

Hari Shrestha

DASH7-Based Indoor Navigation System

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

15 April 2014

Author Title	Hari Shrestha DASH7-Based Indoor Navigation System
Number of Pages Date	45 pages + 17 appendices 10 February 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software and Embedded Engineering
Instructor(s)	Anssi Ikonen, Supervisor
<p>The population of senior citizens has been increasing steadily in most countries, which has resulted in various challenges in maintaining their living standards. Both private and government sectors are looking for the best possible ways of cutting down operation costs while trying to improve their services for senior citizens at the same time. Developing an indoor navigation system is one of the best solutions to cope with these challenges.</p> <p>The purpose of the project was to improve the living standard of elderly people with the development of an indoor navigation system that would navigate and provide security and immediate help to senior citizens living in elderly care homes and rehabilitation centers. Support is provided to them by surveillance of their movements and studying their behaviour. The project aimed to create and develop a desktop application for controlling the indoor navigation system using the DASH7 technology and the platform-independent object-oriented programming language. The desktop application for an indoor navigation system was developed using the Java programming language and the cell-based principle. DASH7-based active RFID hardware devices, Beacon, Mobiletag and Access point, were used for data communication purposes.</p> <p>The objective of the project was achieved and a desktop application for an indoor navigation system was developed and it fulfilled the customers' requirements. It is possible to navigate and track elderly people inside a building. The system uses panic button for handling emergency situations. The outcome of the project can be used in different areas of an indoor navigation system.</p>	
Keywords	DASH7, INS, RFID, RSSI, GUI, Radio Communication, Wireless Technology, Mobiletag, Beacon, Cell-based System

Acknowledgements

Proper guidance and coordination is very important for the successful completion of project work. I am extremely fortunate to have got this all along the completion of my project work. Without such wonderful guidelines, efforts and assistance from many individuals, the successful completion of the project work would have been almost impossible. I would like to extend my sincere thanks to all of those who contributed, in one or another way, towards the successful completion of this project.

First and foremost, I would like to thank and respect the head of the project, Mr. Asko Kippo, for giving me an opportunity to do the project work of developing an Indoor Navigation System for elderly people living in care homes and rehabilitation centers, and providing us with all the support and guidance which made me complete the project on time. Thank you for trusting me with your project idea. I will always remember these words you said to me during the interview, "You have a good education record, you can do this project, I am sure. I have the confidence in you, you will produce good results".

Secondly, special thank go to my team mate, Mr. Wasswa Charles Sewagudde for helping me and sharing his wonderful experience towards the successful completion of the project. His presence in the team made the working environment more international and made me more comfortable and inspired towards the work. Thirdly, I wish to extend my thanks to the language adviser, Ms. Taru Sotavalta for helping me correct this document to meet the standards of an academic paper. Fourthly, I wish to extend my thanks to the thesis supervisor Mr. Anssi Ikonen for being my supervisor and helping me to coordinate the thesis paper.

Last but not least, many thanks go to my supervisor, Mr. Sampo Nurmentaus who has given his full effort in guiding the team in achieving the goal as well as his encouragement to maintain our progress in track. I would also like to appreciate his kind efforts for helping me to find a job after the completion of the project work.

“We never know how far reaching something we may think, say or do today will affect the lives of millions tomorrow.” (B.J Palmer)

Contents

1	Introduction	1
2	DASH7 Technology	2
2.1	DASH7 and Feature Comparison	3
2.2	Near-Field Communication	4
2.3	Real-Time Location System	5
2.4	Radio Signal Strength Identification	5
2.5	Radio Frequency Identification	6
3	Indoor Navigation System	7
3.1	System Development	8
3.1.1	Conceptual Design	8
3.1.2	System Design Overview	9
3.1.3	Requirements and Specifications	10
3.1.4	Hardware	12
3.1.5	Software	13
4	System Prerequisites	13
4.1	System Requirements for PCs	14
4.2	Windows vs. Linux	15
4.3	Serial Communication API Library	16
4.4	Guidelines and Installation	16
4.4.1	Installation of JavaComm on Windows	16
4.4.2	Installation of RXTX on Linux	18
4.4.3	Installing JavaComm on Linux	19
4.5	System Architecture and Data Acquisition	20
5	Positioning Algorithms	21
5.1	Triangulation	22
5.2	Trilateration	23
5.3	Fingerprinting	24
5.4	Cell-based System	25
6	Main Principle and Concept	26
6.1	Cell-based System	26

6.2	Gate Concept	27
6.3	Machine Learning	27
6.4	Auto-discovery	27
7	Desktop Application	28
7.1	Requirement Specifications	28
7.2	Programming Language and Development Tools	28
7.3	Memory Management and Optimization	29
7.4	Java Serial Communication (RS232)	30
7.5	Flow Diagram of Desktop Application	32
7.6	Graphic User Interface (GUI) Design	33
8	Results and Discussion	34
8.1	Implementation	34
8.2	Algorithms	35
8.3	Challenges and Solutions	37
8.4	Achievements	39
8.5	Project Development	40
8.6	Testing	41
8.7	Final product	42
8.8	Future Development	43
9	Conclusion	44
	References	45
	Appendices	47
	Appendix 1: Graphics User Interface for User Position and Alarm View	47
	Appendix 2: Use Case Diagram and Class Diagram	48
	Appendix 3: Java Source Code for Main GUI Design	49
	Appendix 4: Serial Data Reading, Filtering and Selecting Based on RSSI Value	52
	Appendix 5: Main Method Java Source Code	54
	Appendix 6: Structure of Process Data	54
	Appendix 7: Example Source Code View Panel	56
	Appendix 8: Graphics Thread for Updating the Floorviews	57
	Appendix 9: Source Code for Mobicat Positioning Data Structure	58
	Appendix 10: Source Code for Updating the Mobicat Position	59
	Appendix 11: Source Code for Handling XML Files	62

1 Introduction

Senior citizens cover a major portion of society, they bear valuable knowledge gathered from decades of life experience, and they could play important roles for the development of a nation. So, it is the duty and responsibility of all of us to give them the virtue of respect, maintaining their health, stabilize their lives, their rights and interests and improve their social welfare. The number of senior citizens is growing every day and the cost of looking after them is growing exponentially. Both private and government sectors are looking for the best ways of cutting down living costs in elderly care homes. It is a call to us to innovate and find a solution to this challenge.

The purpose of this project was to develop the living standard of elderly people, specially living in care homes and pilots by providing information, support services and personal assistance to senior citizens, their caregivers and loved ones by locating them inside the building, their activities and their conditions along with maintaining their privacy and personal lives. The application developed in this project can be used for locating the people in private homes, hospitals, nursing homes, museums, and shopping centers.

The main objective of this report is to provide the details about the project and highlight the technologies and the procedures that were implemented in the project. The scope of the project was limited to developing a navigation system for building and providing emergency help to the residents of a specific care home and pilots.

2 DASH7 Technology

DASH7 is an open source wireless sensor networking technology based on the ISO 18000-7 standard. DASH7 uses ultra-low energy that provides multi-year battery life and ranges of up to 2 km using the license-free 433 MHz ISO band. DASH7 is an ISO 18000-7 standard band for active RFID that is similar to the use of the Wi-Fi brand for IEEE 802.11 communication. DASH7 is a new technology promoted by the non-profit consortium called the DASH7 Alliance which is getting popularity in the current market of the IT industry and is available worldwide. [1]

DASH7 uses the radio signal transmission technology for communication purposes, and can provide the transmission ranges of up to 2 km depending on the transmitters output power supply. Higher output power supply to transmitters increases the communication between the receiver and transmitter at further distances. DASH7 utilizes the globally available and license-free frequency of 433.92 MHz frequency. This frequency is ideal for wireless sensor network applications because of its advanced features such as communication ability to transmit and receive signals over longer ranges of distance without consuming a large power drawn on a battery and ability to penetrate concrete and water. [1]

DASH7 used the concept of the BLAST networking technology. The full form of BLAST is Bursty, Light-data, Asynchronous and Transitive. Bursty represents the data transfer which is abrupt and that does not include content such as audio, video or other isochronous forms of data. The term light refers to the size of packets which are limited to 256 bytes for most of the application. Asynchronous indicates the method of communication. [2, 5] DASH7's communication is done by command-response that requires no periodic network "hand-shaking" or synchronization between devices. A DASH7 system of devices is fundamentally transitional and does not need to be managed extensively by a fixed infrastructure such as base stations. [2, 6]

2.1 DASH7 and Feature Comparison

DASH7 uses the worldwide acceptable and globally available 433 MHz frequency for its communication purposes. It can provide the maximum outdoor ranges of between 200 meters and 2 kilometres depending upon the location and transmitting power supply. DASH7 has excellent signal propagation inside the building that allows it to penetrate walls, bend around HVAC (heating, ventilation, and air conditioning) ducts, and more, unlike competing technologies. [1] Figure1 shows the feature comparison of DASH7 and different competing wireless communication technologies.

	Range	Battery Life	In Building Coverage	Low Latency	Multi-hop	Co-existw 802.11n	Penetrates Concrete	Penetrates Water	"Bends" Around Metal	Tracks Moving Things	Security	Globally Available	Small Protocol Stack	Data Rate	Major Customers
DASH7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Zigbee	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
LE Bluetooth	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
WiFi	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✓	✓
Low Power UWB	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗




 Good
  Fair
  Poor

Figure 1: Feature comparison of different wireless communication technologies [2]

According to the feature comparison table of figure 1 above, DASH7 offers a mix of features unlike any other low-power wireless networking standard. However the DASH7 data rate of transmission is lower in comparison to competing technologies.

2.2 Near-Field Communication

Near-Field Communication, abbreviated NFC, is a standard for the very short-range high frequency wireless communication technology of radio transmission that enables the exchange of data between devices by touching or bringing them into proximity, usually over about a 10 cm distance. NFC standards cover communications protocols and data exchange formats, and are based on the existing radio-frequency identification (RFID) standards including ISO/IEC 14443 and FeliCa. FeliCa is a contactless RFID smartcard system from Sony in Japan, primarily used in electronic money cards. [10]

NFC is an upgrade of the existing proximity card standard (RFID) that combines the interface of a smartcard and a reader into a single device. It is used for different purposes such as sharing the contents between digital devices, wireless and contactless bills payment systems, and electronic travelling tickets integrated into a digital device such as a phone. It allows users to use their smartphones as an electronic travelling ticket on existing contactless infrastructure that is already in use for public transportation. Information and data can be saved and stored in the passive NFC tags that can be read by an NFC-enabled device such as mobile phone. [10]

Due to its shorter range, NFC provides a higher degree of security than a Bluetooth and makes NFC suitable for crowded areas where correlation of a signal with its transmitting physical device might otherwise prove impossible. Apart from these, an NFC connection between two NFC devices is established at once under 0.1 second, which is a shorter set-up time in comparison with Bluetooth. [10]

2.3 Real-Time Location System

An efficient and effective means of tracking resources has been a long-standing and continuous issue for enterprises. Different types of wireless technologies have been introduced to facilitate such a tracking system. Among them, a radio frequency (RF) based wireless tracking system has become more advanced and highly used for achieving the efficient and effective means of tracking resources. The different forms of resources come from high valued assets, daily used products, customer goods, animal farming, semi-finished goods, and raw materials to tracking of people. [12, 16]

Real-Time Location System, also called RTLS, typically refers to the tracking and locating resources in real time by using radio frequency signals and sensors. With the use of an RTLS system, tracking and locating equipment become easier along with higher safety of equipment. The device needed to be tracked can be provided some RTLS-tags, so their location can be monitored. There are different types of technologies that RTLS-solutions follow, such as light, camera vision, infrared, sound, ultrasound, cellular, Wi-Fi, Bluetooth, ultra wideband, RFID and GPS and many more technologies. With these technologies, RTLS systems can offer solutions of tracking at an enterprise level, for both indoors and outdoors tracking solutions. [12, 15]

2.4 Radio Signal Strength Identification

Radio Signal Strength Identification, abbreviated RSSI, is the unit of power level indicated by the receiver antenna. According to an IEEE 802.11 system, RSSI is the relative received signal strength in a wireless environment, in arbitrary units. The higher the RSSI value, the higher the signal strength which results in better communication through the radio segment. There is no standardized relationship of any particular physical parameter to the RSSI reading. Therefore, no signal variable can be used to provide accurate ranging estimation under all circumstances. Moreover, some of the chipset maker companies, including vendors, use their own accuracy and range for actual power and RSSI values. The actual power is usually measured as mW or dBm and the range of RSSI values are starting from 0 to RSSI_Max. [13]

2.5 Radio Frequency Identification

Radio Frequency Identification (RFID) is a wireless communication technology that uses a wireless non-contact radio system to transfer data from a tag attached to an object, for the purpose of automatic identification and tracking. In terms of the power used, RFID tags can be categorised into three different types: active RFID tags, passive RFID tags and semi-passive RFID tags. Active RFID tags always need power to operate whereas passive tags require no battery and are powered by the radio waves used to read them from the reader. Passive tags rely entirely on the reader as their power source. A semi-passive tag does not require power all the time even though it contains more hardware than a passive RFID tag. It can be used similarly to the passive tag. The battery is used only either to monitor environmental conditions or to offer greater range and reliability but not to generate RF energy. [12, 29]

The RFID technology is similar in theory to bar code identification. The main key difference between the RFID technology and the bar code identification is that RFID does not use line-of-sight for reading and identifying tags whereas line-of-sight is essential for the bar code identification technology. Apart from this, RFID scanning can be done at greater distances than barcode scanning. The tag contains two parts of RFID components: a microchip and an antenna. Microchip stores and processes the information whereas the antenna receives and transmits the signal. The unique serial number for one specific object is assigned to the tag.

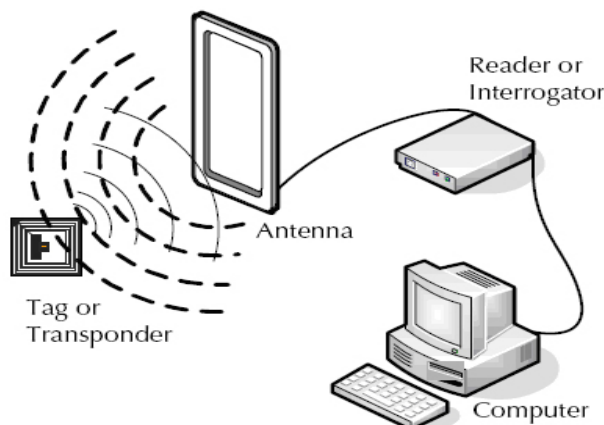


Figure 2: RFID system architecture [14]

Figure 2 shows how the RFID system communicates between the transmitter and the receiver.

In order to read the encoded tag information, an interrogator or reader uses an antenna to emit a signal to the tag. The interrogator is also called a two-way radio transmitter-receiver that can receive and transmit the signal. The tag responds to the interrogator with the information encoded in its memory bank. The interrogator or reader receives the signal responded by the tag and transmits the result to the RFID-enabled computer program. [14]

3 Indoor Navigation System

An indoor navigation system (INS) is a system of network devices used to wirelessly locate and navigate the people or objects inside a building. [7] Generally the indoor navigation system navigates the position on the objects or the people on the basis of a real-time location system (RTLS). An indoor navigation system does not use the Global Positioning System (GPS) data collected from the satellite because of the signal attenuation and reflection caused by the construction material. An indoor navigation system uses different kind of indoor transmitters and the receivers that work through the wireless communication technology. Different kind of wireless communication technologies such as Bluetooth, Wi-Fi, Worldwide Interoperability for Microwave Access (WiMAX), ZeeBee, RFID and DASH7 can be used for the development of an indoor navigation system. In this project, the DASH7 technology was used for the development of an indoor navigation system. [5]

3.1 System Development

System development is the process of defining, designing, testing and implementing a new software application or program. It is the conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the complete application.

There are several types of System development life cycle (SDLC) methodologies such as the Waterfall model, the Spiral Model, or the Agile Model used in system development process.

3.1.1 Conceptual Design

During the first phase of the project development process, a conceptual design was prepared to overcome the possibilities of challenges along with the estimation of cost and time in order to enhance the feasibility and functionality of the final product. The Conceptual design is an important phase in the project development stage because it provides sufficient information of the project, its operation and the developing process, which help in the distribution of the work among the developers. The estimation of cost and time for the project can be determined through the conceptual design. In this project, the conceptual design for the software development of an indoor navigation system for desktop computer was prepared based on four main principles:

- Cell-based system
- Gate concept
- Machine learning
- Auto discovery

The navigation system will use the specific algorithm called Cell-based system for determining the position of the user inside the building. The Cell-based system is the technology used in wireless mobile phones or cell phones to track the user in the networks. It uses the connected transmitter or cells that allow the user to move and locate while remaining in contact with the network or cell. Basically the cell-based system used for indoor navigation provides the location of a person in a certain cell or room instead of giving the pin-point of that position. The reason behind this is that in the

DASH7 technology, the Radio Signal Strength Identification (RSSI) value of the received signal is directly proportional to the length between the transmitter and receiver, which is not 100 percent true in real life. The RSSI value gets affected by many environmental factors such as reflection, position, angle of orientation, diffraction, or obstacles. This system uses the gate concept each room has a gate for getting in and out from the room. Similarly, the system uses certain rules for movements. The system also uses the auto-discovery mechanism to detect the position when it fails to read the consecutive Beacons.

3.1.2 System Design Overview

The system design consists of three hardware devices called Beacon, Mobiletag and Access-point. The hardware parts of the Beacon, Mobiletag and Access-point are the same. The software used to program these devices is different, so that they behave differently. The main function of Beacon is to transmit the signal that contains the unique ID. Beacons are placed in a certain position in the building such as in the rooms' floors, kitchen, bathroom and corridors. Mobiletag function is to receive and collect the signal transmitted by the Beacons and transmits the collected signal in to the Access-point in the form of packets. The packet contains the information of the unique Mobiletag ID and the collected Beacons with its corresponding RSSI value. Access-point is connected to the computer. Figure1 below shows the overall system design overview of the project.

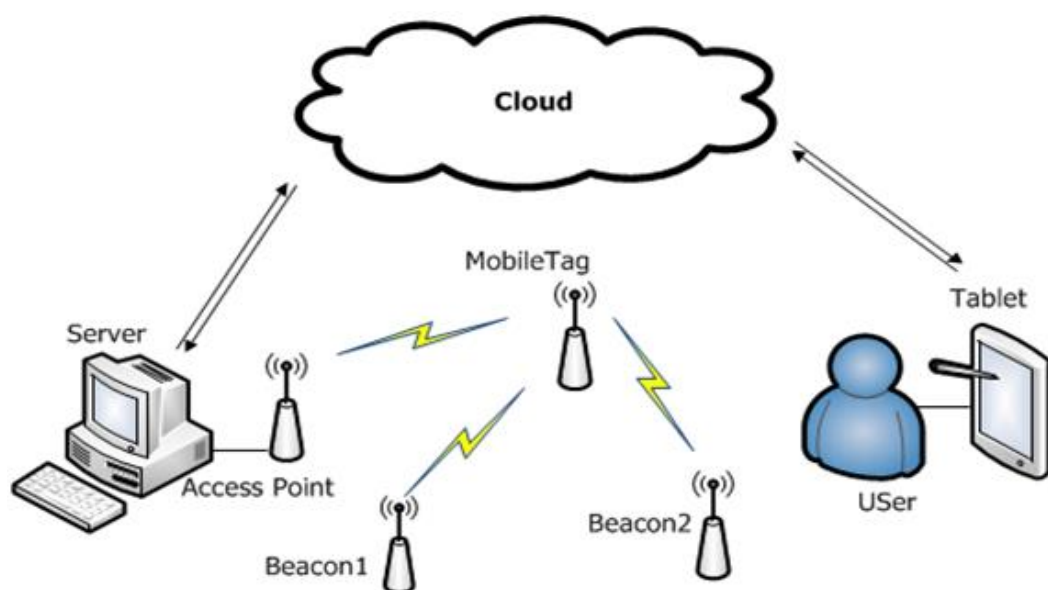


Figure 3: Indoor Location System Overview

When the packet data is received by the Access-point, the packet is sent to the computer through serial communication. The computer will filter the received data and process the data for further use such as GUI application on desktop and data to the cloud server. The cloud server communicates and distributes the filtered data to the other devices such as tablets and smartphones for multiple uses, so that the devices can have the same system at the same time.

3.1.3 Requirements and Specifications

Requirements and specifications are very important components that play significant roles in the development of any project system. In the system design process, requirement analysis is the first step to clarify the user's requirements and to prepare documents to generate the corresponding specifications. Well-documented and clarified requirements of system with the involvement of the customer are very important for the development of a safety-critical system. The stage of the requirements and specifications has a significant impact on the downstream results in the development of system life cycle. It is because any errors developed during the requirements and specifications stage may lead to errors in the design stage. The engineers and developers are working based on the requirements and specifications. When the errors are found, the engineers will have to revisit the requirements and specifications to fix the problems. Fixing those errors consumes more time and also increases the possibility of other requirement and specification errors. Therefore, it is very important that the requirements are specified correctly for generating clear and accurate specifications. [19]

Some requirements of the project that were identified from the verbal description of the project from my supervisor and customer included:

- the system should navigate the people inside the building
- the system should provide the users with information and the current location
- the panic button should be implemented so that the user could use it for asking for help
- the system should restrict people from moving to restricted area
- the important information of the user should be stored in some database
- the device size should be small enough to be easily carried by the users

It is now clear that the main requirement of the project is to develop a system that would help the nurse to know the status of the elderly people inside the building. The system developed in the project should navigate the users' position and their information. The panic button should be implemented to facilitate the elderly people to call for help by pressing the button during an emergency situation. The nurse should get and acknowledge the alarm coming from the users. The system should also prevent the users from being entering the restricted places. The requirements for the device and the software are discussed under the section 3.1.1 and 3.1.2.

3.1.4 Hardware

The project contained three hardware devices: Mobiletag, Beacon and Access-point. The PCB and the components of these devices are the same. The same device can be used for programming the Mobiletag, Beacon and Access-point. The size of the hardware device for the Mobiletag should be small enough to be easily carried by users in the form of a wrist watch or identification card or door key. Figure 4 shows the PCB design that can be used for programming Beacon, Mobiletag and Access-point.

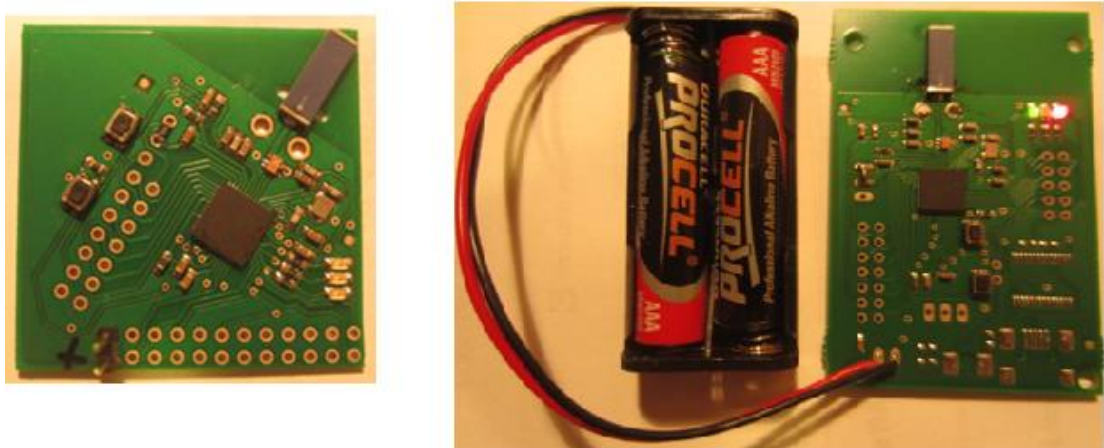


Figure 4: PCB design for hardware device

Figure 4 shows the PCBs designed for the Mobiletag, Beacon and Access-point. The device consists of a microcontroller, radio chip, RF transmitter, antenna, USB interface, led and the buttons. The only difference between the two PCBs design is the size which can be used to program any hardware devices such as Mobiletag, Beacon and Access-point needed for the project. The left-hand side PCB device is designed for the Mobiletag since the size of the Mobiletag should be small to be used by the user in the form of either a wrist watch or door key. The right-hand side PCB is designed for Beacon and Access-point. The blinking led can be used for debugging and testing purposes. For instance, the green light can represent the receiving signal and the red light can represent the transmitting signal. DASH7 is a part of active RFID that needs power all the time to operate. The device can be operated by using two small batteries of 1.5 volts and it gives the battery life of up to 2 years.

3.1.5 Software

Software is regarded as the non-intangible components of a computer and an asset of the business. Computer hardware and software are regarded as the two tyres of a bicycle. One needs the other and neither can be realistically used without the other. Software is the clearly defined instructions for the hardware to perform a task accordingly upon execution. Computer software includes all computer programs including its architecture and all the related files, for example, executable files, libraries and scripts. High-level programming languages are more efficient, human-readable and easy to use and often require much less work in comparison to low-level or machine language. Because of this reason, software is usually programmed using a high-level language. High-level languages are compiled and interpreted into the machine language object code. Software written in a low-level assembly language needs an assembler to convert the assembly language into the object code. The object code is the metadata for the computer or also called the machine language used by the computer.

In this project work, there were no any specific requirements regarding the software for the desktop application. The desktop application could be developed using any high-level programming language that would support multiple platforms and run on a normal computer. The Java programming language was used for the development of the software for the desktop application, since Java supports most of the platforms and is free software including development tools. The programming language and the development tools required for developing the application are discussed under the section 7.2.

4 System Prerequisites

System prerequisites are the process of verifying the requirement of the system environments to develop and run the applications. The process includes the existing installation of the libraries, the operating system processor and space requirements, hardware, requirements of the hard disk space and software. It ensures that the environment required for running and installation of software is configured and the prerequisite software exists in order to have the system application running.

4.1 System Requirements for PCs

The desktop application can run on any computer regardless of the operating system. The desktop application was developed using the Java programming language which supports most of the operating systems such as Windows and Linux operating system. JavaComm API libraries or RXTX API libraries is required for serial communication and had to be used on all the operating system. Similarly, Java development kit tools (JDK), and Java Runtime Environment (JRE) are required for running Java application and had to be installed for developing the Java application on all operating systems. Java Runtime Environment is compulsory to be installed on a computer for running the Java program or developing the Java application whereas JDK is only needed if the Java application has to be developed. During the JDK installation, JRE is automatically installed. Table 1 shows the requirement to install the JDK and JRE on Windows OS.

Table 1: Windows system requirement for JDK and JRE [15]

Requirements	32-bit Platform	64-bit Platform
Processor for both JRE and JDK	minimum a Pentium 2 266 MHz processor	minimum a Pentium 2 266 MHz processor
Disk Space for JRE		
Java Runtime Environment, including JavaFX Runtime	124 MB	124 MB
Java Update	2 MB	2 MB
Disk Space for JDK		
Java Development Tools, including JavaFX SDK	245 MB	245 MB
Source Code	27 MB	27 MB
Public Java Runtime Environment	128 + 2 MB	128 + 2 MB
Memory Requirements		
Windows 7, 8, Vista, Server 2008	128 MB	128 MB
Windows XP	64 MB	128 MB

Table 1 above shows the requirements that differ on different platforms. The requirements are similar in the case of the Linux OS. The recommendation is to check the Oracle documentation from the official website www.oracle.com.

4.2 Windows vs. Linux

The Windows and Linux operating systems have some different features and supports different libraries. Only the differences are highlighted that were discovered during the project development. The differences are on API libraries, port naming and the selection process. The JavaComm API library supports only Windows but does not support Linux, so for the Linux OS, a third party RXTX API library has to be used. The RXTX API library can be used for both the Windows and Linux OS. The installations of these libraries are different for different operating systems. The installation guidelines for some of the libraries are discussed in section 4.4.1, 4.4.2 and 4.4.3.

By default, Windows always assigns a COM port number every time when a new USB device is connected and the port number goes on increasing as a new device is connected. This happens even if the same device is plugged into a different USB port. Windows reserves that port and the next time it assigns the same port number when an already used USB serial device is detected. The naming of the port number starts with COM0, COM1 and so on. In the case of Linux, the port reservation is not possible, and the naming of serial ports on Linux starts with ttyS0, ttyS1 and so on. The feature of port reservation has some advantages and disadvantages when an automatic connection is needed for the system to run. For instance, it is very difficult to assign the correct port to the serial device on Linux, since the ports are always changing when the devices are connected and disconnected. The connection does not start unless the right ports are selected. So, automatic selection of the used port is difficult in Linux whereas Windows facilitates the automatic selection of the used port.

4.3 Serial Communication API Library

Java supports communication to serial ports that require an external library to be installed. Recently, there are two options available for achieving serial communication: RXTX and JavaComm. JavaComm is the official API library for serial communication in Java that enables a Java application to get a low-level access to the serial port on a PC but has not been implemented for all platforms. Serial interfacing requires a standardized API with platform-specification implementations, which is difficult in the case of Java. Initially, JavaComm was not implemented for Linux until 2006 which led to the development of the free software RXTX library. [18] RXTX has been adopted to provide the same interface as JavaComm which is only different in the package name. RXTX is available for a number of platforms, not only for Linux. RXTX supports more platforms than JavaComm and can be used in conjunction with JavaComm or can be used stand-alone. [17]

4.4 Guidelines and Installation

Installation of the serial communication API library, RXTX and JavaComm will have to follow the installation rules accordingly in order to make successful installation. The guidelines for the installation of the JavaComm API library on a Windows machine and RXTX on both a Windows and Linux machine will be described in detail in sections 6.4.1, 6.4.2 and 6.4.3. During the project development, the API library was installed and tested.

4.4.1 Installation of JavaComm on Windows

The Windows version of JavaComm is no longer officially available because it is not available in the Java products archive. However, the 2.0 Windows version of JavaComm is still downloadable from the link: <http://wind.lcs.mit.edu/download/>. [18]

There are certain rules for the installation of Java Communications API that have to be placed on a Windows system machine. There are some core files of Java Communication API which are very important to install on the system for a proper and successful operation. These core files include:

- comm.jar
- win32com.dll
- javax.comm.properties

The installation files are very important and should be properly placed in a specific place on the machine according the installation rules. These files are important for Java Development Kit tools (JDK) to communicate with the serial port of the machine.

%Java_HOME% is the location of JDK directory in the local machine. Another way for finding the JDK path is to navigate through the C folder for instance

```
C:\Program Files (x86)\Java\ in case of 32-bit Windows OS or
```

```
C:\Program Files\Java\ for 64-bit Windows OS
```

The required JDK and JRE files are placed inside the Java folder.

comm.jar are required for application access to the RS-232 hardware and the serial port that has to be copied in the following directories:

```
%JAVA_HOME%/lib and
%JAVA_HOME%/jre/lib/ext
```

win32com.dll are needed in the Window OS for communication with serial devices that should be copied in the following directories:

```
%JAVA_HOME%/bin
%JAVA_HOME%/jre/bin and
%windir%System32
```

javax.comm.properties is the properties defined for Java Communication API and should be copied in the following directories:

```
%JAVA_HOME%/lib and
%JAVA_HOME%/jre/lib [18]
```

4.4.2 Installation of RXTX on Linux

The RXTX API library includes: the Java extension library RXTXcomm.jar and the serial driver. The serial driver is integrated with the operation system. The prerequisite for the installation of the RXTX is the Java Runtime Environment (JRE) and the user with 'root', Administrator or similar privileges. The detailed instructions for the installation of the RXTX are explained below.

Step 1: First of all, RXTX bins package can be download from:

```
http://www.linux.org.uk/~taj/rxtx-bins.1.tar.gz
(In IE, right click and "Save Target As")
```

Step 2: Decompress the package and Untar with the command

```
/bin/gzip --decompress rxtx-bins.1.tar.gz
/bin/tar xf rxtx-bins.1.tar
```

Step 3: There is an rxtx-bins.1 directory. Next, copy the shared objects into the Java installation using following commands:

```
cp rxtx-bins.1/1.4/i386-pc-linux/libParallel.so
/usr/java/j2sdk1.4.0/jre/lib/i386/

cp rxtx-bins.1/1.4/i386-pc-linux/libSerial.so
/usr/java/j2sdk1.4.0/jre/lib/i386/
```

Note: Adjust both the /i386-pc-linux/ and the /i386/ accordingly if installation on an architecture is other than an x86 platform.

Step 4: Install the jcl.jar file:

```
cp rxtx-bins.1/1.4/jcl.jar
/usr/java/j2sdk1.4.0/jre/lib/ext/ [17]
```

Installation is finished after step 4. In this project, the RXTX third party API library was used for supporting serial communication. RXTX is an alternative to JavaComm Library for serial communication. The installation guidelines and the library version can differ with different versions of prerequisites software such as JRE, JDK along with the operating system and platform.

4.4.3 Installing JavaComm on Linux

The Java Comm API installation guidelines for the Linux operating system are mentioned below. The installation guidelines may vary with different versions of the software and operating system. So, it is very important to check and follow the installation guidelines and rules for successful installation of the software.

Step1: download javacomm library from the link.

```
http://java.sun.com/products/javacomm/
```

Step2: Decompress and untar the package with command

```
/bin/gzip --decompress javax_comm-2_0_2-solsparc.tar.Z
```

```
/bin/tar xf javax_comm-2_0_2-solsparc.tar
```

Step 3: Install the comm.jar file:

```
cp commapi/comm.jar /usr/java/j2sdk1.4.0/jre/lib/ext/
```

Step 4: To create the properties file that Comm API will use to load the drivers, use the following commands:

```
/bin/echo Driver=gnu.io.RXTXCommDriver >  
/usr/java/j2sdk1.4.0/jre/lib/javax.comm.properties  
[17]
```

The installation commands and guidelines can differ with the installation of different versions of the software, operation system and platforms. The particular library can have different versions and each version of library can support specific platform. Different version of API library installation requires different prerequisites library to be installed. For example, specific version of JDK installed on an operating system does not support the old version of the prerequisites libraries mentioned above in steps 1, 2 and 3 and vice versa. So, it is very important to install the correct version of the libraries and software in the correct place based on the installation guidelines. Sometimes un-installation and installation of the same software is necessary to ensure the correct installation and testing.

4.5 System Architecture and Data Acquisition

One of the first and foremost challenges during the project development phase was to analyse the structure of the input data. The receiver and transmitter communicate with each other by sending and receiving signal data. To analyse the input data types and their structure, data acquisition is needed. Data acquisition is the process of collecting information to document or analyse the raw data for processing. Different tools are available for analysing raw data. RealTerm: Serial Capture was used in the project for analysing the raw data received by the Access-point. In this project, the Mobiletag collects the signal data and transmits those collected signal to the Access-point in the form of a packet. The structure of the packet is shown in the figure 5.

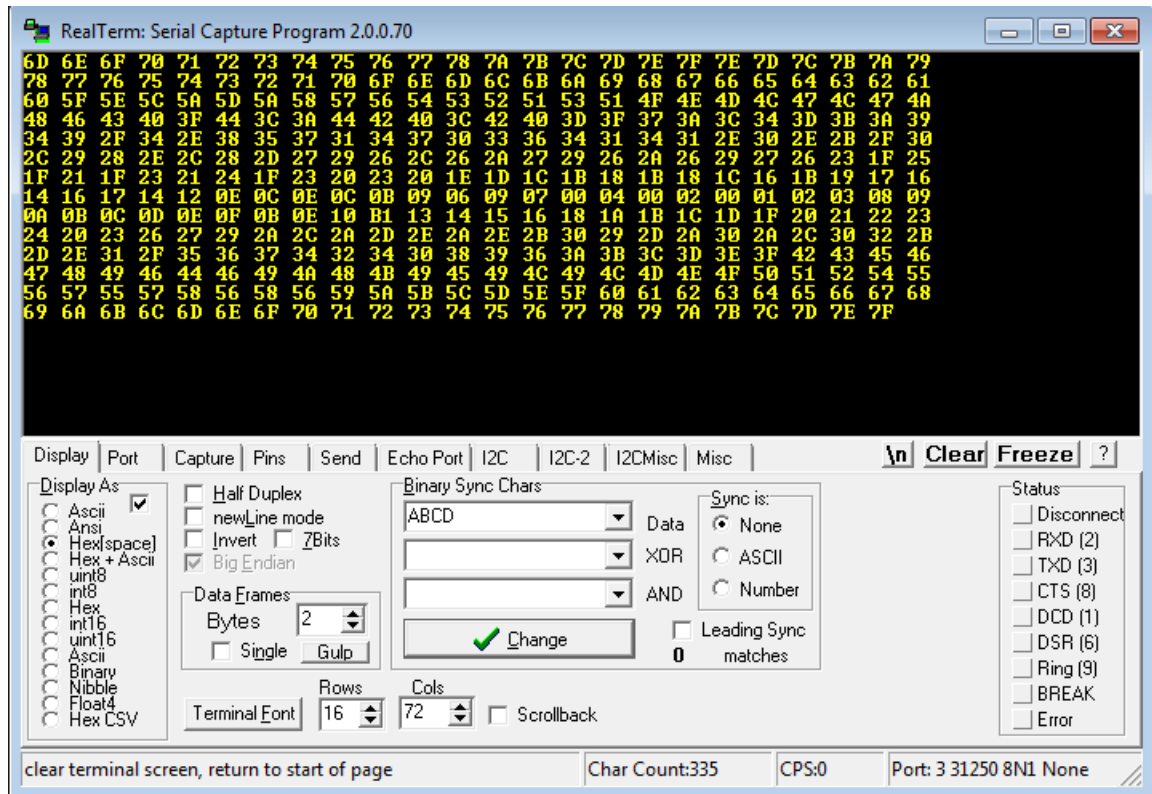


Figure 5: Reading data using RealTerm: Serial Capture program

Figure 5 shows the structure of the received data by the Access-point that can be used for further processing to develop the navigation system. On the screen, each line contains the list of the numbers that are in hexadecimal form, represent the packet. During the data communication process, the devices can transmit the signal data at the mini-

imum interval of 300-400 milliseconds. It means it can transmit the packet data approximately three times in a second. However, packets are received only at an interval of one second. The software can be programmed to the hardware so that the structure of the packet can be defined to facilitate the data processing. In the line, each hexadecimal number represents either the unique ID of the device or the RSSI value.

5 Positioning Algorithms

The need for a location tracking and identification system has been dramatically increased with the increase in the research and development of the standardised wireless technology along with the success of GPS as a consumer navigation application. In addition to the developments in the technologies of tracking and locating the objects and goods, the technology can also be used for tracking an individual in real time which is driving the need for positioning system. There are several algorithms and principles that can be used to compute the position. Some of the algorithms are explained briefly in section 5.1, 5.2, 5.3 and 5.4. In this project, the cell-based system was chosen for the position measurement purpose.

5.1 Triangulation

The triangulation principle is a technique based on the trigonometric proposition that deals with the measurement of the angle of the triangle. It is the technique of determining the point location by measuring the angles to it from the known points at either end of a fixed baseline. According to this principle, if any one side and two angles of a triangle are known, the remaining sides can be computed. Similarly, if the direction of one side is known, the direction of the remaining sides can be determined. A triangulation system consists of a series of overlapping or joined triangles in which the known sides are measured and the remaining sides are calculated from the angles measured at the vertices of the triangles. The vertices of the triangles are known as the triangulation station whereas the side of the triangle, whose length is predetermined, is called the base line. A triangulation network is formed using the lines of the triangulation system that combine together all the triangulation stations. Figure 6 and the mathematical expression show the principle of triangulation.

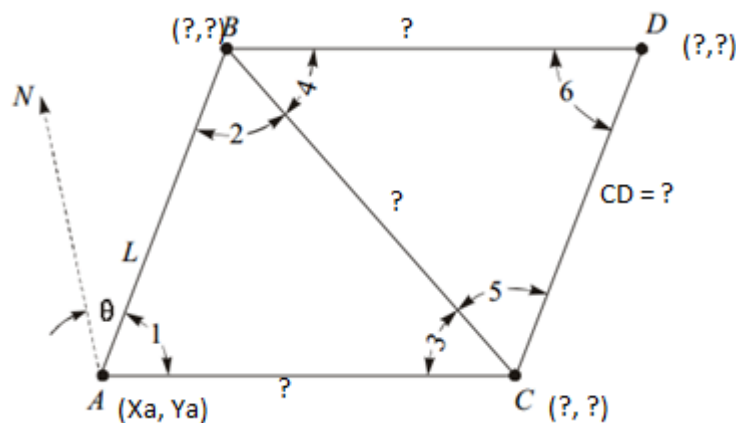


Figure 6: Principle of triangulation [4, 2]

Figure 4 shows two interconnected triangles ABC and BCD. All the angles in both the triangle and the length L of the side AB have been measured. The angle θ of AB has been measured at the triangulation station A , whose coordinates (X_a, Y_a) , are known. So, the coordinates and the distance of the triangulation stations B , C and D can be determined using triangulation. The lengths of all the lines can be calculated by using trigonometric formulas. By sine rule in ΔABC , we have

$$AB/\sin 3 = BC/\sin 1 = CA/\sin 2 \quad [4, 2]$$

5.2 Trilateration

The trilateration principle is the technique for the determining absolute or relative locations of the points by measuring distances, using the geometry of circles, spheres or triangles. In contrast to triangulation, it does not involve the measurement of angles even though it consists of series of joined or overlapping triangles. The trilateration principle can be implemented into two-dimensional (2D) or three-dimensional (3D) space. In 2D space or 3D space, the center of circles and the radii of the circles provide sufficient information to narrow the possible location down to the number of circles where the point lies.[4, 2]

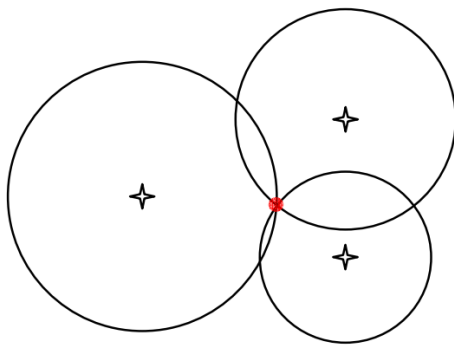


Figure 7: Position estimation by trilateration [20]

Figure 7 shows the principle of trilateration on 3D space. The equations for the three spheres are:

$$r_1^2 = x^2 + y^2 + z^2 \quad (\text{eq. 1})$$

$$r_2^2 = (x - d)^2 + y^2 + z^2 \quad (\text{eq. 2})$$

$$r_3^2 = (x - i)^2 + (y - j)^2 + z^2 \quad (\text{eq. 3})$$

By formulating and solving these equations, the coordinates of unknown points x , y and z and distance d can be determined. In the project, DASH7 also called active RFID, that deals with the RSSI value was used for the estimation of the users position. The following formula in figure 8 shows the relationship between signal strength and the distance.

$$P_r = P_t + 20 \log\left(\frac{\lambda}{4\pi}\right) + 10n \log\left(\frac{1}{d}\right)$$

where

P_t is the transmitter power (in dBm)

P_r is the power at the receiver

λ is the wavelength

n is the path loss exponent ($n = 2$ in free space)

d is the distance between the transmitter and receiver

Figure 8: Relationship between RSSI and Distance [20]

An estimated distance between the device and each transmitter can be obtained using the formulas mentioned in figure 8 and the distance calculated using equations (eq.1) or (eq. 2) or (eq. 3) according to the dimensional space. [20]

5.3 Fingerprinting

The fingerprinting principle is the technique of determining the location of an object by comparing the current measured data value with the already compared and stored data in the database. The fingerprinting principle is also called calibration or training where the actual measurements of data in each and every position of the place where the location has to be tracked are recorded and stored in a database. One of the examples of the data type is the actual measurements of RSSI values from various Access-points. During the runtime process, the current data value, usually the RSSI value is compared with the values stored in the database, and if matched, then it gives the corresponding coordinate or location of that position. In the fingerprinting technology, the location accuracy depends upon the number of fingerprints and the algorithms to find the best match. [12, 191]

The fingerprinting technology is a complex process where a very large number of actual measurement values of different points and the corresponding location of those points are stored in a database. The database has to be updated frequently if some change occurs. [12, 191] Furthermore, the measurement and calculation of the RSSI values and position on each and every point in a facility take a longer time and it is also a tedious process. The RSSI values are not constant and might change frequently due to different environmental factors that affect it when measuring at the same point at different intervals of time. For instance, the transmission and receiving power, angle of orientation, obstacle, refraction and reflection also affect the RSSI value. The Measured RSSI values at a certain point differ with different heights, angle of orientation, signalling time, receiving time, Time of Arrival (TOA), Time of Flights (TOF) and Angle of Arrival (AOA) and many more. [12, 202] These factors have to be considered during the measurement of RSSI values to increase the accuracy in determining the location.

5.4 Cell-based System

A Cell-based system is similar in theory to the cellular network principle. In the cellular network principle, the location of a cellphone is assigned with the location of the serving base station to which the cellphone is connected. The size of the location can be a few hundred meters to kilometres depending upon the cell size and the density of cell towers. So, it is almost impossible to give the pinpoint location of the cellphone using the cellular network principle. In the cell-based principle, a DASH7 device called the Beacon represents the serving base station whereas the Mobiletag represents the cellphone. Mobiletag is assumed to be near the Beacon that receives the largest RSSI value. Mobiletag receives and collects most of the receiving signals from the Beacons which are placed inside the building, and transmits the signal to the Access-point in the form of a packet for the location determining purpose. Figure 9 shows the principle of cell-base system.

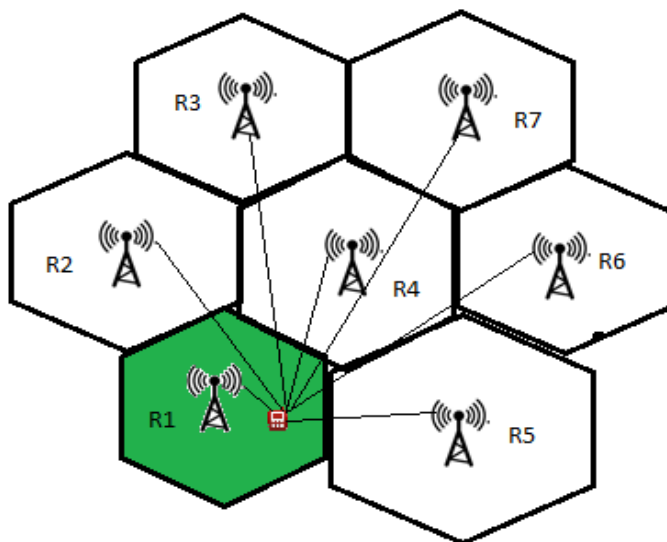


Figure 9: Cell-based principle

In figure 9 above, each hexagon circle represents the cell or room and the base station represents the Beacon. The highest RSSI value is calculated and the position of the Mobiletag is assigned with the position of the Beacon that has the highest RSSI value. The green colour shaded part is the position of the Mobiletag since the RSSI value is higher among other cells. So, instead of giving the pinpoint position, the cell-based principle can only give the position of the person for example a room, a kitchen, a bathroom or in the corridor. The location accuracy can be increased by increasing the number of Beacons.

6 Main Principle and Concept

The DASH7 based indoor navigation system is based on a certain principle and concept for the development of an application of an indoor navigation system. The implementation of the application should follow the basic principle and concept to ensure that the system functionality works as expected. The implementation can be done for different applications such as web application, desktop application, and mobile application that run on different devices and platforms. In this report, the application developed for a desktop computer has been discussed.

6.1 Cell-based System

The RSSI value is the only main input data for the development of the system, and the measurement of the RSSI value is irregular and inconsistent. Theoretically, the signal strength is higher when the distance between the transmitter and receiver is closer and vice versa. That is not always true in practice. The RSSI value changes all the time due to different physical and environmental factors which was discussed in previous sections 2.4 and 5.4.

The triangulation and trilateration principles of position algorithm were used first for the development of the application. However, the triangulation and trilateration principles deal with the angle and the distance measurement procedure, which is very inappropriate due to the inconsistent nature of the RSSI input data. The fingerprint principle could have been used, but it consumes more time and is not suitable for a changing environment. Furthermore, the database would have to be rebuilt every time when some changes happened inside the building. In contrast to these principles, the cell-based principle, which does not entirely depend on the accurate RSSI value, seemed more a reliable principle. It only needs the highest RSSI value measured by the Mobiletag.

6.2 Gate Concept

The Gate concept was used to improve the navigation system that could help to filter and distinguish the user position inside the room and corridor. Beacons are placed inside the building in most places where the position has to be tracked. The concept of gate was applied to indicate the system that the person has entered the room and has got outside from the room after a certain time. So, the gate concept filters out the reading of the neighbouring room Beacons and corridor and gives the correct information to the system. In order to implement this concept, the Beacon will have to be placed and installed in every entry point of the room and the user will be allowed to enter the room or get out of the room only through the doors.

6.3 Machine Learning

The system is bounded with certain rules and regulations that have to be followed. The system allows the user to move to a certain place and it can also prevent the user from entering the restricted area. An alarm system was implemented for the notification of user movement to the restricted place. The main purpose of this alarm notification is to prevent the user from entering the restricted area by giving information to the nurse. For instance, elderly people might leave the building during midnight or in an unsuitable environmental condition. The machine learning system was implemented to prevent such a condition.

6.4 Auto-discovery

Auto discovery is the function that could allow one to recover the position of the user automatically when the user returns in connection to the system. The user Mobicat is not always connected to the system because the user may have gone outside and the connection between the user Mobicat and the access point may have been lost due to some physical and environmental problems. In such a case, the position of the user is not updated, thus the information is given incorrectly. In order to prevent such a condition, the auto discovery function was implemented. So, when the devices communication disconnects and the connection is lost for a longer time, the system will have to reallocate the position to the user when the Mobicat has reconnected to the system.

7 Desktop Application

The desktop application is a standalone application that runs in the local desktop computers or laptops supporting different operating systems. A desktop application is used especially for controlling the system software, visualizing and monitoring the system devices, debugging and also for testing purposes.

7.1 Requirement Specifications

The desktop application has to provide the accurate position of the users' Mobiletag on the screen and the movement of the users inside the building. The information of the users related to the room number, diseases and the condition of the user has to be easily available and for modification whenever necessary by the system administrator using the desktop application. The alarm notification should be visible on the screen with some alarming sound. The nurse has to acknowledge the alarm that provides the voice message to the users. The information of the alarmed user regarding the current position will have to be available when the alarm notification appears on the screen. The important information related to the user has to be stored in the history database. Similarly, the system administrator can also be able to give the alarm signal to users during an emergency situation such as a fire.

7.2 Programming Language and Development Tools

Java is an open-source, purely object-oriented, platform-independent and high-level programming language widely used for developing fast, secure and reliable applications. The Java platform is used to run and develop Java applications and it also provides a wide variety of tools that help developers work efficiently with the Java programming language. The Java Graphics User Interface (GUI) was used for the development of the Desktop Application. The Java application used Java platform, plug-ins and Java development tools such as Java Development Kit Tools, also called JDK, and Integrated Development Environment (IDE) for the development of the desktop application.

7.3 Memory Management and Optimization

Memory management is one of the very important issues in the software development process. The size of the memory is fixed for the system and during allocation of new objects, memory management finds free space and assigns the space to the new object. The more objects a Java application allocates, the more resources will be used for memory management. If the memory size is full, then the Java application will stop running. In order to prevent such a condition, Java uses a garbage collector that automatically removes the unused and old objects and provides free space for new objects. Listing 1 is an example of an exception that occurs when the memory size is full.

```
Exception in thread "Win32SerialPort Notification thread" java.lang.OutOfMemoryError: unable to create new native thread
  at java.lang.Thread.start0(Native Method)
  at java.lang.Thread.start(Unknown Source)
  at com.electria.Mobiletag_Everything.Mobile_Tag_FirstFloor.<init>(Mobile_Tag_FirstFloor.java:223)
  at com.electria.Beacon_Everything.Coridoor_BEACON_INFO.update(Coridoor_BEACON_INFO.java:966)
  at com.electria.Hea_Project.SerialPortReading.serialEvent(SerialPortReading.java:156)
  at com.sun.comm.Win32SerialPort.sendDataAvailEvent(Win32SerialPort.java:649)
  at com.sun.comm.NotificationThread.run(Win32SerialPort.java:878)
```

Listing 1: Java threads exception code

The above exception occurred during the software development of the project work. The developer always has to be careful when dealing with the Java thread and looping especially when it creates new objects. Because the creation of a new object always consumes space and if space has run out, then it will give the error that will stop the program from running.

7.4 Java Serial Communication (RS232)

Serial communication is the communication process of data transmission that sends data of one bit at a time through a communication channel or computer bus in sequence order. The Serial is a communication protocol commonly used by different devices for connection with the computer. The Serial is typically used for transmitting ASCII data and the communication is performed and accomplished using three transmission lines: Transmit, Receive and Ground. The serial communication is full-duplex and asynchronous, that is, it can send and receive data at the same time. In order to do that, the port can use one line for transmitting data and another line for receiving data while the third line is available and can be used for handshaking. The third line is not always required. [16]

There are some important serial characteristics that have to be matched for two ports to communicate. They are baud rate, data bits, stop bits and parity. Baud rate is a speed measurement for communication that indicates the number of bit transfer per second. For example, 400 baud is 400 bits per second. Similarly, a data bit is the actual measurement of the data bits during a transmission process. The standard values of the data packet are 5, 7, and 8 bits. Setting of data bits has to be done based on transferring information and the selected protocol. For instance, the values of standard ASCII have ranges from 0 to 127 which are 7 bits whereas the extended ASCII used 0 to 255 which are 8 bits. Stop bits indicate the end of communication for a single packet. Stop bit typical values are 1, 1.5 and 2 bits. Parity is used for error checking in serial communication and it helps to detect data corruption that might occur during transmission. There are four types of parity: even, odd, marked and spaced. During the serial communication, parity can be chosen either even, odd, marked, spaced or none at all. [16]

There are many different types of serial device and connectors available that can be used for serial communication purpose. The figure 10 shows the structure and functional diagram of DB-9 connector that can be used for serial communication.

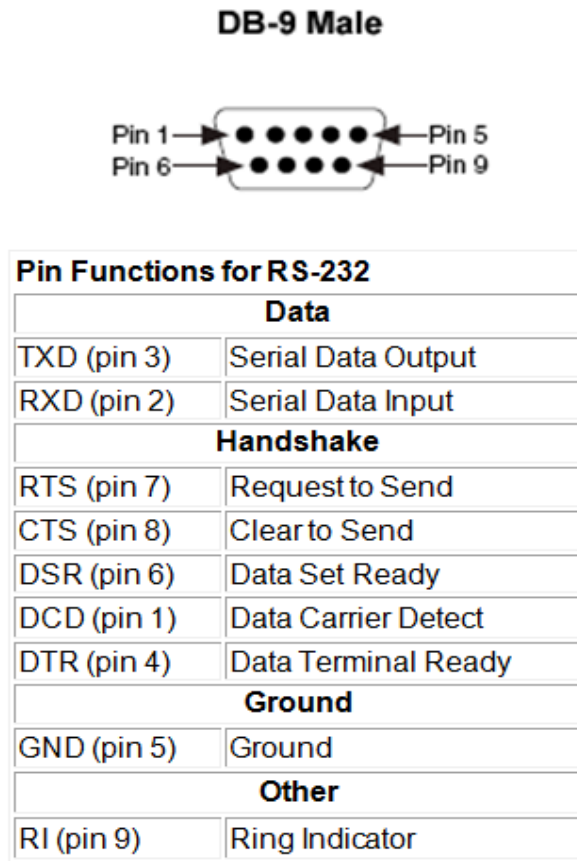


Figure 10: Definition of DB-9 connectors and RS-232 pin functions [16]

Figure 10 shows the pin out of a DB-9 connector used for RS-232 communication. RS-232 is the serial connection used to connect external devices such as mouse, modem, or printer including industrial instrumentation. RS-232 is limited to point-to-point connections between the devices and PC serial ports that can be used for serial communications of up to the distance of 50 feet. [16]

7.5 Flow Diagram of Desktop Application

Flow diagram in figure 11 shows the process and algorithm of software development. It also shows the different steps and the relation between the components. Figure 11 shows the flow diagram of the desktop application designed for the project.

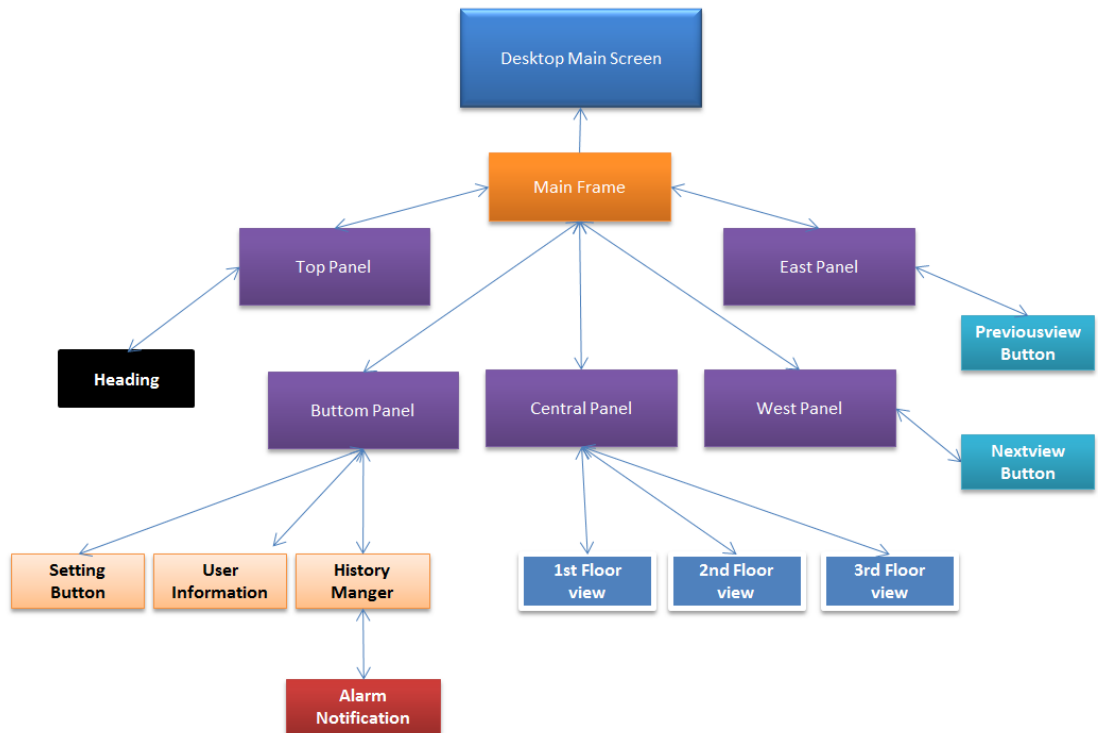


Figure 11: Flow diagram of the desktop application

A flow diagram can be used for analyzing, designing, and documenting and it also helps to understand the navigation of the process. It gives a clear understanding of the software components and their relation to teams who are not part of the programming or core developer. It also helps a newcomer to follow the software modules and structures.

7.6 Graphic User Interface (GUI) Design

User interface Design is the design of a software application with the focus on the user's experience and interaction. The main goal of the user interface design is to make the user's interaction as simple and efficient as possible to accomplish the user's goals. Java Graphics User Interface (GUI) was used here for the development of the desktop application. A desktop application is a standalone application that has its own platform, development tools and plug-ins to run the application. Figure 12 shows the graphic user interface design of the main view of desktop application developed using the Java programming language.

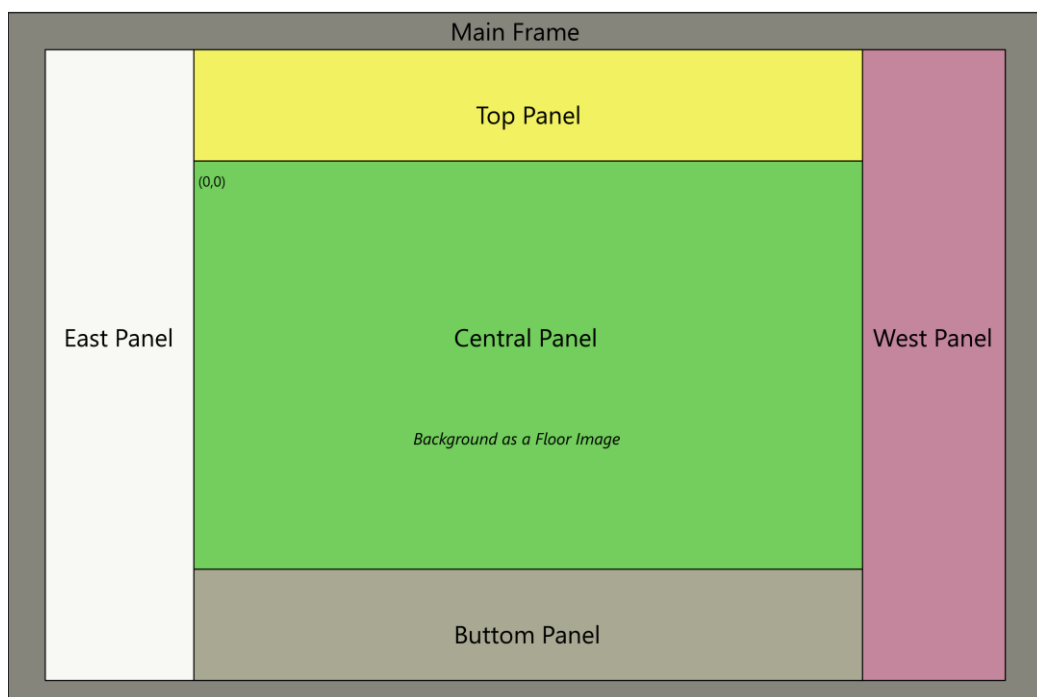


Figure 12: Graphic User Interface Design

The user interface consists of the main frame, panels and different bottom components. Java frame is also called desktop whereas panels are small frames that have to be included in main frame. In this application, there are five panels the central panel contains the image of the building floor as a background. The position of the user is shown in the central panel based on x and y-coordinate. The coordinate value of x and y is (0, 0) in the topmost left side of the central panel as shown in figure 12.

8 Results and Discussion

8.1 Implementation

Implementation is the most important part of the project development work. Implementation is done according to the product specifications and requirements. Flexibility and scalability have to be performed during implementation to modify or to do some changes in the future for further development.

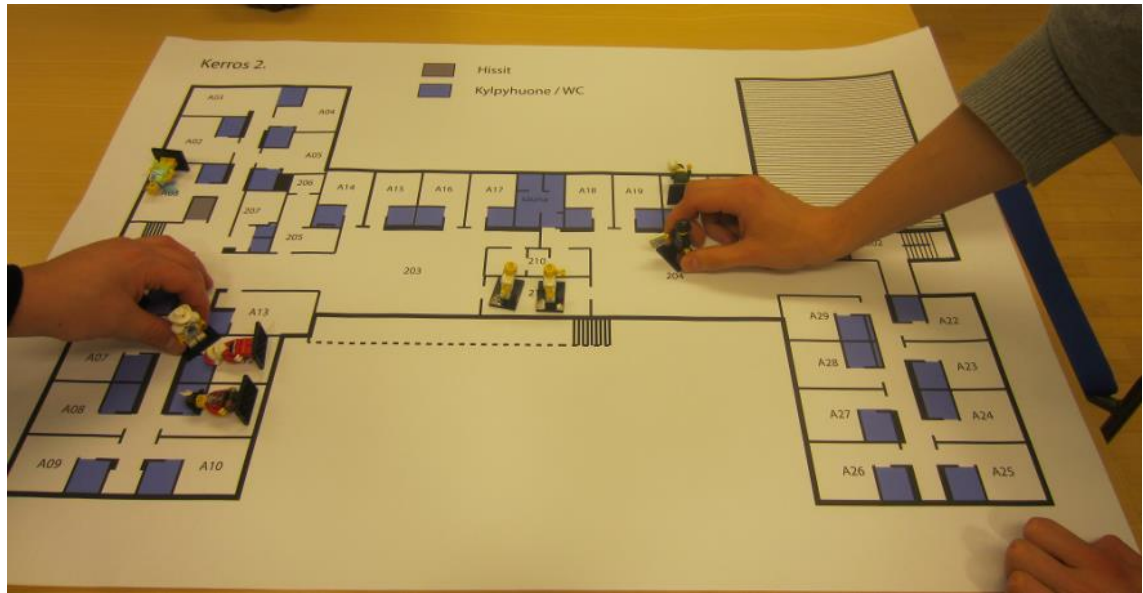


Figure 13: The game showing the nurse and the elder people activities

Figure 13 above shows the game which could correlate to infer the daily activities related to elderly people and the nursing staff. This game provides the idea of how nurses react against the elderly people's action. For example, what the nurses do when the elderly people try to get out of the bed or go to bathroom. The game was played between professional nurses and staff who were working in an elderly care home and with the developing teams. The results from the game show that the nurses need immediate action over each activity of the elderly people. Some of the elderly people need immediate help when they wake up. Similarly, handicapped patients also need immediate support from nurses when they wake up and try to move. The outcome and the feedback from the staff were really helpful for the development of the project work. It ensured that the final product developed in the project was really helpful and met the demands of the customer.

8.2 Algorithms

System algorithm represents the step-by-step finite list of well-defined instructions and procedures for the calculation of the function designed to perform by the system. The Algorithm for the desktop application software was programmed using the Java programming language. When the system is installed on the computer, it will expect the serial connection with Access-point. The system then scans the device, its connected ports and the communication between the software and the device will start automatically when the ports are found and connected. Listing 2 shows an example of the Java source code method that searches for the available ports of the Linux computer. The naming of the port in Linux starts with ttyS0 which is equivalent to the COM0 for Windows OS.

```

/*
 * search for all the serial ports and filters the default ports
 * pre: none
 * post: adds only the used ports to a combo box on the GUI and
 * removes default ports.
 */
public void searchForPorts() {
    ports = CommPortIdentifier.getPortIdentifiers();
    while (ports.hasMoreElements()) {
        CommPortIdentifier curPort = (CommPortIdentifier)
        ports.nextElement();
        //get only serial ports
        if (curPort.getPortType() == CommPortIdentifier.PORT_SERIAL)
        {
            if (curPort.getName().equals("/dev/ttyS1") ||
                curPort.getName().equals("/dev/ttyS0")) {
                System.out.println("Default serial port");
            } else {
                PortUIFrame.JSerialPort.addItem(curPort.getName());
                portMap.put(curPort.getName(), curPort);
            }
        }
    }
}
} // End of searchForPorts

```

Listing 2: Source code used for searching for Linux Serial ports

The radio signal transmission and receiving start after the successful connection of the device with the computer. It receives the signal transmitted by the Beacon. Beacons were placed inside the building where the position of a person has to be tracked and visualized. The position of the person was calculated based on the received RSSI signal strength value. Theoretically, the RSSI signal strength is directly proportional to the

distance between the transmitter and the receiver which basically means the person is in a particular position where the Beacon is placed.

Moreover, a cell-based algorithm was used to represent the appearance of the person inside the building. It provides the cell-based position referring that the person can be inside the room, in the bathroom or in the corridor. It cannot provide the exact position or the pinpoint of the people's position because of some challenges and issues related to the signal strength. The radio signal is affected by many physical factors: angle of orientation between the transmitter and the receiver, weather condition, blockage medium, or transmission power.

8.3 Challenges and Solutions

There were many challenges occurring during the project development stage. The first and foremost challenge faced during this project development was to transmit and receive the signal data between Beacons and Mobicat. The transmitted and received data had to be visualized on the computer for further processing. RealTerm: Serial/TCP Terminal Application software was used to visualize the types of received and transmitted data and also used for testing and debugging difficult communication problems signal data. Using that software, the data types, its structure and the phenomena were studied. The signal data were transmitted in the form of Packet. The Packet contains the collection of signal data which basically contains the transmitter device IDs and the corresponding RSSI value in the form of byte arrays. The devices can transmit the signal data at a minimum interval of 300-400 milliseconds which means it can transmit the packet data approximately 3 times a second. It was realized that the devices can receive the signal every second.

Once the data was fully visualized and investigated, another important challenge was faced when finding the appropriate algorithms for the calculation and implementation of those data in the indoor navigation system. Different algorithms and principles such as Triangulation Principle, Trilateration Principle, Finger Print Technology and Cell Based System were considered and an appropriate technology was selected for the project. The Fingerprint technology seems to be more effective in comparison to the trilateration and triangulation principle since it does not depend on the direction of the angle and distance. Theoretically, the distance is directly proportional to the RSSI signal strength measured between the transmitter and receiver but in real life it is not 100 percent correct.

The RSSI value gets affected by many factors. Some of major factors that affect the RSSI values are Angle of Orientation, Types of Obstacle, Reflection, Refraction and Transmission output power. Apart from these factors, signal properties like Time of Arrival (TOA) and Angle-Of-Arrival (AOA) also affects the RSSI value. The drawback of the Fingerprint technology is that it is a time-consuming process. Most of the time has to be spent on taking the measurement and storing the signal strength data value to the database. The signal strength measured at a certain point is always changing.

If some changes are made inside the building, for example some furniture or the wall are moved or replaced, then, the previously stored signal data in the database will have to be updated and changed in order to make it work. Because of this reason, the Fingerprint technology was not considered an appropriate solution for the indoor navigation system in the project.

The cell-based system was selected for this project work. The cell-based system is the principle that represents the position of the user on the cell-based instead of giving the exact pin-point of the position. Using this principle, the system says that users is located in a certain place, inside the particular room, corridor or in the bathroom which is acceptable in the project. After facing all the challenges mentioned above, the system now can measure and calculate the position of the single user in the Desktop User Interface Screen. The system can track the user position and the movement on the screen. However there were still some challenges for further development and for making the system robust.

There was the challenge of the appearance of multi-users on the screen. A Wireless link between Access-point and Mobile tag was disconnected for a longer time and the possibility of missing the Gate Beacons was higher. These problems were solved by selecting the appropriate antenna, transmitting the output power and setting the angle of orientation of the transmitter placed on a fixed position inside the buildings. The signal transmitted by the antenna can be controlled by using some dense objects such as metal or aluminium foil placed above the unwanted direction to prevent the signal transmission to those directions and focus on the particular direction.

8.4 Achievements

The final product developed in the project succeeded in achieving the main goal the project. The system is fully capable of tracking multiple users and can show their positions on the desktop screen at the same time. Similarly, the alarm and feedback with voice functionality were implemented and are in use. Important data collected by the system, which is related to the users, can be stored for future reference. The system and the information provided by the system can be helpful for nurses to study the behaviors of the users, especially elderly people, their regular activities and interests. The user interface of the desktop application was shown below in figure 14:

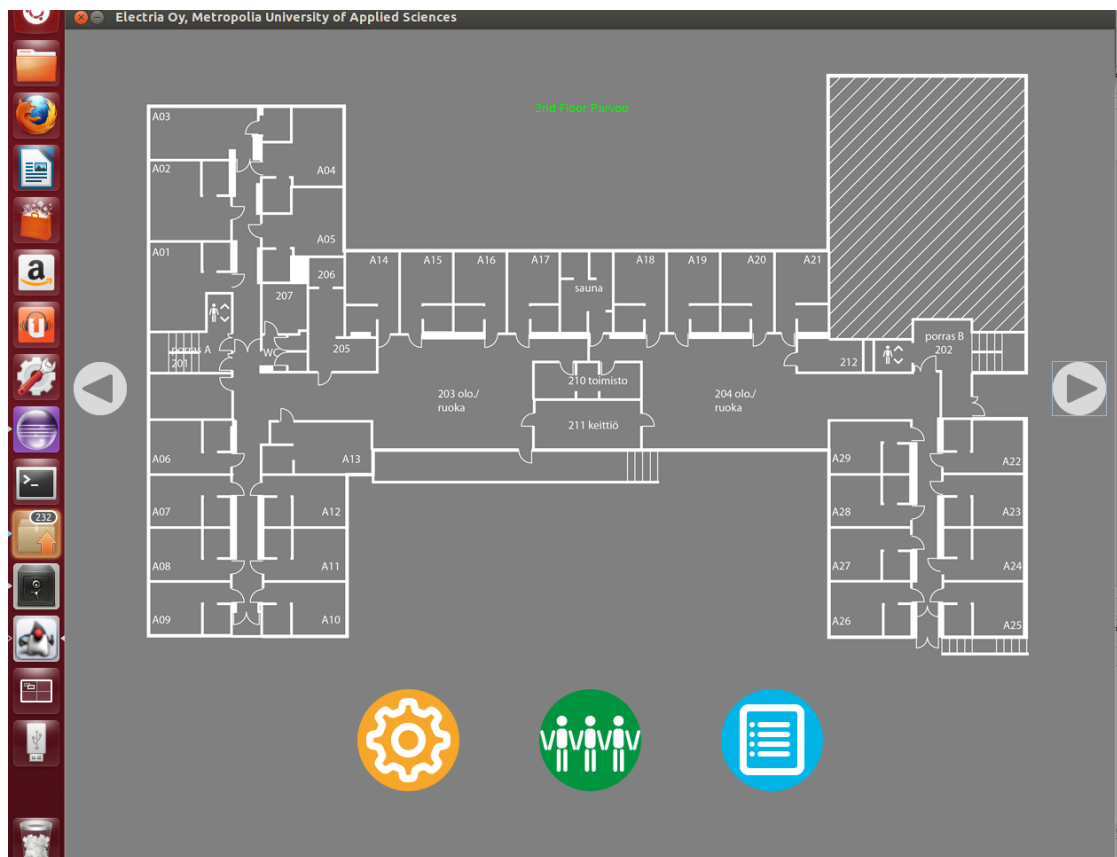


Figure 14: Main view of the desktop application

The system was designed to make it more intelligent and robust, so that the customer does not need to go for manual setting in order to run the application. The system runs the application automatically, connects the port and detects the user on the screen.

8.5 Project Development

The project was started at the beginning of May 2012. The estimated time for the project development was based on the project success and the possibility of improvements and achievements. The rough schedule of the project development was prepared (Figure 14). The project length was extended later for improvement and development. The decisions were made after the first demonstration of the project success.

Weeks 2-5	Weeks 2-4	Weeks 3-4	Weeks 1-3
Research and Development	Hardware And Software Selection	Desktop Application	Testing and Conclusion

Figure 15: Project development schedule time in weeks

During the first four months of project development, the first demo of the indoor navigation system on Desktop Application was prepared. The demo was shown to the customers and the continuation of the project was based on the customers' feedbacks and decisions. The project demo was shown to the supervisor, management team, administrator and customers. The decision for further development of the project was based on the customers' satisfaction and selection.

Initially, the project was developed on the basis of the Waterfall model. The customer was not involved and the sequential development was carried out through different phases such as requirements analysis, design, implementation, testing and validation, integration, and maintenance. [6]

After four months of project development, the Waterfall model project development method was changed to the Agile software development method because of the involvement of customers. [7] The requirements and solutions were changed based on the customer demands. The customers' feedback and suggestions were considered and used to proceed for further development of the project.

8.6 Testing

Testing is one of the important parts of the project development phase, as it verifies whether the system performs according to expectations. Testing also helps the designer and the developer to make an assessment on whether the designed piece of hardware or software works as expected. Once the implementation part was completed, the system developed in the project was installed and tested in the lab building. Beacons were powered and placed at a distance of 6 to 10 meters inside the building corridors, rooms and gates. Access-point was connected to the lab computer and the mobile tag user was instructed to walk inside the building where the Beacons were placed. The result was promising and the system worked properly as expected.

During demonstration, all the management team members, the supervisor, visitors and customers along with developers were in the conference hall. The laptop computer running the application was connected with the Access-point. The projector was used to show the application to the audience. One of the developer team members was holding the mobile tag and walking inside the designed path of the building. The outcome was very satisfactory. Therefore, the project was decided to be continued for further developments. Furthermore, this project work was accepted for installation in customer elderly care homes. Customers welcome the project work and were pleased to have the system installation in their elderly care homes.

8.7 Final product

The final product of this project was successfully demonstrated and launched. The outcome of the project was very fruitful. The final product developed in the project meets the requirements of the customer and the product could also be used as an industrial product. Some of the products developed in the project include Beacon, Mobiletag, Access-point and Desktop Application.

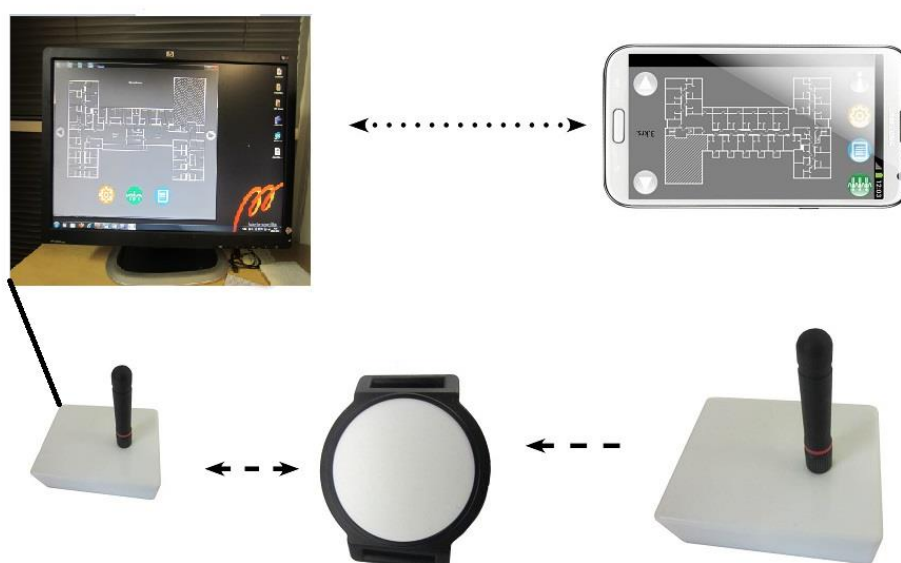


Figure 16: Final product developed in the project

The diagram illustrated in figure 15 shows the final product of the project. Beacon and Access-point were designed and built as the same device. Because the size for these devices need not be small, an extra antenna is required for these devices to communicate with the devices at longer distances. Mobiletag was designed in the form of a wrist watch to facilitate portability. The desktop application is fully functional and fulfills the requirements of the project. The development of an Android application was also part of the initial requirements of the project which were changed later and the Android application parts were not discussed in this report.

8.8 Future Development

The DASH7 technology encompasses many scopes though it was originally created for military use only. Its advanced features and multipurpose functionality can be used in different fields of the wireless sensor networking technology. It is a verified technology and can replace the old technology. It makes the system more reliable, stable, economically viable and interoperable in the field of RFID wireless communication. The system developed in the project can be used in different fields such as in hospitals, or nursing care homes to look after patients. Similarly, it can also be used in museums and stores to locate and guide visitors. [1] Besides the navigation system, DASH7 can be used in different fields of automation technology, vehicle parking management systems, housing automation smart energy systems, tire pressure monitoring, in-transit temperature monitoring of perishable goods and also on the smartphone that complements near field communication (NFC). The Near field communication system is a wireless sensor networking standard used in smartphone for the application of a mobile payment system. [6] Figure 17 shows the increasing trends of wireless sensing devices in projected market growth.

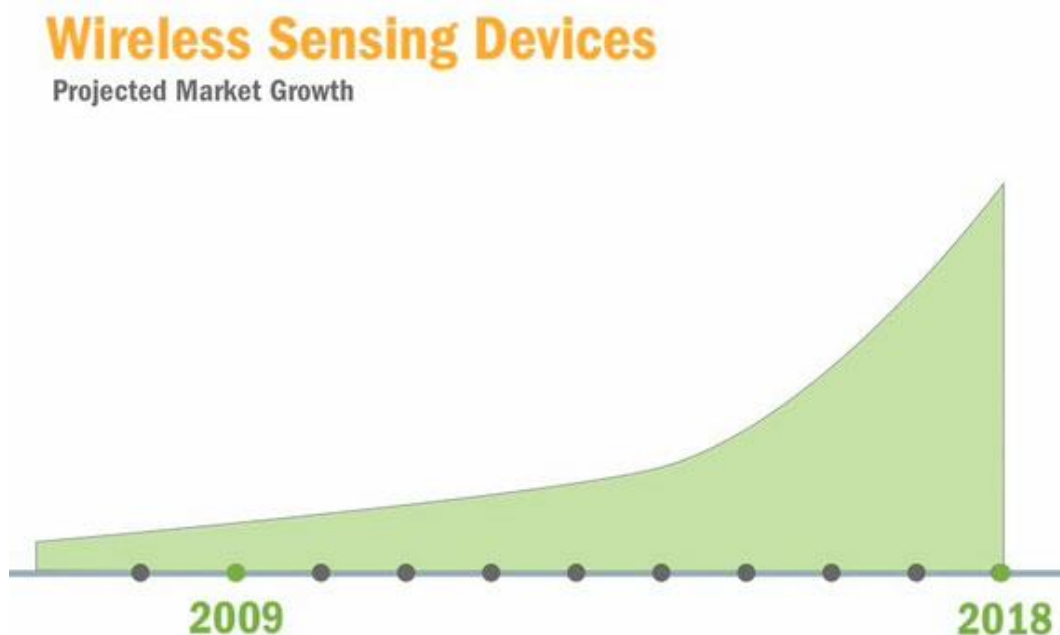


Figure 17: Wireless Sensing Devices [6]

Recently, DASH7 Alliance has developed the ReadWriteWeb that is going to be an integral part of the Internet and helps to run social networking applications that use sensor data. [6]

9 Conclusion

The purpose of this project was to develop an indoor navigation system and thus to provide emergency help to senior citizens living in the pilots and care homes. The project was related to the software and embedded field of engineering. The project also involved different technologies such as the DASH7 wireless radio technologies, micro-controllers, printed circuit board (PCB) devices and Radio Signal Strength Identification (RSSI) technologies. Gaining knowledge of the above technologies were prerequisites for the success of the project.

The project was successfully completed. The main objective of the project was achieved; the indoor navigation system was successfully designed and implemented for a desktop computer using a Java application. The indoor navigation system includes the functionality of the emergency help and voice feedback. The system is capable of tracking and locating the information of multi-users on the screen. The hardware devices were designed and implemented with a power saving, smaller in size, attractive, cheaper and robust operation.

However, the system developed in this project has some limitations. The system is capable of tracking and locating the people inside the building but the system cannot detect the people getting in and out of bed and falling down. Furthermore, the system cannot give the pinpoint position of the people. Instead it can give only the cell-based position. These limitations can be avoided by using different kinds of sensor devices such as motion sensor, pressure sensor and other technologies.

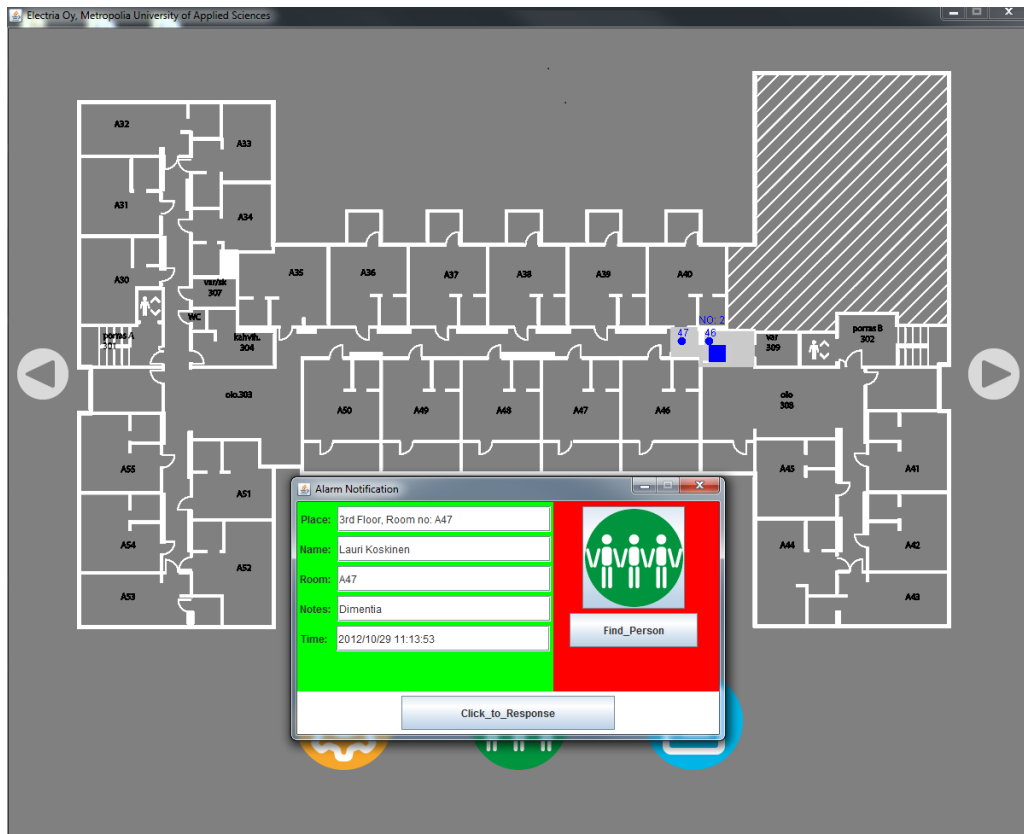
References

1. DASH7.org. DASH7 Alliance [online]. Morgan Hill, CA 95037: DASH7 Alliance, Inc.; 2012.
URL: <http://www.dash7.org/>. Accessed 09.02.2013.
2. DASH7.org. DASH7 Alliance [online]. Morgan Hill, CA 95037: DASH7 Alliance, Inc.; 2012.
URL: <http://www.dash7.org/feature-comparison>. Accessed 09.02.2013.
3. NorairJP. Introduction to DASH7 Technologies.
<https://dash7.memberclicks.net/assets/PDF/dash7%20wp%20ed1.pdf>; 2009.
4. Chandra AM. Triangulation & Trilateration. Higher Surveying. Daryaganj, New Delhi: New Age International(P) Ltd; 2005.
5. Dimitris E. Manolakis. Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration [online]. Athens, Greece:Hellenic Air Force Academy; 2002.
URL:
<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&number=543845&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F7%2F11782%2F00543845.pdf%3Farnumber%3D543845>. Accessed 09.12.2013.
6. Finnish Geodetic Institute. Indoor navigation [online]. Masala, Finland: Finnish Geodetic Institute; 2013. URL: <http://www.fgi.fi/fgi/themes/indoor-navigation>. Accessed 09.12.2013.
7. MacManus Richard. DASH7:Bringing Sensor Networking to Smartphones [online]. Say Media Inc; 2010.
URL:
http://readwrite.com/2010/04/11/dash7_bringing_sensor_networking_to_smartphones. Accessed 09.12.2013.
8. Kevin Curran, Eoghan Furey, Tom Lunney, Jose Santos, Derek Woods and Aiden Mc Caughey. An Evaluation of Indoor Location Determination Technologies. Journal of Location Based Services Vol. 5, No. 2, pp: 61-78, June 2011, ISSN: 1748-9725, DOI:10.1080/17489725.2011.562927: Taylor & Francis; 2011.
9. Tse David, Wiswanath Pramod. Fundamental of Wireless Communication. Cambridge: Cambridge University Press; 2005.
10. Oracle. The Java Tutorials [online]. Redwood Shores, CA 94065: Oracle Corporation; January 2014.
URL: <http://docs.oracle.com/javase/tutorial/uiswing/components/frame.html>. Accessed 05.02.2014.
11. NearFieldCommunication.org. Near Field Communication [online]. NearFieldCommunication.org: NFC Forum; 2014.
URL: <http://www.nearfieldcommunication.org>. Accessed 24.02.2014.

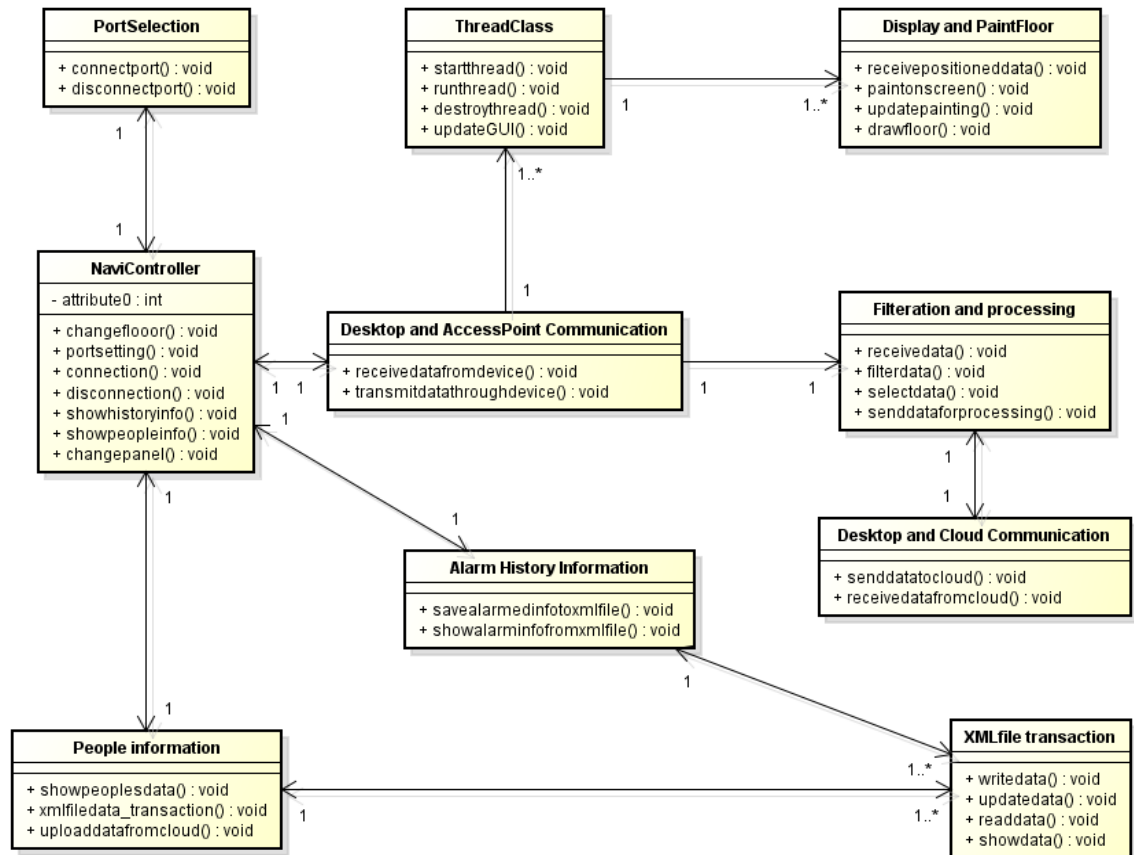
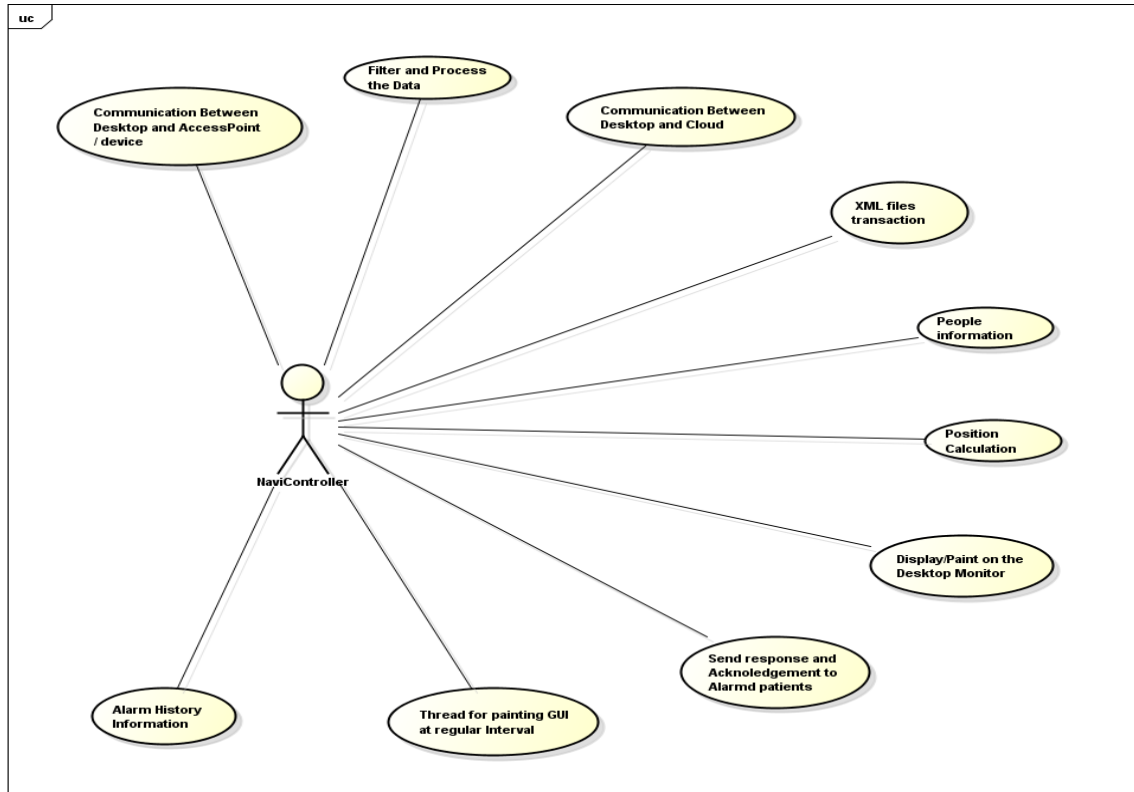
12. Malik Ajay. RTLS For Dummies. Hoboken, NJ: Wiley Publishing , Inc.; 2009.
13. Nihonjin Sivakumar. Received signal strength indication and IEEE802.11k2008 [online]. January 4, 2011.
URL: <http://sivanihonjin.blogspot.fi/2011/01/received-signal-strength-indication-and.html>. Accessed on 06.02.2014.
14. Bar code Graphics. How does a RFID system work? [online]. Chicago, IL 60611: bar Code Graphics, Inc.; 2014.
URL: <http://www.epc-rfid.info/rfid>. Accessed on 27.02.2014.
15. Oracle. The Java Tutorials [online]. Redwood Shores, CA 94065: Oracle Corporation; January 2014.
<http://docs.oracle.com/javase/7/docs/webnotes/install/windows/windows-system-requirements.html#processor-32>. Accessed on 27.02.2014.
16. National Instruments Corporation Finland Oy. Serial Communication General Concepts [online]. Espoo, Finland: National Instruments Corporation; 2013.
URL: <http://digital.ni.com/public.nsf/allkb/2AD81B9060162E708625678C006DFC62>. Accessed on 25.02.2014.
17. Wassenberg Wads. Java Comm Serial API How-To for Linux [online] 2014.
URL: http://www.agaveblue.org/howtos/Comm_How-To.shtml . Accessed on 29.02.2014.
18. Circuit Negma. How to install The Java Communications API in a Windows Environment [online]. 2007.
URL: <http://circuitnegma.wordpress.com/2007/02/07/how-to-install-the-java-communications-api-in-a-windows-environment/>. Accessed on 03.03.2014
19. Tran Eushiuan. Requirements & Specifications [online]. Carnegie Mellon University; 1999.
URL: https://www.ece.cmu.edu/~koopman/des_s99/requirements_specs/. Accessed on 12.03.2014.
20. Cook, B., Buckberry, G., Scowcroft, I., Mitchell, J. and Allen, T. Indoor Location Using Trilateration Characteristics [online]. London, UK: School of Science and Technology, Nottingham Trent University; 2005.
URL: <http://www.ee.ucl.ac.uk/lcs/previous/LCS2005/12.pdf>
Accessed on 12.03.2014.

Appendices

Appendix 1: Graphics User Interface for User Position and Alarm View



Appendix 2: Use Case Diagram and Class Diagram



Appendix 3: Java Source Code for Main GUI Design

```

/* Main user interface class */
import java.awt.*;
import java.awt.image.BufferedImage;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

import com.metropolia.electria.marimills.PortUI;
import com.metropolia.electria.Floorview.FirstfloorView;
import com.metropolia.electria.Floorview.SecondfloorView;
import com.metropolia.electria.Floorview.ThirdfloorView;
import com.metropolia.electria.Floorview.GraphicsThread;
import com.metropolia.electria.MainView.PanelSetting;
import com.metropolia.electria.MainView.PanelHistory;
import com.metropolia.electria.MainView.PanelPeople;

public class NavigationUI extends JFrame {
    /* ImageIcons for the Buttons background */
    final public ImageIcon people = new ImageIcon(getClass().getResource("/peoples.png"));
    final public ImageIcon history = new ImageIcon(getClass().getResource("/history.png"));
    final public ImageIcon setting = new ImageIcon(getClass().getResource("/setting.png"));
    final public ImageIcon next = new ImageIcon(getClass().getResource("/NEXT.png"));
    final public ImageIcon previous = new ImageIcon(getClass().getResource("/PREVIOUS.png"));

    /* JPanels in the main frame */
    public JPanel PanelTop = new JPanel();
    public JPanel PanelNext = new JPanel();
    public JPanel PanelPrevious = new JPanel();
    public JPanel PanelFloor = new JPanel();
    public JPanel PanelButton = new JPanel();

    /* JButton to the main frame */
    protected JButton peopleButton = new JButton(people);
    protected JButton portsettingButton = new JButton(setting);
    protected JButton historyButton = new JButton(history);
    protected JButton nextButton = new JButton(next);
    protected JButton previousButton = new JButton(previous);

    /* Buttons for Exiting the panel to mainframe panel*/
    protected JButton EXITPEOPLE = new JButton(people);
    protected JButton EXITHISTORY = new JButton(history);
    protected JButton EXITPORT = new JButton(setting);

    protected int Floormapnumber; // for floor changing purpose

    /* Importing other Panels that changes on button click */
    public PanelSetting panelsetting = new PanelSetting();
    public PanelHistory panelhistory = new PanelHistory();
    public PanelPeople panelpeople = new PanelPeople();

```

```

/* Main Frame Constructor */
public NavigationUI() {
super("Electria Oy, Metropolia University of Applied Sciences");

/* Default floor map presented on the GUI*/
add(SecondfloorView.getInstance());
SecondfloorView.getInstance();
Floormapnumber = 2;

/* Add the Panels to the Main Frame*/
add(PanelTop, BorderLayout.NORTH);
add(PanelPrevious, BorderLayout.WEST);
add(PanelNext, BorderLayout.EAST);
//add(PanelFloor, BorderLayout.CENTER);
add(PanelButton, BorderLayout.SOUTH);

/* Add the buttons to the panels*/
PanelNext.add(nextButton);
PanelPrevious.add(previousButton);
PanelButton.add(portsettingButton);
PanelButton.add(peopleButton);
PanelButton.add(historyButton);

/* setting the size of the Panels and background color */
PanelTop.setPreferredSize(new Dimension(1184, 50));
PanelTop.setBackground(Color.gray);
PanelPrevious.setPreferredSize(new Dimension(80, 80));
PanelPrevious.setBackground(Color.gray);
PanelNext.setPreferredSize(new Dimension(80, 80));
PanelNext.setBackground(Color.gray);
PanelFloor.setPreferredSize(new Dimension(1024, 656));
PanelFloor.setBackground(Color.gray);
PanelButton.setPreferredSize(new Dimension(1184, 230));
PanelButton.setBackground(Color.gray);

/* Decorating the Buttons with size, color and background*/
previousButton.setBackground(Color.gray);
previousButton.setPreferredSize(new Dimension(80, 700));
previousButton.setBorder(null);
previousButton.setBorderPainted(false);
previousButton.setContentAreaFilled(false);

nextButton.setBackground(Color.gray);
nextButton.setPreferredSize(new Dimension(80, 700));
nextButton.setBorder(null);
nextButton.setBorderPainted(false);
nextButton.setContentAreaFilled(false);

portsettingButton.setPreferredSize(new Dimension(200, 180));
portsettingButton.setBorder(null);
portsettingButton.setBorderPainted(false);
portsettingButton.setContentAreaFilled(false);

peopleButton.setPreferredSize(new Dimension(200, 180));
peopleButton.setBorder(null);
peopleButton.setBorderPainted(false);
peopleButton.setContentAreaFilled(false);

historyButton.setPreferredSize(new Dimension(200, 180));

```



```
historyButton.setBorder(null);
historyButton.setBorderPainted(false);
historyButton.setContentAreaFilled(false);

EXITPEOPLE.setPreferredSize(new Dimension(200, 230));
EXITPEOPLE.setBorder(null);
EXITPEOPLE.setBorderPainted(false);
EXITPEOPLE.setContentAreaFilled(false);

EXITHISTORY.setPreferredSize(new Dimension(200, 230));
EXITHISTORY.setBorder(null);
EXITHISTORY.setBorderPainted(false);
EXITHISTORY.setContentAreaFilled(false);

EXITPORT.setPreferredSize(new Dimension(200, 230));
EXITPORT.setBorder(null);
EXITPORT.setBorderPainted(false);
EXITPORT.setContentAreaFilled(false);
```

Appendix 4: Serial Data Reading, Filtering and Selecting Based on RSSI Value

```

/**
 * Reading the data from serial port with event based
 * pre: input stream data in byte by byte form
 * post: Data acquisition and processing
 * for eg; data to server and graphics
 */
public void serialEvent(SerialPortEvent event) {
    switch (event.getEventType()) {
        case SerialPortEvent.OE:
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[40];
            final int d = 1;
            byte[] Payload = { -35,0,2,2,4,1,127, (byte) MobiletagID, 0,0,-65,125
            };// UART payload
            byte[] RayLoad = { -35, 0, 2, 2, 4, 1, 16, 16, 0, 0, -65, 125 };

            try {
                while (inputstream.available() > 0) {
                    int numBytes = inputstream.read(readBuffer);// received Byte
                    int[][] ID_RSSI_INT = new int[8][2]; // length

                    /**
                     * Removing noise and unwanted data, filtering based on Packet received
                     * note: check the packet structure and array elements
                     */
                    //if(readBuffer == null || readBuffer[6]!=-86 || readBuffer[8]<1
                    ||readBuffer[9]==0 || readBuffer[10]==0){
                    if(readBuffer == null || readBuffer[6]!=-86 ||readBuffer[9]==0 ||
                    readBuffer[10]==0){
                        break;
                    }else{
                        /* Original Data reading form the serial port in dbm byte by byte */
                        for (int i = 7; i < numBytes - 7; i++) {
                            //System.out.print(String.format("%x ", readBuffer[i]));
                            Por-
                            tUIFrame.ShowReadingdata.append(Integer.toString(readBuffer[i])+"");
                        }
                        System.out.println();
                        PortUIFrame.ShowReadingdata.append("\n");

                        /* PUTTING THE DATAS INTO DIMENTIONAL FORM FOR EASY SORTING*/
                        for (int i = 0, k = 9; i <= 4; i++) {
                            for (int j = 0; j <= 1; j++) {
                                ID_RSSI_INT[i][j] = (int) readBuffer[k]; //Look multiplication with
                                -1
                                //System.out.print(" " + ID_RSSI_INT[i][j]);
                                k++;
                            }
                        }
                        //System.out.println(); //ifyouwantoprintintwodimensionalform

```

```

} //System.out.println(); //ifyouwantoprintintwodimensionalform
MobiletagID = readBuffer[7]; //System.out.println(MobiletagID);
AlarmNotification = readBuffer[8];

/**
 * SORTING ALGORITHM BASED ON RSSI
 * Sorting tricks: changing position of returns or -1 multiplication to
 * get rid of nevasive datas value.
 * Note: Check the packet structure and array elements
 */
Arrays.sort(ID_RSSI_INT, new Comparator<int[]>() {
@Override
public int compare(final int[] entry1, final int[] entry2) {

if (entry1[d] < entry2[d])
return 0;
else
return 1;
}
});

BeaconId =ID_RSSI_INT[0][0]; //Select the nearest Beacon based on RSSI
value
RSSIValue = ID_RSSI_INT[0][1]; //System.out.println("BeaconId::"+ Bea-
conId+ " RSSIValue::"+ RSSIValue);
System.out.println("MobiletagID:BeaconId:AlarmNotification::
"+""+MobiletagID+",""+ BeaconId+",""+ AlarmNotification);

/**
 * Resetting the Mobiletag
 */
if(AlarmNotification == 126){
outputstream.write(PayLoad);
System.out.println("Alarm Signal has received");
}else if(AlarmNotification ==15){
outputstream.write(RayLoad);
System.out.println("Alarm Red signal has reset");
}
/**
 * Batchupdate Data sending to the server in event based. Received
 * datas are filtered, processed and sent to the Server
 * in every time receiving data in the form of JSON Array that con-
 * tains MobiletagID, BeaconId and RssiValue or AlarmID.
 */
//updateJSON_Server(MobiletagID, BeaconId, AlarmNotification);

/**
 * Mapping multiple Mobiletag for corridor Beacon selection
 * Create a hash map and Put elements to the map
 * Checks if Mobiletag is already registered, if registered then values
 * are just updated
 * Otherwise, new Mobiletag is created.
 */
if (multiple_mobile.containsKey(MobiletagID)) {
multiple_mobile.get(MobiletagID).update(MobiletagID, BeaconId, Alarm-
Notification, RSSIValue);
System.out.println("ReadingDatafor update: " + "" + MobiletagID+ "::::"
+ multiple_mobile.get(MobiletagID).getBeaconid());
} else {
multiple_mobile.put(MobiletagID, new Coridoor_BEACON_INFO(MobiletagID,
BeaconId, AlarmNotification, RSSIValue));
}

```

```

// System.out.println("ReadingDatafor adding: "+""+MobletagID+":::"+
multiple_mobile.get(MobiletagID).getBeaconid());
}

}
} catch (IOException e) {
System.out.println(e);
} catch (Exception e) {
e.printStackTrace();
}
break;
}
}
}

```

Appendix 5: Main Method Java Source Code

```

public static void main(String[] args) throws IOException {
NavigationUI Harisframe = new NavigationUI();
Harisframe.setSize(1184, 785);
Harisframe.pack();
//Harisframe.setLocationRelativeTo(null);
Harisframe.setVisible(true);
Harisframe.setResizable(false);
GraphicsThread gt = new GraphicsThread();
gt.start();

//new PortUI();
/**
 * System will automatically starts with the selected port and default
parameters,
 * In other case, setting has to be done manually and has to press the
connect button.
 * In windows OS, the programs works fine since each device has its
own port
 * In linux OS, port can be changed for same device, if multiple de-
vice connected
 */
PortUI autostart = new PortUI();
autostart.getSerialportConnection(null);
}

```

Appendix 6: Structure of Process Data

```

package com.metropolia.electria.marimills;
/*
 * Data structure for holding the information of data that will send to
the Server.
 * Stores the information of Mobiletagid, Beaconid and alarmid and also
contains update method
 */
public class senddata_structure {
private int mid;
private int bid;
private int aid;

public senddata_structure(int mid, int bid, int aid) {
super();
}
}

```

```
this.mid = mid;
this.bid = bid;
this.aid = aid;
}

public int getMid() {
return mid;
}
public void setMid(int mid) {
this.mid = mid;
}
public int getBid() {
return bid;
}
public void setBid(int bid) {
this.bid = bid;
}
public int getAid() {
return aid;
}
public void setAid(int aid) {
this.aid = aid;
}

public void updatadatatosent(int bid, int aid){
this.bid = bid;
this.aid = aid;
}
}
```

Appendix 7: Example Source Code View Panel

```

package com.metropolia.electria.Floorview;
import java.awt.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class FirstfloorView extends JPanel {
private static FirstfloorView instance = new FirstfloorView();
private static final int WIDTH = 1024;
private static final int HEIGHT = 656;
private BufferedImage img;
private Graphics buffer;
final public Image background = new Image-
Icon(getClass().getResource("/First_Floor_Parvoo.png")).getImage();

public FirstfloorView(String selectedonlypaint) { //Alarm response
also use this operation
super();
}

private FirstfloorView() {
setPreferredSize(new Dimension(WIDTH, HEIGHT));
img = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE_INT_RGB);
buffer = img.getGraphics();
buffer.fillRect(0, 0, WIDTH, HEIGHT);
}

public static FirstfloorView getInstance() {
return instance;
}

public void updatelstscreen() {
repaint();
}

public void paintComponent(Graphics g) {
super.paintComponent(g);
buffer.drawImage(background, 0, 0, null);
buffer.drawString("1st Floor Parvoo", 450, 43);
buffer.setColor(Color.WHITE);
g.drawImage(img, 0, 0, this);
}
}

```

Appendix 8: Graphics Thread for Updating the Floorviews

```

package com.metropolia.electria.Floorview;
import com.metropolia.electria.MainView.PanelSetting;
/**
 * Thread class for painting all the floor in certain time interval
 * Output: updates all the floors data with time interval and handles
 threads
 */
public class GraphicsThread extends Thread {
    private boolean running = true;

    public GraphicsThread(){
    }
    @Override
    public void run() {
        while (running) {
            try {
                Thread.sleep(200);
                FirstfloorView.getInstance().update1stscreen();
                SecondfloorView.getInstance().update2ndscreen();
                ThirdfloorView.getInstance().update3rdscreen();
                //System.out.println("Graphics updated
"+Thread.activeCount());
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    public void finish(){
        running = false;
    }
}

```

Appendix 9: Source Code for Mobicat Positioning Data Structure

```

package com.metropolia.electria.MainAlgorithm;
import java.awt.Image;
public class PSI_ForMobicatHolder {
private int X_Position;
private int Y_Position;
private int X1, Y1;
private Image Shadow_Image;

public PSI_ForMobicatHolder( int x_Position, int y_Position, Image
shadow_Image, int x1, int y1) {
super();
X_Position = x_Position;
Y_Position = y_Position;
Shadow_Image = shadow_Image;
this.X1 = x1;
this.Y1 = y1;
}
public int getX_Position() {
return X_Position;}
public void setX_Position(int x_Position) {
X_Position = x_Position;}
public int getY_Position() {
return Y_Position;}
public void setY_Position(int y_Position) {
Y_Position = y_Position;}
public Image getShadow_Image() {
return Shadow_Image;}
public void setShadow_Image(Image shadow_Image) {
Shadow_Image = shadow_Image;}
public int getX1() {
return X1;}
public void setX1(int x1) {
this.X1 = x1;}
public int getY1() {
return Y1;}
public void setY1(int y1) {
this.Y1 = y1;}
}

```


Appendix 10: Source Code for Updating the Mobiletag Position

```

/**
 * Initialise and updates the system. The input datas are taken directly from the serialport reading class
 * The system will initialised with 3 reading of same Beacon from wherever from the floors and assigne the
 * position of that Beacon which has been harcoded into the arraylist. The cordinates position are always
 * fixed. Only the Beacon id will changed in case of damage or sth else..
 * The corridors has assigned certain rules that it can moves.
 * @param Mobiletagid
 * @param bID_FIRST
 * @param AlarmInfo
 * @param RSSI
 */
public void update(int Mobiletagid, int bID_FIRST, int alarmInfo, int
rSSI) {
//Beaconimport.Beaconimporting();
//Coridoor_BEACON_INFO.BeaconVariable(Beaconimport.AllBeaconVariable);
MOBILE_TAG_ID = Mobiletagid;
Beaconid = bID_FIRST;
Alarm_Notification= alarmInfo;

if (OriginGot == false) {
/* Resetting array contents */
if (i > 2) {
i = 0;
j[0] = 0;
j[1] = 0;
j[2] = 0;
}
j[i] =bID_FIRST; System.out.println("Firstreading_BID: " + j[i]);
i++; System.out.println("COrdinates: " + " " + j[0] + " " + j[1] + " "
+ j[2]);
//System.out.println("All Cordinates ID: "+ Corridor3_ID +","+ Corri-
dor4_ID
+",""+Corridor5_ID+",""+Corridor6_ID+",""+Corridor7_ID+",""+Corridor8_ID);
/* Inatializing the system with first 3 reading of same corridor Bea-
cons */
if (j[0] == j[1] && j[1] == j[2] && (j[2] == Corridor3_ID || j[2] ==
Corridor4_ID || j[2] == Corridor5_ID
|| j[2] == Corridor6_ID || j[2] == Corridor7_ID
|| j[2] == Corridor8_ID)) {
Selected_CorridorID = j[0];
System.out.println("Selected COrdinates: "+ Selected_CorridorID);

/* Selects the matching Beacon from the arraylists and assign the po-
sition */
for (Object Corridor : Corridor_Beacon_Parameter) {
CORRIDOR_BID BC = (CORRIDOR_BID) Corridor;
System.out.println("BC.getBeacon_ID();: "+ BC.getBeacon_ID());

if (Selected_CorridorID == BC.getBeacon_ID()) {
OriginGot = true; //Initialised the system from corridor
MOBILE_TAG_ID = Mobiletagid;
Beaconid = bID_FIRST;
Alarm_Notification= alarmInfo;
Previous_BeaconID = Selected_CorridorID;
X = BC.getCorX();
Y = BC.getCorY();
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println("Your Sistem has initialised " + MOBILE_TAG_ID + "
:: " +Beaconid);
break;

```

```

}
}
}
}
/* Initializing and making the rule for corridor Beacons Corridor Han-
dling algorithm */
if (OriginGot) {
if (Beaconid == Corridor3_ID      && Previous_BeaconID == Corridor4_ID)
{
Previous_BeaconID = Beaconid;
X = CX3;
Y = CY3;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
} else if (Beaconid == Corridor4_ID      && (Previous_BeaconID ==
Corridor3_ID || Previous_BeaconID == Corridor5_ID)) {
Previous_BeaconID = Beaconid;
X = CX4;
Y = CY4;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
} else if (Beaconid == Corridor5_ID      && (Previous_BeaconID ==
Corridor4_ID || Previous_BeaconID == Corridor6_ID || Previous_BeaconID
== Corridor8_ID)) {
Previous_BeaconID = Beaconid;
X = CX5;
Y = CY5;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
} else if (Beaconid == Corridor6_ID && (Previous_BeaconID == Corri-
dor5_ID || Previous_BeaconID == Corridor7_ID)) {
Previous_BeaconID = Beaconid;
X = CX6;
Y = CY6;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
} else if (Beaconid == Corridor7_ID      && Previous_BeaconID ==
Corridor6_ID ) {
Previous_BeaconID = Beaconid;
X = CX7;
Y = CY7;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
} else if (Beaconid == Corridor8_ID      && Previous_BeaconID ==
Corridor5_ID ) {
Previous_BeaconID = Beaconid;
X = CX8;
Y = CY8;
FLoornumber = 2;
People_Shadow_Image(X, Y);
System.out.println(X + " " + Y);
}
}
/* End of Corridor Beacon handling case */

```

```

/* Lost case handling. When the Mobiletag fails to read the consecu-
tive Beacons then this statement will executes */
else {

/* Resetting the array */
if (lost_id_count > 2) {
lost_id_count = 0;
lost_id_Buffer_Cori[0] = 0;
lost_id_Buffer_Cori[1] = 0;
lost_id_Buffer_Cori[2] = 0;
}
lost_id_Buffer_Cori[lost_id_count] = Beaconid;
lost_id_count++;

if (lost_id_Buffer_Cori[0] == lost_id_Buffer_Cori[1]
&& lost_id_Buffer_Cori[1] == lost_id_Buffer_Cori[2]
&& (lost_id_Buffer_Cori[2] == Corridor3_ID
|| lost_id_Buffer_Cori[2] == Corridor4_ID
|| lost_id_Buffer_Cori[2] == Corridor5_ID
|| lost_id_Buffer_Cori[2] == Corridor6_ID
|| lost_id_Buffer_Cori[2] == Corridor7_ID
|| lost_id_Buffer_Cori[2] == Corridor8_ID)) {

Selected_CorridorID = lost_id_Buffer_Cori[0];
for (Object LOST_BEACON : Corridor_Beacon_Parameter) {
CORRIDOR_BID BC = (CORRIDOR_BID) LOST_BEACON;
// BC.getBeacon_ID();
if (lost_id_Buffer_Cori[0] == BC.getBeacon_ID()) {
X = BC.getCorX();
Y = BC.getCorY();
FLoornumber = 2;
People_Shadow_Image(X, Y);
Previous_BeaconID = Selected_CorridorID;
System.out.println("Your idxy " + MOBILE_TAG_ID + " " + X + " " + Y);
break;
}
}

}
}
}
}

```

Appendix 11: Source Code for Handling XML Files

```

package com.metropolia.electria.ButtonClickClass;
import java.io.File;
import java.util.HashMap;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import sun.reflect.ReflectionFactory.GetReflectionFactoryAction;
import com.metropolia.electria.MainAlgorithm.Coridoor_BEACON_INFO;
import
com.metropolia.electria.ButtonClickClass.Beaconimport_Structure;

public class Beaconimport {
public static HashMap<String, Beaconimport_Structure> AllBeaconVariable = new HashMap<String, Beaconimport_Structure>();

/**
 * Function for importing all variable of the Beacons from the file.
 * Function checks the data from file and collects the data to the
 * HashMap for
 * further processing.
 */
public static void Beaconimporting() {
String BeaconName ;
int BeaconID;

try {
File fXmlFile = new File("BeaconID_XML.xml");
if (fXmlFile.exists()) {
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(fXmlFile);
doc.getDocumentElement().normalize();

// Get the Beacon element by tag name directly
NodeList Beaconlist = doc.getElementsByTagName("BeaconName");
for (int index = 0; index < Beaconlist.getLength(); index++) {
Node Bnode = Beaconlist.item(index);
NamedNodeMap attr = Bnode.getAttributes();
Node nodeAttr = attr.getNamedItem("id");
//System.out.print("BeaconName: " + nodeAttr.getTextContent());
BeaconName = nodeAttr.getTextContent();

NodeList list = Bnode.getChildNodes();
Node node = list.item(1);
//System.out.println("                BeaconID: "                +
node.getTextContent());
BeaconID = Integer.parseInt(node.getTextContent());
//////// Storing the data into the Hashmap //////////
if(AllBeaconVariable.containsKey(BeaconName)) {
AllBeaconVariable.get(BeaconName).updateNewBvalue(BeaconID);
//System.out.println("HashMapcontains: "+ BeaconName+"--"+BeaconID);
}else{
AllBeaconVariable.put(BeaconName, new Beaconimport_Structure(BeaconName, BeaconID));
}
}
}
}
}

```

```
System.out.println("Hashmapcontainsnew value: "+ BeaconName+"--  
"+BeaconID);  
}  
}  
Coridoor_BEACON_INFO.BeaconVariable (AllBeaconVariable);  
}  
} catch (Exception e) {  
e.printStackTrace();  
System.out.println("File did not find");  
}  
}  
}
```