewer train attendants are needed per train then it might lead to a reduction in work opportunities which is taxing in the social aspect for an out-of-work attendant.

Chapter 6

Conclusion

The goal for this project was to create a prototype for a ticketing system. The project has resulted in a working prototype which fulfils the purpose as specified by the commissioner Innovative Projects Sweden AB. A system prototype has been built and implemented as a ticketing system that takes passenger input and checks whether the input allows them to travel with the train or not. The input is then passed along to parts within the system where it is easily available for the train attendants which in theory simplifies their work environment. With regards to the system specification, the prototype works as intended although with another type of database handling than initially imagined.

With the project's analysis of scenarios as a take-off to further implementations, one can assess that a commercial system could emerge based on the concept demonstrated by the prototype.

Bibliography

[1] Lisa Hamberg and Christine Hube, "Near Field Communication: A study on the NFC technology's possible field of application and its development within social media," tech. rep., Institution of Media Technology, Södertörn University, 2014. `http://www.diva-portal.org/smash/get/diva2:784174/FULLTEXT01.pdf`. Read: 2016-05-05. Language: Swedish.

[2] Kerem Ok et al., "Current benefits and future directions of NFC services," in *2010 International Conference on Education and Management Technology*, (Cairo), pp. 334–338, IEEE, 2010. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5657642`. Read: 2016-05-27. DOI: 10.1109/ICEMT.2010.5657642. ISBN: 978-1-4244-8616-8.

[3] Gražvydas Jakubauskas, "Improvement of urban passenger transport ticketing systems by deploying intelligent transport systems," *Transport*, vol. 21, pp. 252–259, September 2006. `http://www.tandfonline.com/doi/abs/10.1080/16484142.2006.9638075`. Read: 2016-05-05. DOI: 10.1080/16484142.2006.9638075.

[4] M. Bagchi and P.R. White, "The potential of public transport smart card data," *Transport Policy*, vol. 12, p. 464–474, September 2005. `http://www.sciencedirect.com/science/article/pii/S0967070X05000855`. Read: 2016-05-31. DOI: 10.1016/j.tranpol.2005.06.008.

[5] Dr. Simon Monk, *Programming Arduino: Getting Started with Sketches.*, ch. 1. This Is Arduino. McGraw-Hill Professional, AccessEngineering, 1st ed., January 2012. `http://accessengineeringlibrary.com.proxy.lib.chalmers.se/browse/programming-arduino-getting-started-with-sketches/c9780071784221ch01`. Read: 2016-05-02. ISBN: 978-0071784221.

[6] Arduino, "Download the Arduino Software." Published on the official website: `https://www.arduino.cc/en/Main/Software`, 2016. Read: 2016-03-14.

[7] Arduino and "SM", “Getting Started with Arduino and Genuino on Windows.” Published on the official website: `https://www.arduino.cc/en/Guide/Windows`, February 2016. Read: 2016-03-14.

[8] Arduino, “Compare board specs.” Published on the official website: `https://www.arduino.cc/en/Products/Compare`, 2016. Read: 2016-03-14.

[9] Arduino Tutorials, “Which Arduino board is for me?.” Published online: `http://www.arduino-tutorials.com/which-arduino-board-is-for-me/`. Read: 2016-03-14.

[10] Koninklijke Philips Electronics N.V., Amsterdam, the Netherlands, *PN532/C1 NFC controller short form datasheet*, Revision 1.2 ed., March 2011. `https://www.adafruit.com/datasheets/pn532ds.pdf`. Read: 2016-03-14.

[11] NFC Forum, 401 Edgewater Place, Suite 600 Wakefield, MA 01880, USA, *NFC Forum Technical Specifications*, 2016. `http://members.nfc-forum.org/specs/spec_list/`. Read: 2016-03-14.

[12] NFC Tools, “ISO14443.” Published online: `http://nfc-tools.org/index.php?title=ISO14443`, February 2015. Read: 2016-02-06.

[13] NXP Semiconductors, “MIFARE.” Published online: `https://www.mifare.net/en/`, 2016. Read: 2016-03-05.

[14] NXP Semiconductors B.V., Eindhoven, the Netherlands, *Application Note AN10927: MIFARE and handling of UIDs*, revision 3.1 ed., October 2013. `http://www.nxp.com/documents/application_note/AN10927.pdf`.

[15] Flavio D. Garcia et al., *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings.*, vol. 5283 of *Lecture Notes in Computer Science*, ch. Dismantling MIFARE Classic, pp. 97–114. Berlin, Heidelberg: Springer Berlin Heidelberg, October 2008. `http://link.springer.com/chapter/10.1007%2F978-3-540-88313-5_7`. DOI: 10.1007/978-3-540-88313-5_7. ISBN: 978-3-540-88313-5.

[16] Huanan Electronic Technology Co., Ltd., 4/F, Block B, Anhongji Technology Industrial Estate, Min Zhi Street, Longhua District Shenzhen City, Guangdong Province, China, *Specification for LCD module no.: HTF0177SN-01*, 1st ed., June 2012. `https://www.arduino.cc/documents/datasheets/A000096_Datasheet_HTF0177SN-01-SPEC.pdf`. Read: 2016-05-11.

[17] Encyclopædia Britannica, "Bluetooth." Published online: `http://academic.eb.com/EBchecked/topic/1191284/Bluetooth`, 2016. Read: 2016-03-14.

[18] Guangzhou HC Information Technology Co., Ltd., "HC-06." Published online: `http://wavesen.com/mysys/db_picture/news3/2015121885021101.pdf`, 2015. Read: 2016-03-14.

[19] Digi, "XBee® Zigbee." Published online: `http://www.digi.com/products/xbee-rf-solutions/rf-modules/xbee-zigbee`, 2015. Read: 2016-03-14.

[20] Stefano Tennina et al., *IEEE 802.15.4 and ZigBee as Enabling Technologies for Low-Power Wireless Systems with Quality-of-Service Constraints.* SpringerBriefs in Electrical and Computer Engineering, Heidelberg: Springer Heidelberg, 2013. `http://link.springer.com/book/10.1007%2F978-3-642-37368-8` DOI: 10.1007/978-3-642-37368-8. ISBN: 978-3-642-37367-1.

[21] Digi, "Module Firmware Revision History." Published online: `http://knowledge.digi.com/articles/Knowledge_Base_Article/XBee-XBee-PRO-Series-1-ZigBee-Module-Firmware-Revision-History`, October 2007. Read: 2016-03-14.

[22] Digi, "XBee transparent mode." Published onlinne: `https://docs.digi.com/display/WirelessConnectivityKit/XBee+transparent+mode`, 2015. Read: 2016-05-04.

[23] Digi, "What is API Mode and How Does it Work." Published online: `http://knowledge.digi.com/articles/Knowledge_Base_Article/What-is-API-Application-Programming-Interface-Mode-and-how-does-it-work`, 2015. Read: 2016-05-10.

[24] Digi, "Next Generation Configuration Platform for XBee/RF Solutions." Published online: `http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu`, 2016. Read: 2016-05-09.

[25] Don Wells, "The rules of Extreme Programming." Published online: `http://www.extremeprogramming.org/rules.html`, 1999. Read: 2016-04-12.

[26] Fazecast, "Jserialcomm." Published online: `http://fazecast.github.io/jSerialComm/`, 2015. Read: 2016-05-09.

[27] International Data Corporation, "Apple iOS." Published online: `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`, Smartphone OS Market Share, 2015 Q2. Read: 2016-05-09.

These pages have been left blank for printing purposes