

Tuan Nguyen Gia

Implementation of a Smart Fridge by Using RFID and Web Technology

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

30 October 2012

Author(s) Title Number of Pages Date	Tuan Nguyen Gia Implementation of a smart fridge by using RFID and Web technology 62 pages 30 October 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Instructor(s)	Kimmo Sauren, Senior Lecturer
<p>The purpose of the project was to build a smart system which could be applied in daily life to improve the quality of life. A smart fridge helps users save time in buying necessary products by providing information about food inside the fridge. Furthermore, the smart fridge helps users take care of their health by warning about expired or opened products.</p> <p>The project was built based on HTML, PHP, JavaScript, MySQL, a Sirit Infinity 510 RFID reader, antennas and a fridge. HTML and PHP are programming languages for building the interface of the project. JavaScript, which is a programming language used in the client-side, was used to make the website become more fascinating. MySQL is a relational database management system for processing data. The RFID reader and antennas played important roles in the project. The set of RFID equipment had responsibility for reading and checking product data. The fridge was just a familiar fridge used at home.</p> <p>Users could add their favourite food and dishes to the system which stored the information and updated the shopping list, so users ensured that they always had the wanted products in the fridge. Moreover, the system had some special features in protecting users' health. The system listed all the expired products in the fridge and informed or warned about some opened products. Users would know the exact time when the product was opened and how long the product could be used after being opened. The main objective of the application was to prove that the RFID technology could be applied in daily life and was to provide a convenient, easy-to-use and smart system which could help people in improving the quality of life.</p> <p>The result offers users a high-tech and multi-functional system enhancing the quality of life. Moreover, the project shows that the RFID technology is possible to be applied at home.</p>	
Keywords	smart fridge, RFID, HTML, PHP, JavaScript, MySQL

Contents

Abstract

Abbreviations and technical terms

1	Introduction	5
2	The smart fridge	6
2.1	Overview of the smart fridge and its components	6
2.2	Introduction to RFID	7
2.3	Sirit Infinity 510 RFID reader	11
2.4	Web user interface	17
3	Design of the application	23
3.1	Use case diagrams	23
3.2	Activity diagrams	30
4	Implementation of a smart fridge	35
4.1	Setting up and testing the physical connection	35
4.2	Creating a reader profile	39
4.3	Software design	40
4.4	Software preparation	41
4.5	End-user application	42
4.5.1	Building a PHP file for connecting a PC with a reader	42
4.5.2	Running the file as script in the background	43
4.5.3	Building the user interface of the smart fridge	44
4.6	Testing and locating the system	53
5	Results and discussion	54
5.1	Problems and solutions	54
5.2	Benefits	57
5.3	Drawbacks	58
5.4	Further development	58
6	Conclusion	60
	References	61

Abbreviations and technical terms

CLI	Command Line Interface
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JavaScript	prototype-based scripting language
JIT	Just-In-Time
JSP	JavaServer Pages
LAN	Local Area Network
LBT	Listen Before Talk
PC	Computer
PHP	Hypertext Preprocessor
PNG	Portable Document Format
RF	Radio frequency
RFID	Radio Frequency Identification
RX	Receive
SQL	Structured Query Language
SSH	Secure Shell
TCP/IP	Transmission Control Protocol/ Internet Protocol
TX	Transmit
UML	Unified Modelling Language
VAC	Volt Alternative Current
VDC	Volt Direct Current

1 Introduction

Technology has been developing dramatically in many areas from biotechnology, computer science, electronic to wireless technology. The RFID technology is an example of the wireless technology, and it plays an important part in daily life and in industry. The RFID technology covers many areas and fields, so the idea of taking advantage of the popularity and development of the RFID technology to develop a new product to be launched to the market is feasible. Besides, in the modern and busy era, people need easy-to-use, convenient and smart products which help them to save time, to protect their health, and to improve the quality of life, so the smart fridge project system based on the RFID and web technology was implemented.

The main target of the project is to build a smart fridge with a user-friendly interface based on the RFID technology. Therefore, the aim of the project is that users could easily use the system without understanding how the technology inside the system works. The project was built as a demo for the RFID Lab Finland which is a non-profit organization, whose mission is to help other enterprises to improve their operational efficiency with the RFID technology. Therefore, the smart fridge system was built for implementing the main functions of the smart fridge. The advanced functions of the fridge were not included in the project. These advanced functions could be later studied when the project was able to run in a stable way. The project was built based on the RFID equipment which was supplied by other member companies such as Nokia, and GS1, so the project hardware was a combination of different hardware from different companies.

The goal of the project is to prove that the RFID technology could be combined with other technologies and the RFID technology could be applied at home for daily using. In addition, the report on this project could be used as a guide for a person who wants to build a system using the RFID technology or for other RFID technicians who want to develop the smart fridge system.

2 The smart fridge

2.1 Overview of the smart fridge and its components

Working process

The smart fridge is a comprehensive system combining physical hardware, a software application, a computer, and a screen. The working process of the system is described in figure 1.

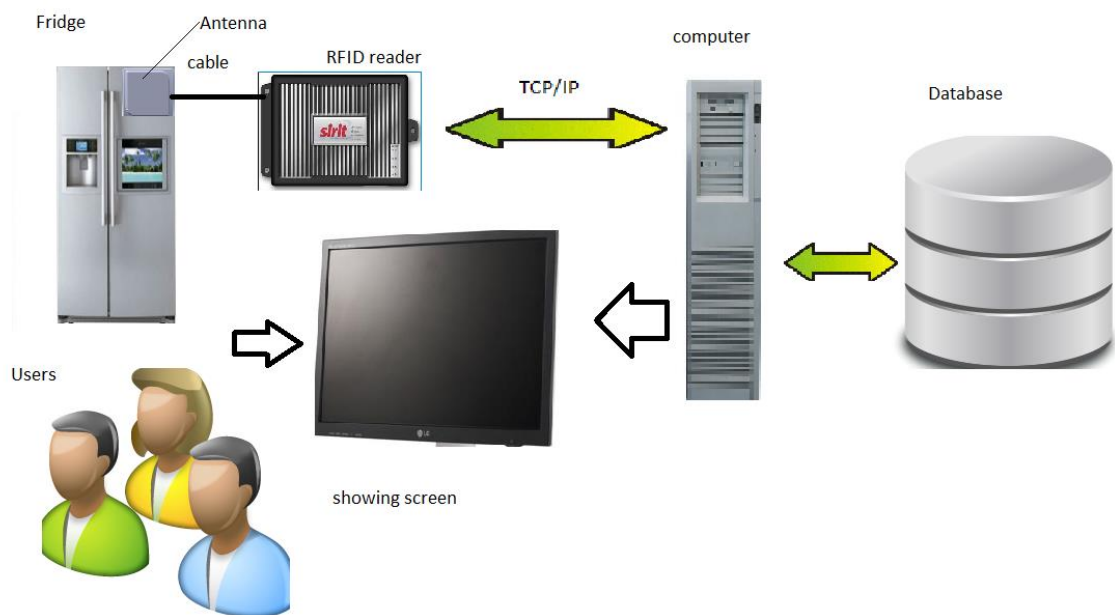


Figure 1: The smart fridge system.

As can be seen in figure 1, an antenna is attached to the fridge for reading a tag information including tag id, password, and mask. After the tag information is sent to the reader through the cable of the antenna, the tag information is stored in the database or sent to the computer through TCP/IP or a serial cable. The computer with a running application gets the data sent by the reader and shows the information in the user interface through a browser. End users can use the smart system through a computer browser or a mobile phone browser.

Hardware

The physical hardware includes an RFID Sirit Infinity 510 reader with two antennas, a fridge, a computer, cables, and a switch. While an antenna attached to the fridge verifies the tag located inside the fridge, the other antenna is used for registering a new tag or verifying a tag taken. The equipment which includes a computer for running and deploying the application and a reader is connected in the LAN through a switch. In order to increase the effect of the user interface, wide or touch screens are recommended.

Software

The software and applications used in the smart fridge project were a database, PHP, JQuery, JavaScript and Eclipse. In order to simplify the complexity of the smart fridge system, Xampp, which is a useful and simple application, was used because Xampp has MySQL, PHP5, Apache, and many other utilities. In addition, Xampp is a free application supporting many operating systems from Windows, Linux, Solaris to MAC OS, so it is simple to install Xampp in many systems. However, using Xampp was more complicated because some configuration values in the configuration file had to be edited. Eclipse was used because of its user friendly interface and development platform. Some ready-made JavaScript and PHP libraries, which are well-known, secured and free of charge, were used. JQuery is one of the useful libraries for harmonizing with JavaScript. Finally, a Firefox browser was used for showing the view of the smart fridge to final end users.

2.2 Introduction to RFID

RFID stands for radio frequency identification which is an example of a wireless system using a radio frequency electromagnetic field. When a tag is placed near antennas, a tag data will be read and transferred to the reader through a radio wave. The reader is connected to the computer, so that the data can be used for many different purposes depending on the businesses. The basic RFID system is described in figure 2. [1]

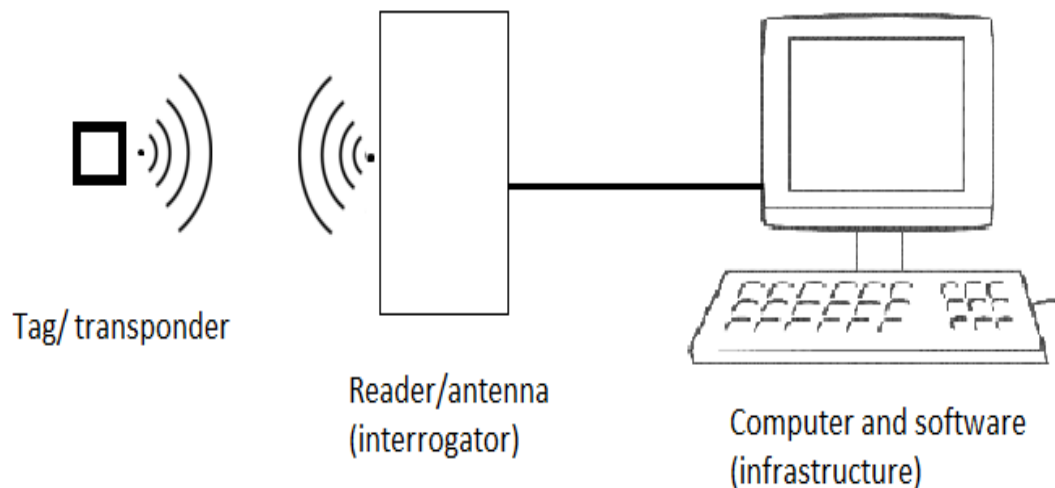


Figure 2: Basic RFID system

Figure 2 describes the basic structure of an RFID system which usually combines three components: tag/transponder, reader/antenna, and computer. The tag usually uses a silicon microchip to store unique information which is called "tag_id". Besides "tag_id", other information can be stored depending on the type of the tag. "tag_id" is transferred in the RFID tag reading process.

Many types of RFID systems are introduced in RFID show rooms and fair centres; however, setting up and using them for reducing the cost of a business is challenging. In general, all types of RFID systems provide a typical function generating data through wireless connections, but in detail, different RFID systems, which have different functions, are used for distinct purposes. For example, the key system in the Helsinki Metropolia University of Applied Sciences is built based on the RFID technology. Each student/staff member has a specific key which stores a personal authority for accessing doors. A user who wants to access a room must show his/her key at the reader attached to the door because the reader is able to read a tag at a distance of a maximum of 10 centimetres. The reader reads the data from the key then decides whether to allow access or not. This system is totally different from a system in logistic industry using expensive readers and antennas. The reader in industry can read a tag at a distance from 1 meter to 100 meters depending on the tag types and the reader types, so the cost of setting up an RFID system varies. [1]

There are two main categories of RFID systems: passive systems and active systems. Passive RFID tags which do not have transmitters and their own power sources, reflect back energy coming from the antennas of the reader. Active RFID tags have their own transmitters and power supplies. Besides the two main categories, semi-passive tags are used in some applications. [1]

The RFID technology is applied in many applications and many fields including asset tracking, manufacturing, supply chain management, retailing, payment system, security and access control. RFID systems used in these fields are different from price to function. In asset tracking, a company attaches RFID tags to items or assets which are usually lost or hard to find. Those items can be easily tracked by an RFID real-time locating system using RFID beacons. In storage industry, every product has one RFID tag with a unique number. When these tags are in the read-range, which is a specific distance in which an RFID reader is able to read the data, the RFID reader reads all the tags information through antennas. Therefore, all products can be efficiently managed. [1; 2]

Active RFID system

Active RFID tags are usually applied in industries which use large cargos or items needed to be tracked over a long distance, from 20 meters to 100 meters. A transponder and a beacon are the two main types of an active RFID system. A transponder is usually in the sleeping mode which helps to save battery life when it is outside the read-range. When a transponder is in the read-range, it receives a signal and is woken up to broadcast a unique "tag_id" to the reader. A beacon, which differs from the transponder, is often used in real-time locating systems that receive the information of a tag's location in an interval time. The interval time, which is every five seconds or five times a day, can be modified depending on the user's purposes. Using beacons is complicated because in order to get "tag_id", at least three antennas must be set in specific areas where assets are tracked. [1]

Active RFID tags have a broad read-range with a limit up to 100 meters. In most cases, information transmitted from tags is reliable as the information is broadcasted to the reader. In some special situations, information transmission is affected by the environment. The price of one active RFID tag varies from 10 dollars to 50 dollars depending

on the form-factor, the tag's memory, and the tag's battery. Besides, the price is influenced by a motion sensor, tamper detection, or a temperature sensor. Although the price of an active tag is quite high, an active RFID system is still used in real-time asset monitoring and many other applications because it provides a better layer of security than a passive RFID and the price of an active RFID reader is lower than the price of a passive RFID reader. Tag life is from three to eight years depending on the tag broadcast rate. [1; 3]

Passive RFID system

A passive RFID tag has no transmitter or power source, so a passive RFID tag just reflects the signal sent from the reader's antennas. The price of a passive RFID tag varies from 10 cents to 40 cents. A passive RFID tag is very simple and cheap, so tag maintenance is unnecessary. Hence a passive RFID tag is used in many companies and many fields. However, the read-range used for a passive RFID tag ranging from a few centimetres to nine meters is shorter than the read-range of an active RFID tag. [1]

A passive RFID transponder includes a microchip attached to an antenna. There are many ways to pack a passive RFID transponder depending on the purpose of the application. A transponder can be embedded, for example, in a plastic key or card. In some special covers, it has capability to resist coldness, heat or even cleaning chemicals. [1]

A passive RFID tag is capable of working at different frequencies from low frequency and high frequency to ultra-high frequency. A low frequency passive tag can be read in the read-range of 0.33 meters with operating frequency at 124 kHz, 125 kHz, or 135 kHz. A high frequency passive tag can be used within 1 meter at 13.56 MHz, while an ultra-high frequency passive tag can be read from 3 meters to 9 meters with a band from 860 MHz to 960 MHz. Radio waves at different frequencies have distinct behaviours influencing applications. The passive tag has a life cycle of 10 years depending on the environment around the tag. [1; 3]

2.3 Sirit Infinity 510 RFID reader

The infinity 510 reader is a flexible, adaptable multifunctional RFID system applied in many fields and many RFID applications. It is easy to set up 510 RFID reader connections, which are described in figure 3.

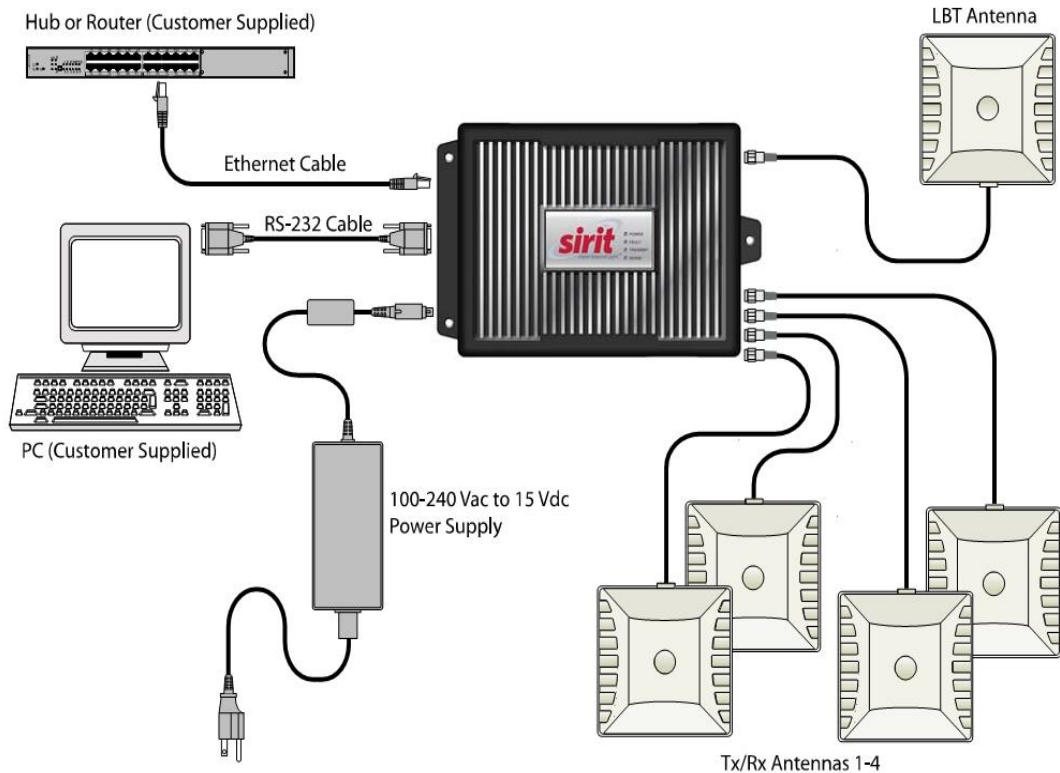


Figure 3: Sirit 510 reader's physical connection. Reprinted from Sirit [4]

As can be seen in figure 3, the reader has four TX/RX antenna ports. These antennas are special components mainly used for reading information from RFID tags and writing data to tags. In addition, the Sirit 510 reader offers one extra LBT antenna which is used to look for a free channel or a network device. All antennas are connected to the reader through cables which have different lengths depending on users' purposes; however, the longer the cable the more "loss signal" strength occurs. The reader has three ports including an Ethernet port for connecting to LAN by an RJ-45 cable, a serial port for connecting to a PC by a serial cable and an I/O port for controlling the reader's input-output. There is no I/O port in figure 3 but in reality, it is placed between the Ethernet port and the serial port. The I/O port, which is used for digital input-output, provides four optically isolated input signals which have values ranging from 5 to 24

VDC and from 1 to 5 mA. Digital inputs can be used to trigger the reader for tag reading or configured for an external read trigger from other devices. Digital output signals provided by the reader have values from 3 to 40 VDC for voltage and a maximum of 100 mA for current. Both digital input and output can be manipulated by an I/O controlling command. The Sirit 510 reader uses 15 VDC from the adapter which converts 100-240 VAC input to 15 VDC output. [5]

Reader users can send specific CLI commands to the reader for administrating the RFID system through channels which are automatically chosen by the reader. The channel number is unique, so the next ready channel number is incremented by the latest channel number used. The Sirit 510 RFID reader can be accessed by a human interface or a machine interface. The human interface allows reader users to send a command directly to the reader and receive responses from the reader. Reader users can use some ready-made applications such as Putty to connect to the reader through the Ethernet port or use their own applications to send commands. A machine interface, which differs from a human interface, supports direct connection between the event channel and the reader's command. The machine interface has two default ports such as "50007" port and "50008" port defined by the producer. The "50007" port is used in the bidirectional channel and reserved for the reader's command and the reader's response while the "50008" port is used in the unidirectional channel and reserved for the event channel, which sends an asynchronous report. [5]

The Sirit 510 reader helps users in management due to a large number of useful commands which are strings of characters invented by Sirit-RFID producer through some RFID standards. Commands, which can be sent by SSH or any application with a socket connection to the reader, are categorized into three main types of commands, including "GET" commands, "SET" commands and "EXEC" commands. The "GET" commands are used to retrieve values of a specified reader configuration variable. The "SET" commands are used for setting values for reader configuration variables. The "EXEC" commands are applied for running the reader's functions. Response commands have four types including "GET" responses, "SET" responses, "EXEC" responses and "ERROR" responses because the "ERROR" responses are reserved in case of the system's malfunction or error request. The "GET" responses show "OK" and information of a configuration variable such as "tag_id", password, or a data value. The "SET" and "EXEC" responses only show a string "OK". When the com-

mand's syntax or a specific type of the command is not supported, the reader will respond with an error response message which is a string of characters showing the error type. All error responses start with "error" string and continue with the name of different error types such as "error.internal.processing_error", "error.app.not_running". In order to help users in programming the system, different data types are supported such as "bool", "int", "string", "enum", "enumlist", "array", "list", "compound", and "compound list". These data types are familiar to all programmers, so it is easy to write an application which can take advantage of these data types. [5]

Furthermore, users configure the reader easily through a reader firmware written by Sirit RFID producer. The firmware, which is secured, is installed in the reader, so users just connect to the reader at the default IPv4 address of the reader through the firmware interface in any browser. The default profile created by Sirit, which users can use to log into the system the first time, has two usernames: administrator and guess. If a user logs in with the guess account, he/she only sees some the reader values which are unchangeable. Meanwhile, if a user logs in with the administrator account, he/she can create his/her username, password and change all configuration values. [5]

The RFID reader firmware is exclusive software of Sirit and is possibly updated to the newest version. Figure 4 shows the interface of the reader firmware which is shown in the browser when connecting by the default IPv4 address of the reader.



Figure 4: The firmware interface of the Sirit 510 reader [5]

The interface is user friendly and easy-to-use. When a user wants to use the system, he/she must log into the system by using the “login” button in the top left of the interface. When a user clicks this button, one form with a username field and a password field will show in the interface. The reader supports the profile in managing the system. With the administrating account, an administrator can create different profiles which are used in distinct situations. When a ready-made profile is loaded, all configuration values will be loaded to the reader. There are two types of configuration: basic configuration and advanced configuration. Basic configuration can be configured and used by all users while advanced configuration is recommended to technical users who have knowledge of the reader and radio transmission. [5]

For a basic configuration, the reader administrator can add and modify some necessary typical reader values which are shown in the list box with an explanation. He/she can manage his/her profiles through the “manage profile” mode in which the user

can add, remove, edit, and load ready-made profiles or restore the default. The "Setting up Ethernet/LAN" mode, which includes an IPv4 address, a DNS server and a domain name, is important in connecting between the application and the reader in a local area network because the reader cannot be accessed with unsuitable values configured in this mode. However, the Sirit infinity 510 offers an extra serial port which can be directly connected by a serial cable from a PC to the reader in case the Ethernet/LAN does not work. The user can use the default mode provided by Sirit or set up a serial port configuration with specific values which are clearly explained in the note form in the basic configuration. The serial connection can be set by using Putty software or any software used for a serial, TCP/IP connection. After the connection is set, the reader user can give commands through a CLI and he/she can fix the IPv4 problem by modifying the Ethernet configuration information. Besides, the basic configuration mode provides an antenna configuration which includes all the necessary values such as conducted power, attenuation, cable loss, gain, gain units and mux sequence which is for setting the antenna's order in reading the signal. These values are set due to the cable's length and users' purposes. Last but not least, the "set regulatory" mode, which is used for setting up working regions and sub regions, is provided. All values of this mode are initially set by Sirit; however, the user can change it when he/she deploys the system in a specific geographic region. [5]

For an advanced configuration mode, the reader requires users who understand about the reader and have knowledge of an RFID to set configuration values because of the complexity of this mode. The "Firmware management" mode allows users to update the reader firmware to the newest version, or rollback the reader firmware to the previous version. A firmware version, which has unique value, is a string of numbers defined by producer. However, when the reader manager does not store the previous working version of the firmware, the reader cannot continue a rollback process. Unlike the other modes, the "firmware management" mode does not have a reset button because when different firmware versions are applied, the user interface will change. Sirit always recommends users to use the newest firmware version which supports new features and functions. Before updating the newest version, the previous working firmware version is kept in a safe place in order to keep the reader working. In addition, the Sirit 510 reader supports a "transfer file configuration" mode which allows the reader manager to transfer the current working configuration file into a text file, a XML file or load an existing configuration file with a XML or text format. The advanced configuration mode provides a CLI which is used for sending commands and receiving

responses from the reader. Furthermore, the advanced configuration mode supports the "expert configuration" mode comprising a "setup" sub-mode, a "tag" sub-mode, a "version" sub-mode, an "information" sub-mode, a "communication" sub-mode, an "antennas" sub-mode, a "digital IO" sub-mode and a "modem" sub-mode. Every sub-mode of the "expert configuration" mode is similar to the sub-mode of the basic configuration but it has more information and detail values for configuring. The user needs to understand every value used in the "expert configuration" mode because different values of the reader's variable give different results which may delimit some reader's functions or stop the reader working. The reader has the "user application management" mode which the user can use to upload a Java or Python application written by the third party. In addition, the user can change the reader's working mode by modifying the "operating" mode which includes three different sub-modes such as an "autonomous" mode, a "polled" mode, and a "standby" mode. However, in the newest firmware version, Sirit updates some functions and the reader's operating modes, so "autonomous" and "polled" modes are obsolete and replaced by an "active" mode which is used for an asynchronous event in which the reader reads the tag information continuously and stores the tag data in the database, after that the reader reports automatically. The user can register to the event or poll to the database or combine functions of two modes in the "active" mode. The last mode which does not support the tag information reading, responding and reporting automatically is a "standby" mode in which the reader does not send any signal or transmit any energy automatically. The process of reading and responding is done by using commands. [5]

After setting up the reader with basic and advanced configurations, the user can use a status mode to check the current stage of the reader, to view all tags in an area and to view a log which is important feature for debugging. The view log helps debuggers see what changes other users have done, so they can fix problems or jump back to some previous working states. When using the reader, the user must consider the reader's database capacity because the reader's database capacity stores a maximum of ten thousand tags. The reader provides two methods including the method of reading and reporting tag information immediately and the method of purging the content of the database for solving the problem of the database capacity. [5]

2.4 Web user interface

In general, when a user uses a browser to access a website hosted by a server, a website receives a request from the user and responds the necessary data to the user. A website is created by combining many web pages including texts, images, audios, and videos. A web page is a plain text document and is written following a hypertext markup language (HTML) format. The web page must be hosted by a server that is a physical hardware computer running many services in order to serve users' needs. When the web page is placed on the server, it can be connected through the Internet or a private local area network (LAN) by request/response protocols such as a Hypertext Transfer Protocol (HTTP), a Hypertext Transfer Protocol Secure (HTTPS). Basically, the web includes the client side and the server side which can be placed in the same system or in different systems depending on applications. At the client side, a browser sends a request under the Uniform Resource Locator (URL) form to a server and waits for a response. When the server receives the request, it analyzes the request URL and sends the response back to the browser. If the server cannot understand the URL, it sends some errors or warning messages to the browser. [6; 7]

HyperText Markup Language (HTML)

Hypertext markup language (HTML) is a main language for displaying web pages in a web browser. The first HTML document was created by Berners-Lee in 1991. The newest version of HTML is HTML 5 which has been popularly applied in many web pages. A HTML form is simple with tags enclosed by angle bracket. HTML tags are mostly represented with a pair of tags: the opening tag and the closing tag whilst some unpaired tags which are known as empty elements. For example, one pair of HTML tags for declaring a paragraph is "`<p>`" and "`</p>`", while unpaired tag such as "``" is for inserting images to a HTML document. The HTML document allows other programming languages embedded in such as PHP, JavaScript, and JSP. Furthermore, HTML supports Cascading Style Sheets (CSS) which is a style sheet language used for describing the look and formatting of a document. Three CSS styles are inline, internal, and external. CSS is applied to many pages by adding an external CSS file in the external style while in the inline style and in the internal style, CSS is used for one single occurrence of an element and one single document respectively. The structure of a basic HTML page is described in figure 5. [8; 9]

```

<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>

```

Figure 5: Structure of one basic HTML file.

Figure 5 shows the basic structure of one basic HTML page starting with the “<HTML>” tag and ending with the “</HTML>” tag. The <head> element is used for containing all the head elements such as “<title>”, “<base>”, “<link>”, “<meta>”, “<script>” and “<style>”. The “<title>” tag is used for presenting the name of web page in browser. The “<body>” tag is for defining the body of the web page which contains all contents of HTML document such as texts, hyperlinks, images, tables, lists. The “<p>” tag specifies a paragraph in the HTML document. In addition, a HTML element can be provided some additional information through attributes which are located in the opening tag and created by two parts: a name and a value. The name and value of the attribute are separated by equal sign. A HTML code “<p lang=“en-us”> this text is in English</p>” has the attribute name: “lang” and the attribute value: “en-us”. [8; 9]

PHP: Hypertext Preprocessor

PHP, which is a server-side scripting allowing websites to be dynamic, was created by Rasmus Lerdorf in 1994. PHP can be written in separated PHP files or embedded in HTML files depending on the size of PHP code and programmers’ willingness. There are more than 20 million websites and 1 million web servers using PHP because of its advantages. PHP is an open source programming language so many groups and organizations are using, and developing it. As a consequence, many PHP websites are built for discussing, researching about PHP. For instance, the PHP.net, a reliable and popular website, created by a PHP group is often used as a guide for PHP program-

mers. In addition, it is easy and quick to build websites by using PHP. All PHP scripts are interpreted by web servers with a PHP processor module; therefore basic PHP scripts can be built without compilation. [10; 11]

Furthermore, PHP, which is an object oriented programming language supporting classes, functions, inheritances, abstract classes, and interfaces, provides the clear structure of the program, so the program creating, modifying and maintaining becomes easier. PHP has ability to work in many operating systems such as Linux, UNIX, and many versions of Windows operation system, so a programmer can choose the best operating system to develop PHP depending on the familiar operating system, programming tools installed and the production environment. Moreover, Model-View-Controller (MVC) is supported by many PHP frameworks: Zend framework, Prado, CakePHP, CodeIgniter and many other frameworks. [10; 11]

PHP scripts are located between “<?php” and “?” and there is no limitation for a number of “<?php” and “?” in PHP files or HTML files, so the programmer can implement necessarily PHP scripts anywhere in the website. There are two ways for commenting in PHP. The first way uses “//” for a sentence comment. The second way uses “/*” and “*/” for a paragraph comment. All comments are not interpreted when the PHP script is executed. The basic PHP script is showed in listing 6.

```
<html>
  <body>
    ?php
      // a sentence comment
      /* a paragraph comment or a block comment*/
      $a = 0;
      echo $a;
    ?>
  </body>
</html>
```

Listing 6: The basic PHP script

Listing 6 represents the PHP script which starts with “<?php” and ends with “?”. PHP can be embedded in a HTML document, so in listing 4, the PHP code is located in

between "`<body>`" and "`</body>`" tags for displaying the web page content. A PHP variable starts with "\$" character and supports many programming language types such as "string", "int", "double", "float". The "Echo \$a;" method is used for printing the value of the variable "a" on the screen. Semicolon ";" in the PHP script is used for ending one PHP statement.

Apache HTTP Server

Apache acts as a web server with the main function to parse any file which is requested by browsers for displaying correct results. An Apache server has many features such as an UNIX threading, a new build system, a multiprotocol support, a new apache API, an IPv6 support, a filtering, multi-language error responses, a simplified configuration, a native Windows NT Unicode support. In addition, Apache features include regular expression libraries updated, password-protected pages for a multitude of users, asynchronous supports, overriding configurations, a name virtual host directive, reduced memory usage. Apache is free, flexible and quite powerful, so many students and teachers in many technical schools and universities such as the HCM National University, the Helsinki Metropolia University of Applied Sciences used Apache servers for studying and teaching. According to Netcraft website, Apache servers have been running over 27 million the Internet servers. [10; 12; 13]

MySQL

MySQL is an open-source relational database management system allowing PHP and Apache to work together to access the correct data in the readable format. MySQL is popular because it has many features such as multi-layered designs with independent modules, very fast thread-based memory allocation system supports, many data types supporting, and many others. [13; 14]

MySQL can be interacted through commands which are "CREATE", "INSERT", "UPDATE", "DELETE", "ALTER". The "CREATE" command is used when a database user wants to create new databases, new tables. In order to insert values for tables, the "INSERT" command must be used. The "UPDATE" command is used for updating new data for the database. When a user does not need any table, any database, he/she can delete them by the "DELETE" command. The "ALTER" command is used in case a

user wants to change tables such as replacing columns or modifying names. A user can use other commands for selecting the data existing in the database. The “SELECT” command is always used for selecting the data. However, it is hard to select the desired data with only the “SELECT” command because the data may be large, complicated. Therefore, other functions and commands such as group clauses, group functions and conditional clauses are introduced to support the “SELECT” command in selecting the correct and necessary data. Group clauses include “GROUP BY”, “ORDER BY”, group functions consist of “COUNT()”, “AVG()”, “SUM()”, “MAX()”, “MIN()”, “GROUP CONCAT()” while conditional clauses comprise of “WHERE”. Besides, many other commands and functions are supported in MySQL with the purpose of helping users in manipulating the data in the database. All commands for MySQL are introduced in mysql.com website with full explanations and examples. [14]

In managing data, a security plays an important role, so MySQL supports a privilege and password system to guarantee the security. Furthermore, passwords are encrypted when users connect to a server. Administration in MySQL is manageable because MySQL offers the command line program and the graphical user interface. The command line program supports “mysqladmin” and “mysqldump” commands. The “mysqladmin” command is applied for checking the server configuration and current status. The “mysqldump” command is used for backing up the database while the “mysqlcheck” command is used for maintaining the database. [13; 14]

JavaScript

JavaScript is a programming language used in the client side and embedded in a HTML document. In order to use JavaScript in the HTML document, the JavaScript code must be located between the “<SCRIPT>” tag and the “</SCRIPT>” tag. The example of the JavaScript code is shown in listing 7.

```
<html>
  <head>
    <title>hello world!</title>
    <script language="JavaScript">
      // This is a comment
      function show_alert(){
        alert("this is JavaScript");
      }
    </script>
  </head>
</html>
```

Listing 7: JavaScript in the HTML document

Listing 7 shows that JavaScript is embedded in the HTML document between the “<head>” tag and the “</head>” tag. The JavaScript code starts with the “<script>” tag and ends with the “</script>” tag. JavaScript has a single line comment and a block comment. The single line comment needs “//” at the beginning of the line while the block comment needs to have “/*” at beginning and “*/” at the ending.

The main function of JavaScript is to make websites become interactive. Therefore, programmers can use JavaScript to validate the data, respond directly to users, control multiple frames navigation and carry out many other activities on browsers. JavaScript does not have any class, although JavaScript is object oriented programming language. Due to lacking of the class, objects can be inherited from each other or from the object prototype chain. JavaScript has some ready-made objects; however, programmers can create or delete their own objects. JavaScript is an interpreted language that can be executed and interpreted by browsers without any preliminary compilation or conversion. However, it is possible to interpret or compile the script by using the just-in-time (JIT) compiler depending on browser’s decision. With the JIT method, JavaScript will be run faster. JavaScript and JavaScript JIT compilers are supported by almost all browsers. [15; 16]

3 Design of the application

Designing, which is a process of planning a software solution, is very important for building the final application. A software design, which is unapparent to users, is often presented in the form of UML diagrams. Depending on scales of projects and ways of building projects, a number of diagrams and types of diagrams can be different. In this project, functions and system activities of the fridge are very important, so “use case” diagrams and “activity” diagrams were created.

3.1 Use case diagrams

A use case diagram describes interaction between actors and the system. An actor carries out actions to the system, and in the meanwhile several actors can do one action simultaneously. Each action is described in one “use case” diagram. In the project, six “use case” diagrams were built, and accordingly the smart fridge has six main functions which users can use. The six diagrams are described in figures 8-13.

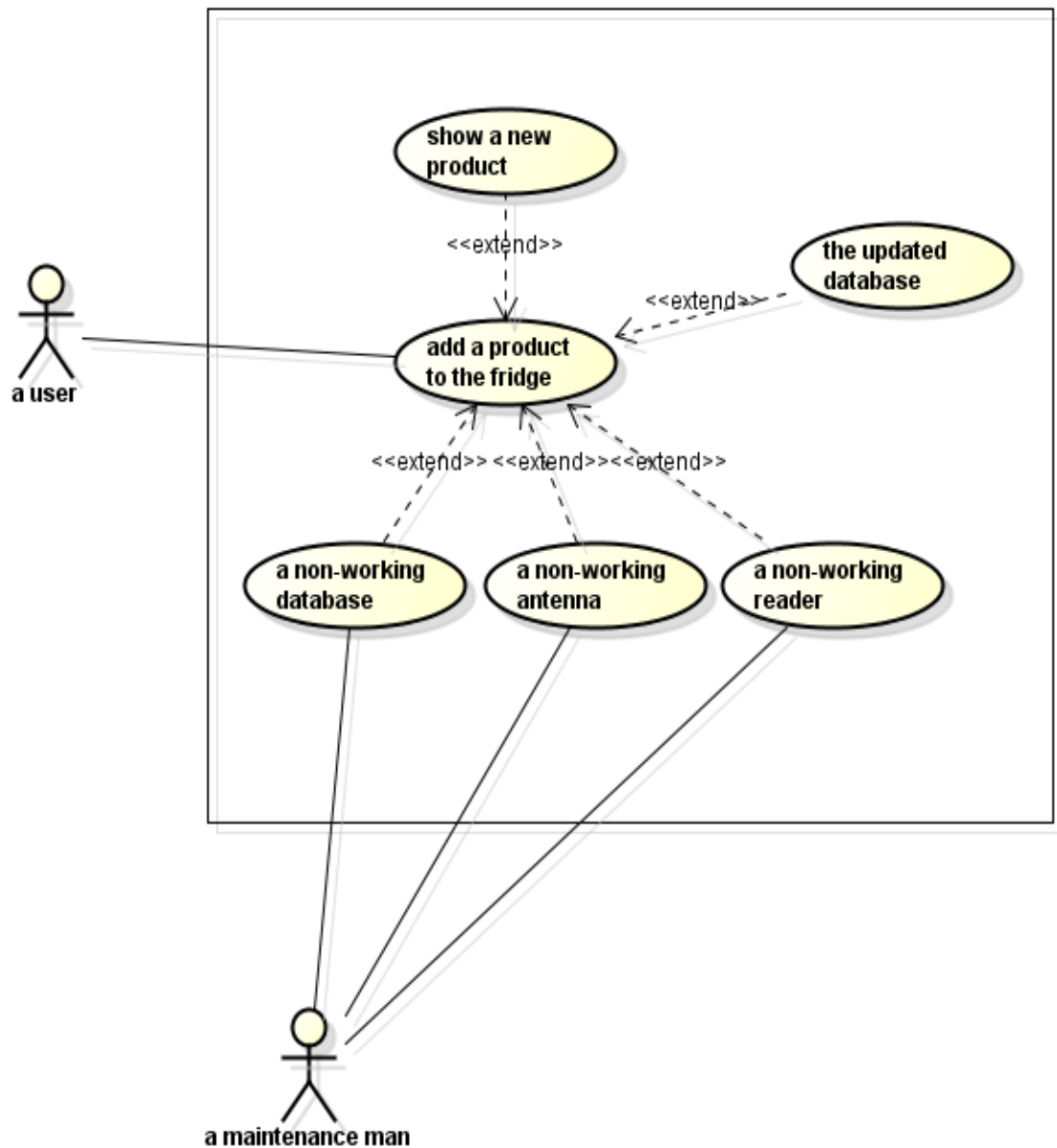


Figure 8: A use case adding a new product to the fridge.

Figure 8 demonstrates the working process of adding a new product to the fridge and its relationship with users and a maintenance man. Firstly, when a user brings a product close to the first antenna read-range, the product id is stored in the database. After that, if the product is put inside the fridge, the second antenna attached to the fridge reads and stores the product id into the other database. The maintenance man needs to be ready to fix the system when the database, or the antenna, or the reader is not working.

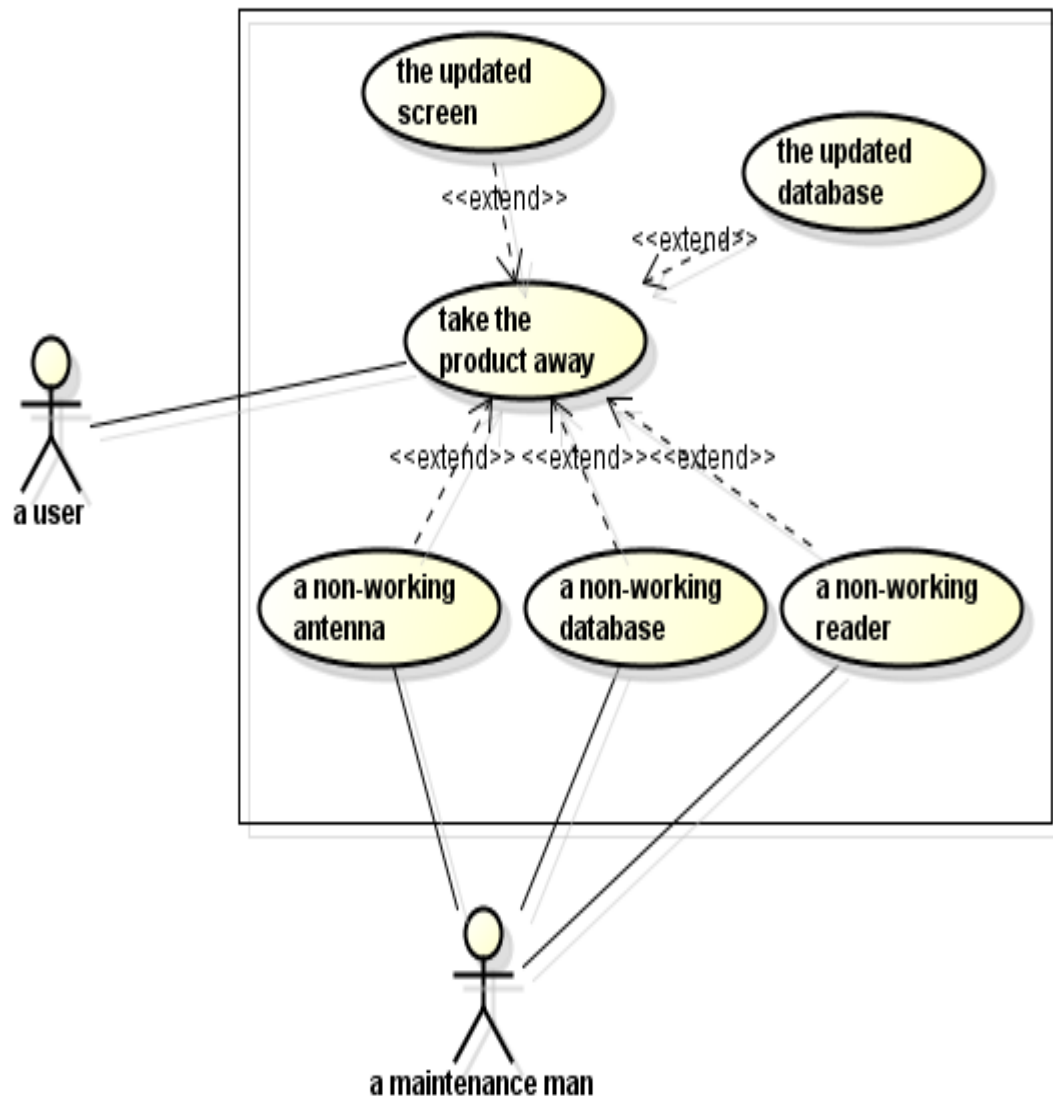


Figure 9: A use case taking the product away

Figure 9 presents the process of taking one product out of the fridge. When the product is taken away, the database updates the information such as deleting product id in the database or changing status: inside or outside the fridge. Finally, the screen is updated with the new information. The maintenance should be done regularly to fix all system problems.

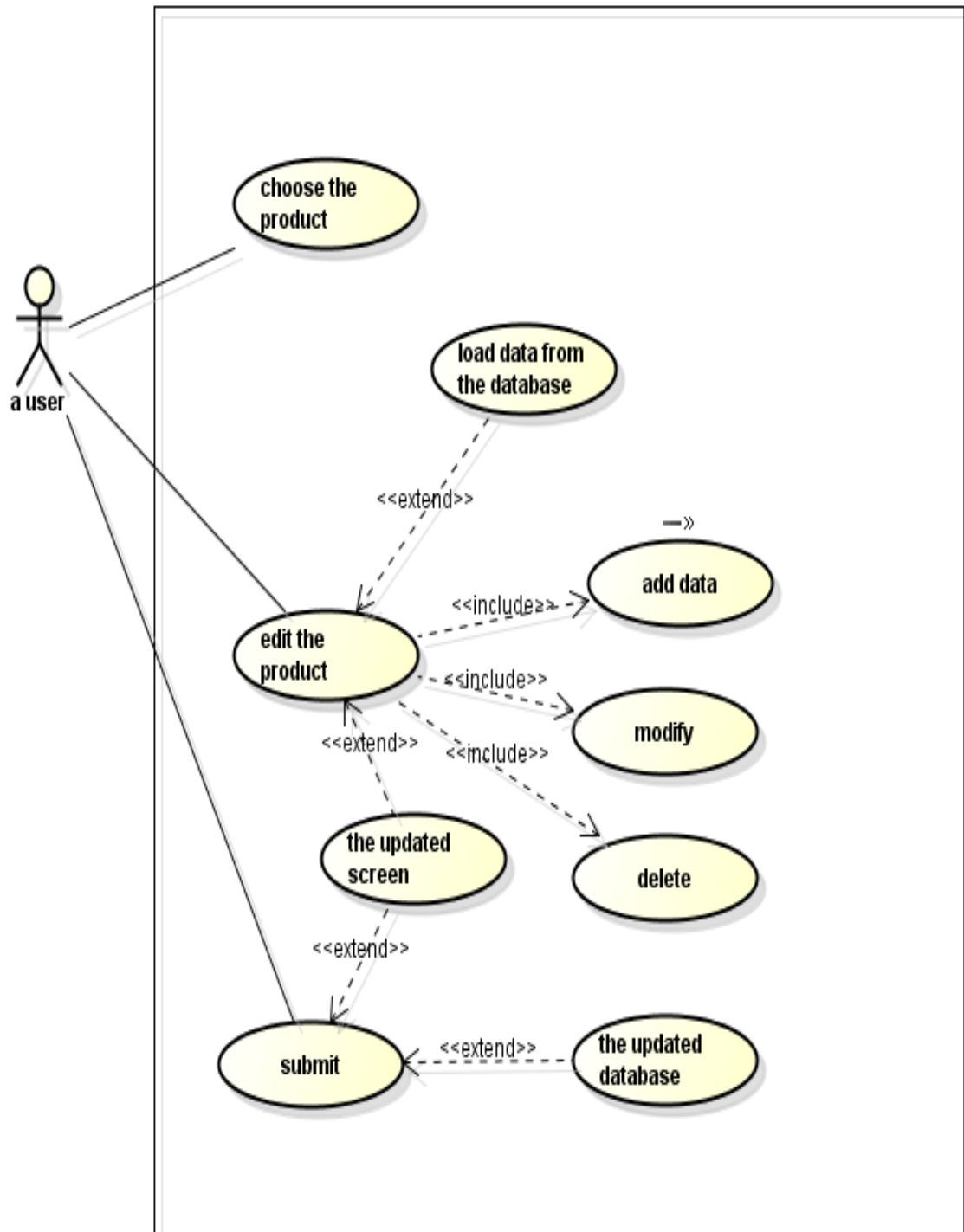


Figure 10: Use case editing the product

Figure 10 shows the process of editing the product. At first, a user needs to choose one product id on the screen. The user can add, edit the information or delete the product data. The updated information of the product will be shown by notifications on the screen.

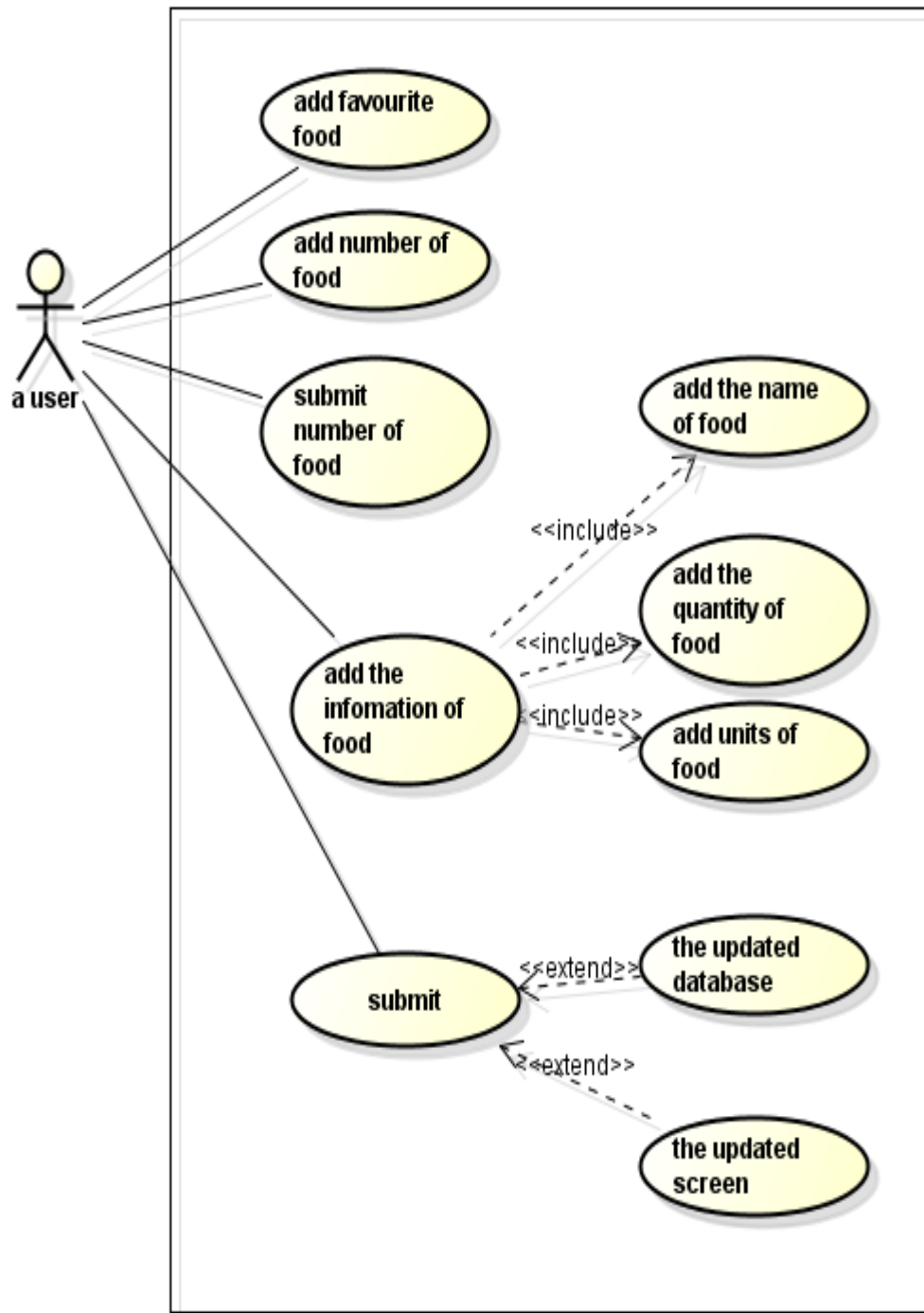


Figure 11: Use case adding favourite food

Figure 11 presents the process of adding favourite food to the system. Firstly, a user activates the adding favourite food process by using the "add favourite food" button in the main view. After that, he/she submits the information including unit, name, and quantity of food into the database. The updated data will be shown on the screen.

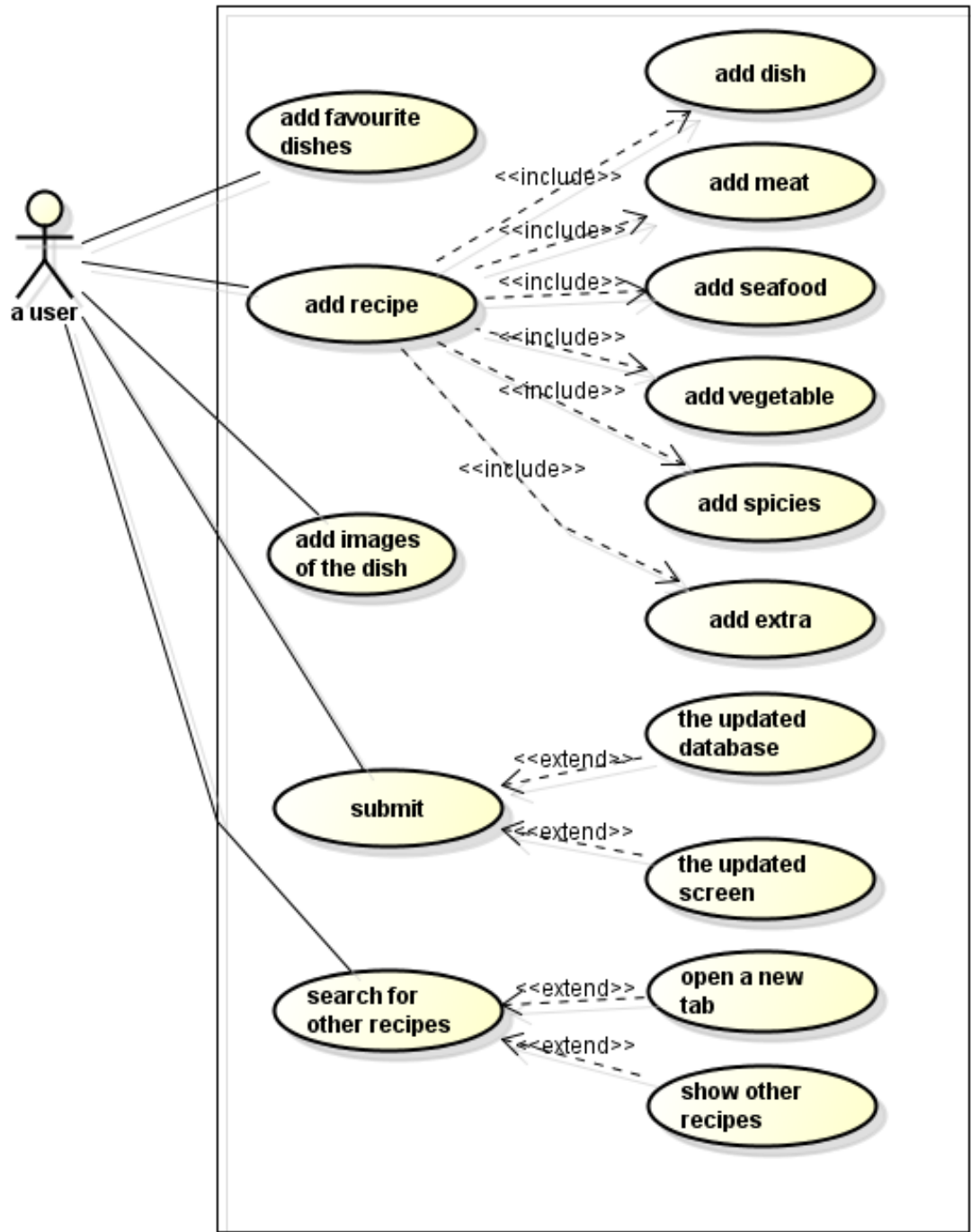


Figure 12: Use case adding the favourite dish

Figure 12 represents the process of adding the favourite dish to the system. A user chooses the "add favourite dish" button to activate the process of adding the favourite dish. The information of the dish such as "name", "meat", "seafood", "vegetable", "spices", and "extra ingredients" can be added through boxes. If a specific dish does not have meat or vegetable, these boxes can be empty. Besides, the user can upload images for the dish into the database. If the user does

not like recommended recipes, he/she can use the “searching other recipe” button which opens a new tab in a browser and shows information about other recipes which are retrieved from other cooking websites with thousands of dishes and recipes.

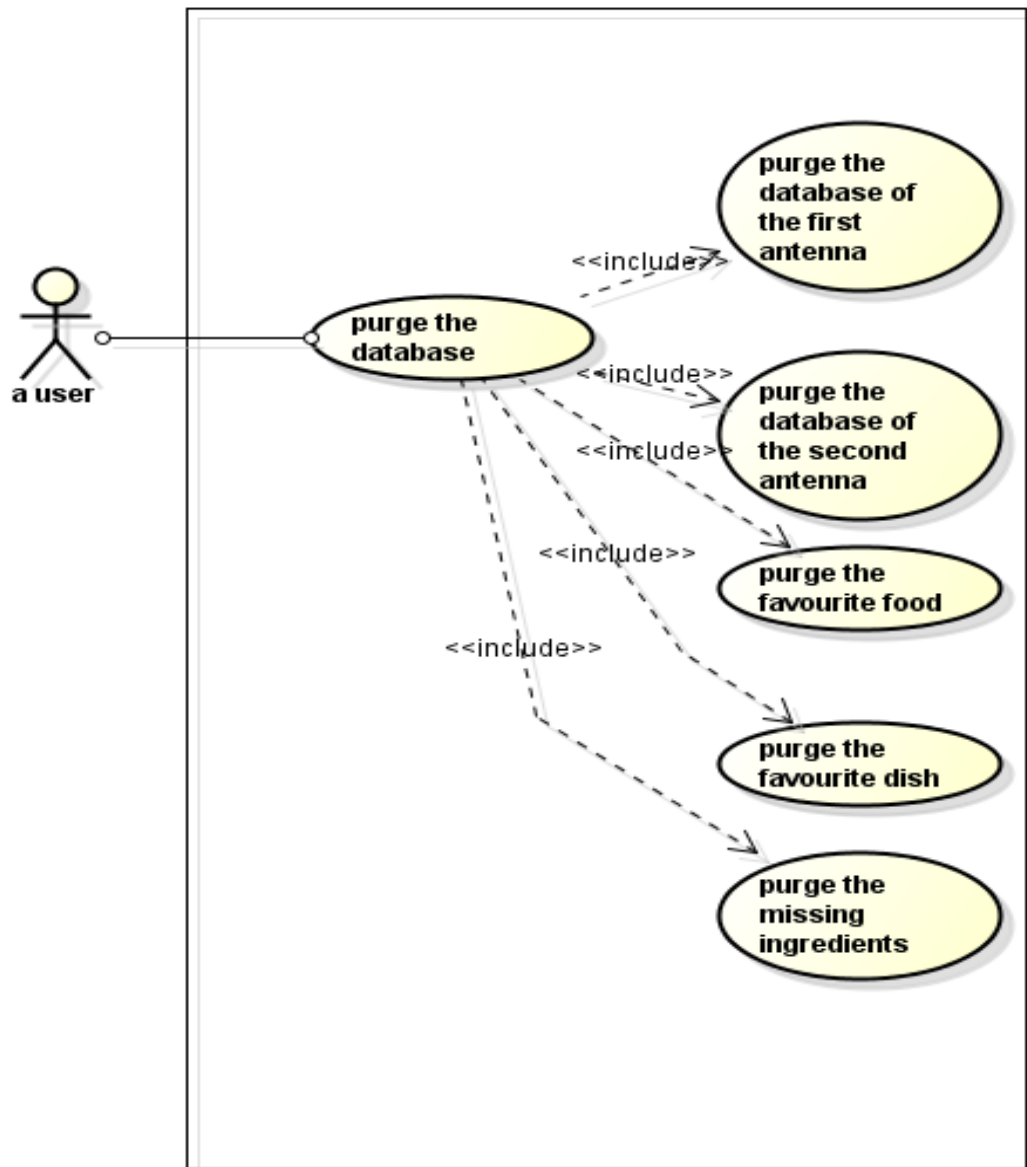


Figure 13: Use case purging the database information

Figure 13 demonstrates the process of clearing the database information in the system. The different information is deleted depending on which button is activated. A user can use the button to clear the information of favourite food, the favourite dish, and the missing ingredients. Moreover, when the user uses all purging buttons, he/she clears all the database information. Consequently, the system begins at the initial stage when the next using time.

3.2 Activity diagrams

An "Activity" diagram, which is a graphical representation for describing the system activities, interacts with actions, iteration and concurrency. When users have done some actions to the system, the system will carry out some internal processes. In the project, five activity diagrams, represented in figures 14-18, were built.

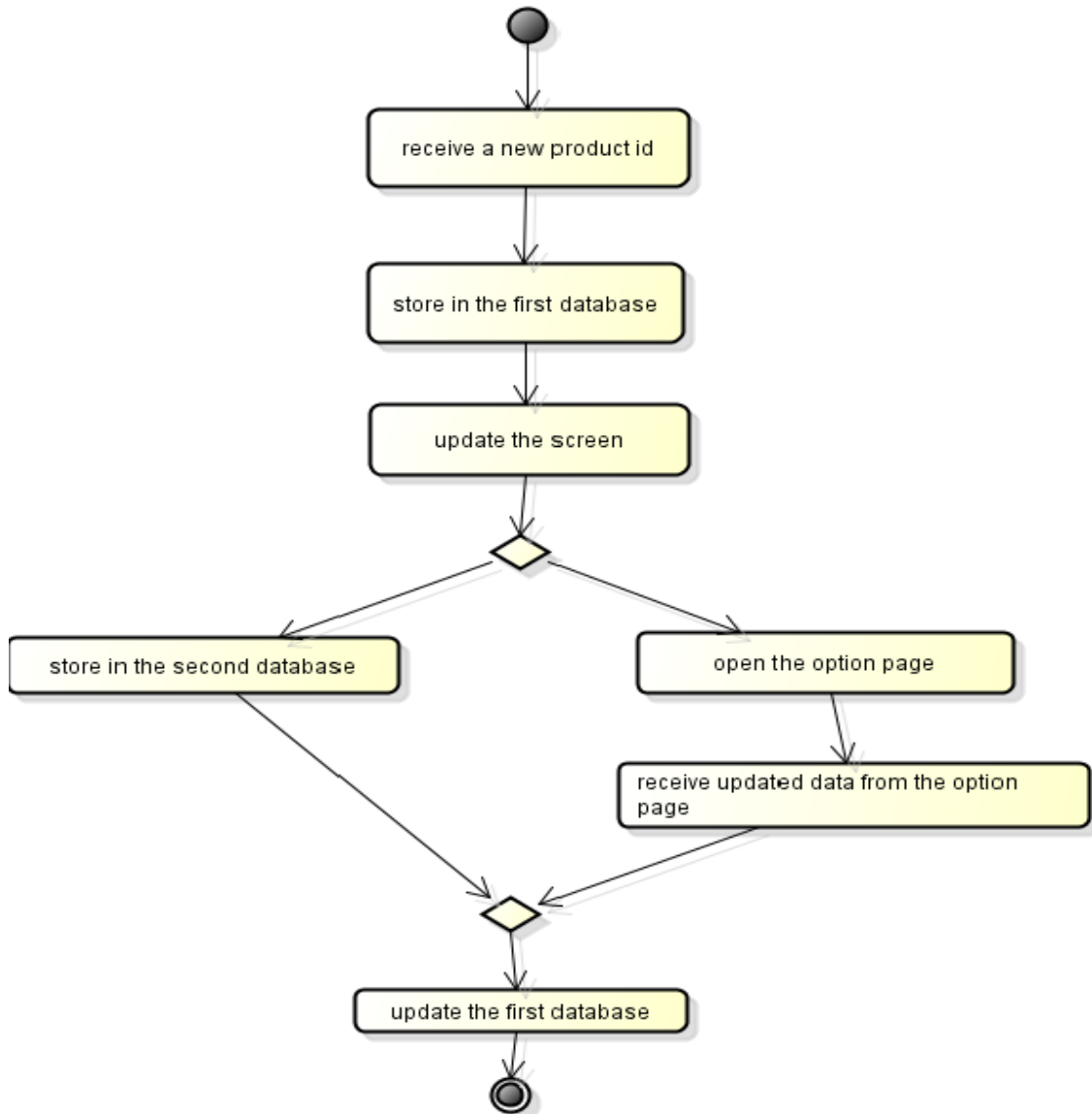


Figure 14: Adding a new product activity diagram

Figure 14 displays the activity of the system when adding a new product. Firstly, when the new product is put near the first antenna, its information will be added to the system database. After that, the product id will be shown on the screen. Two situations includ-

ing "product inside the fridge" and "product outside the fridge" occur. If the product is still placed near the first antenna and located outside the fridge, the system automatically opens the "option" page where the system can receive the updated data from a user. If the product is put inside the fridge, the product id is stored in the second database. After that, the system will update the first database in which the status of the product is changed such as "inside the fridge", "outside the fridge".

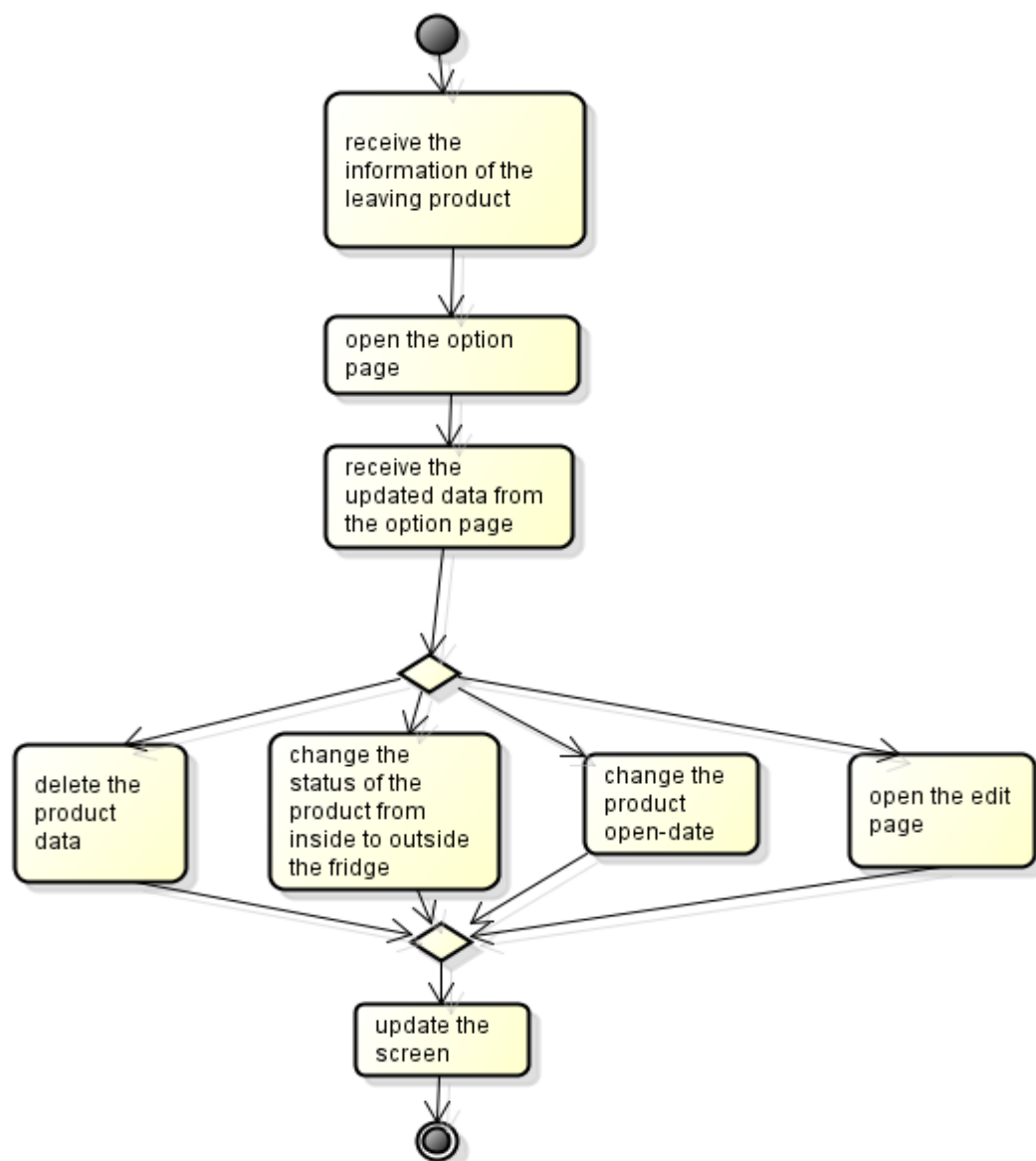


Figure 15: Taking a product away activity diagram

Figure 15 demonstrates the process of taking a product away. When a user takes a product out of the fridge, the system will open the "option" page comprising four

options such as "remove product", "take product", "open product", and "update product". When the system receives the data chosen by the user from options, the system decides which direction it must carry on depending on the submitted data. If the user chooses the "remove product" option, the system will delete the product data in the database. If the user decides the "take product" option, the system just changes the status of the product from "inside the fridge" into "outside the fridge". If the user determines to open the product, the system will set the status of the product to be opened and records the opened date of the product. If the user chooses the "update product" option, the system will open the "edit" page. Finally, the system updates the screen.

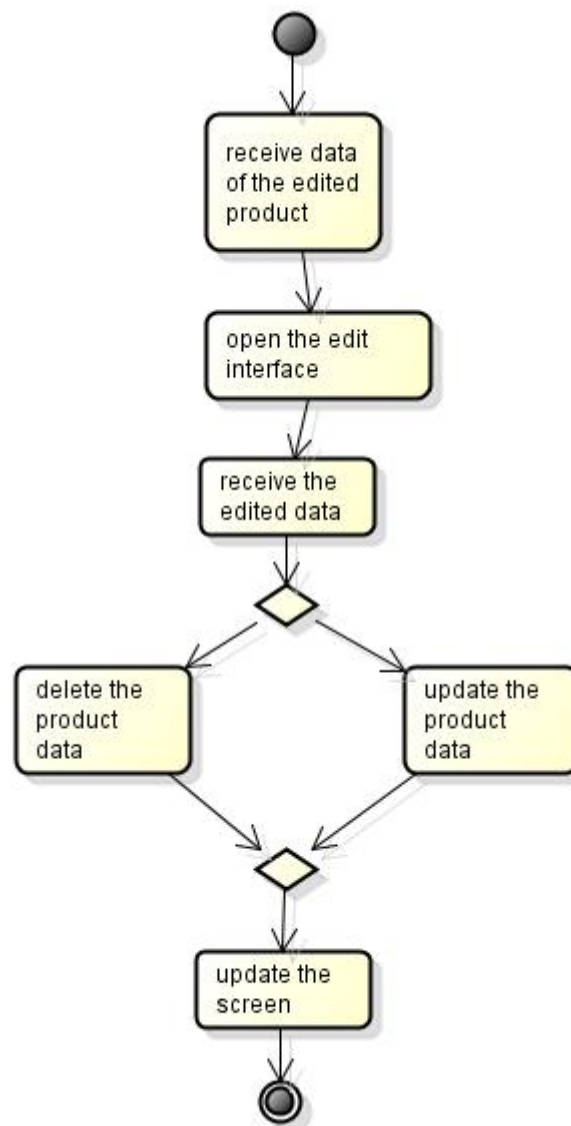


Figure 16: Editing a product activity diagram

Figure 16 represents activities of the system when editing a product. First of all, the system receives the data ("tag_id") submitted by a user. Based on the product id, the system can load the "edit" page with the product information. Next, the system receives the updated information taken from the "editing" interface. Depending on the content of the updated information, the system decides to delete or edit the product information. Finally, the system updates the information on the screen.

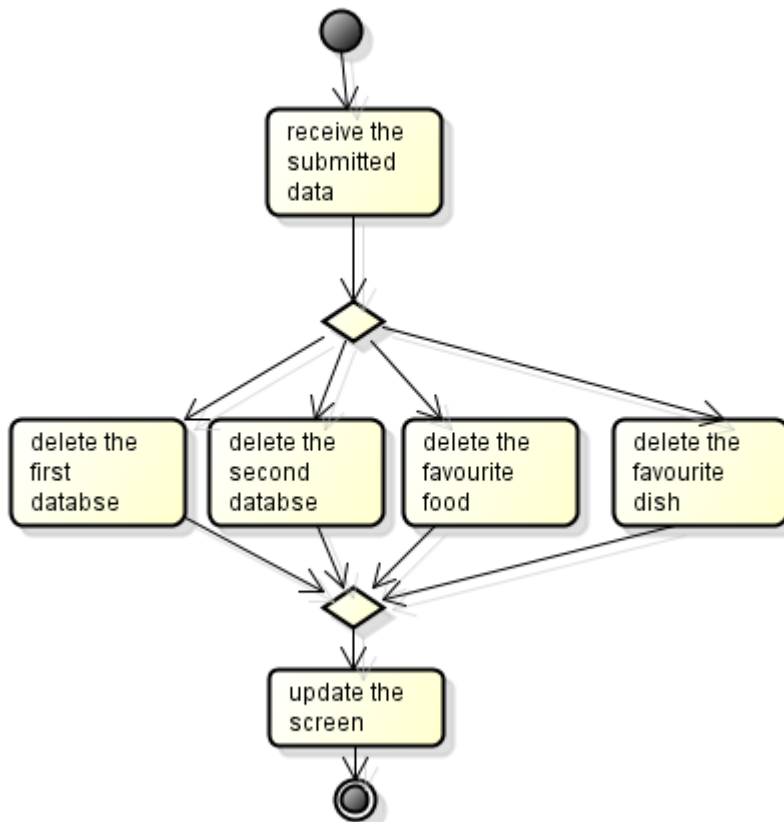


Figure 17: Purging database activity diagram.

Figure 17 displays activities when purging databases information. When the first antenna reads the product id, it stores the product information. After that, if the product is put inside the fridge, the product information is stored in the second database. When the favourite dish is added, its information is stored in the "favourite dish" database. Therefore, when receiving the submitted data from the user, the system decides which database is used and emptied. Finally, the system updates the new information on the screen.

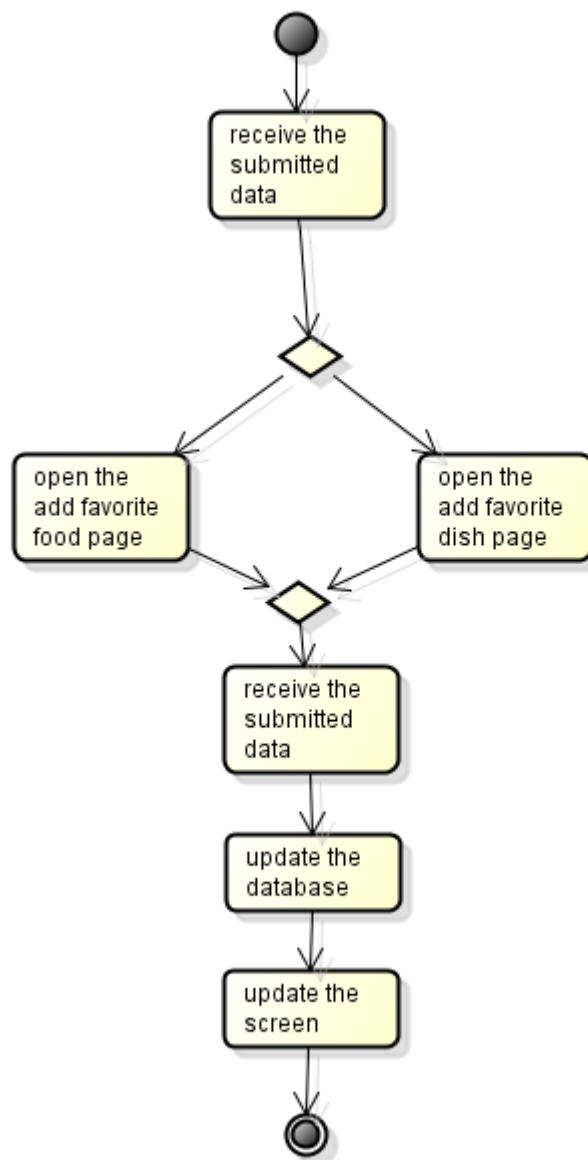


Figure 18: Adding a favourite item activity diagram

Figure 18 shows activities in the system when adding favourite food or adding a favourite dish. When the system receives the submitted information from a user, it decides which page should be opened. After that, the system will get the new data submitted by the user in the corresponding interface and it updates the database and shows the data on the screen.

4 Implementation of a smart fridge

The smart fridge was built with many stages including setting up and testing physical connections, creating the RFID reader profile, software designing, setting up the software, constructing the final end user application for connecting between physical components and the software, testing and debugging, and setting up the whole system in the RFID lab show room. All stages in the system were important and constructed step by step because some stages could not be continuously built and tested if the previous stage did not work.

4.1 Setting up and testing the physical connection

The smart fridge system worked based on "tag_id" which was read at antennas and later sent to the reader and the computer; therefore, the connection between the Sirit 510 reader with antennas and the computer played an important role in the system. If the connection was successful, the computer could get the tag information used for recording a new product. Setting up this connection was a challenge because the Sirit 510 reader is only used in industries or in large companies. Connections between the reader and two antennas, the reader and the computer were described in figure 3 in section 2.3.

After the hardware was connected, it had to be tested by using the Putty application to open the channel and apply some commands for reading the tag information and receiving the tag information. Two ways for testing the connection between the computer and the reader were a LAN connection and a serial connection.

Serial connection

At first, the reader was tested by using a serial connection because it was more convenient and easier to set up. If reader is tested by using a LAN connection, the IPv4 address of the reader must be known in advance. The information needed for the serial connection was serial line, speed (baud rate), data bits, stop bits, parity, and flow control. The project used "COM 1", "115200" in speed, "8" in data bits, "1" in stop bits, "none" in parity and "none" in flow control. These values were taken from the

Sirit 510 reader manual. The Putty interface used for the serial connection is shown in figure 19.

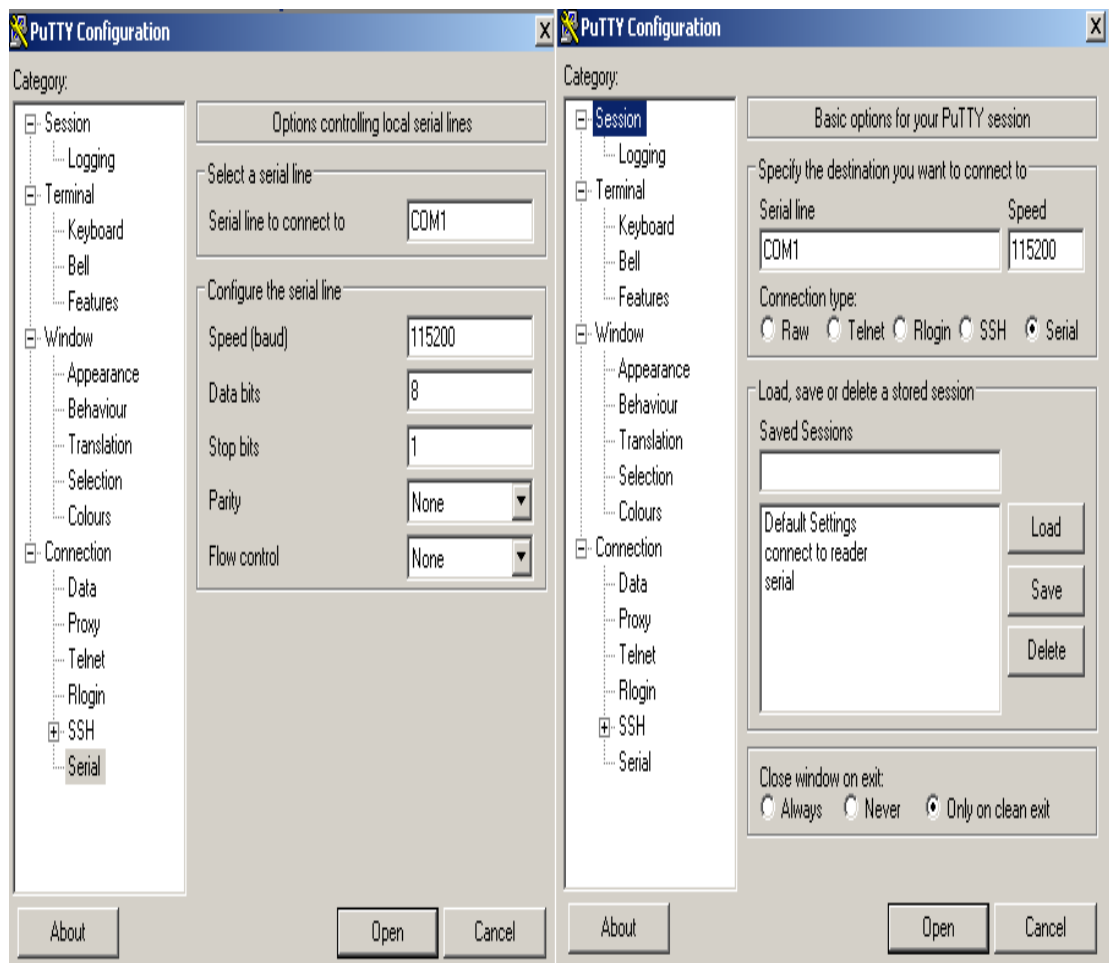
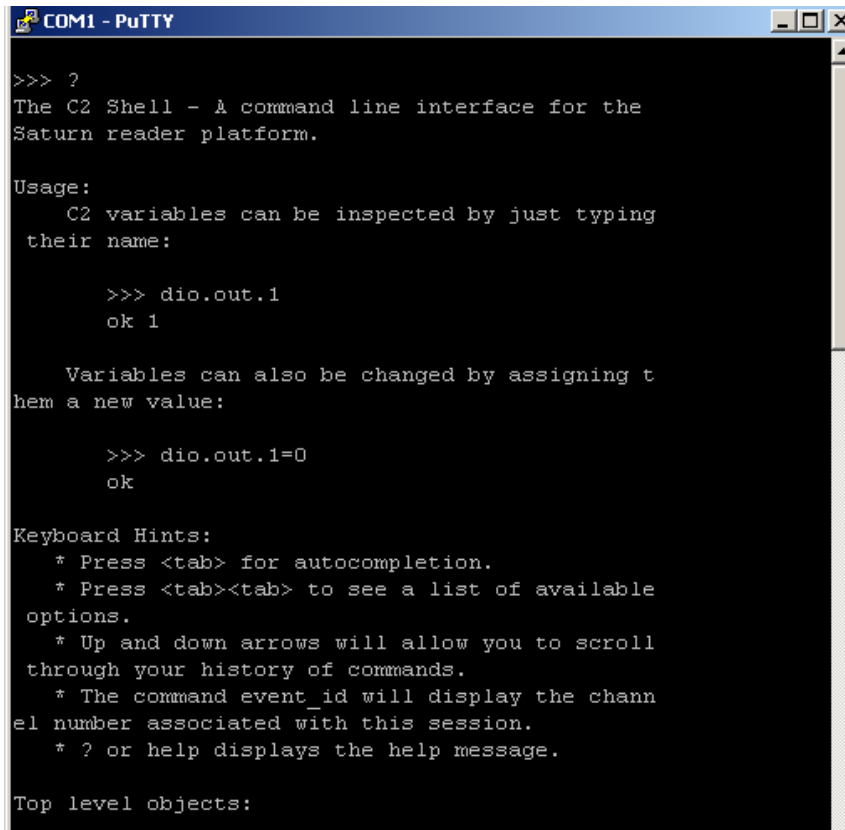


Figure 19: Setting up the serial connection in Putty

As can be seen in figure 19, the necessary data was filled in the Putty. When opening the connection, one new interface which was used for setting commands was shown. The easiest way for testing the reader was using “?” in the help mode. If the reader responded with some instructions such as texts in figure 20, it meant that the reader was working.



```

COM1 - PuTTY
>>> ?
The C2 Shell - A command line interface for the
Saturn reader platform.

Usage:
  C2 variables can be inspected by just typing
  their name:

  >>> dio.out.1
  ok 1

  Variables can also be changed by assigning t
  hem a new value:

  >>> dio.out.1=0
  ok

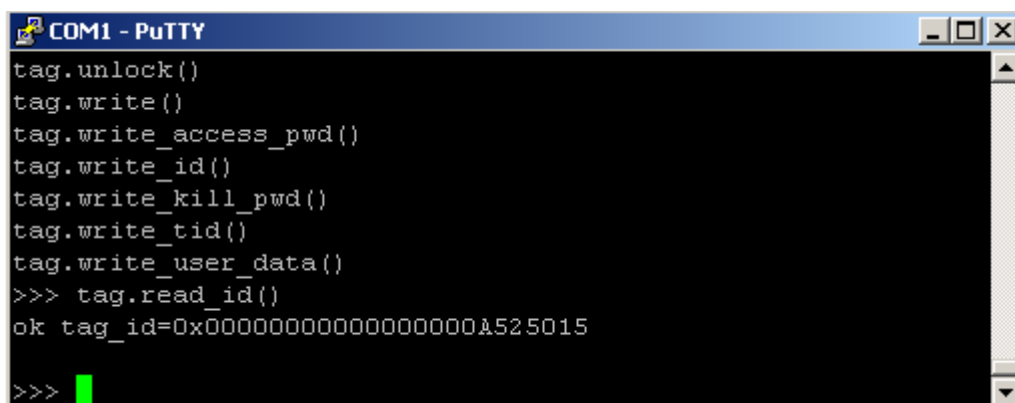
Keyboard Hints:
  * Press <tab> for autocompletion.
  * Press <tab><tab> to see a list of available
  options.
  * Up and down arrows will allow you to scroll
  through your history of commands.
  * The command event_id will display the chann
  el number associated with this session.
  * ? or help displays the help message.

Top level objects:

```

Figure 20: Responses from the help mode of the reader

The next step was testing the connection between the reader and antennas by sending “tag_read_id()” command which was unable to work if there was no RFID tag near antennas. When the reader responded with texts shown in figure 21, it demonstrated that antennas worked.



```

COM1 - PuTTY
tag.unlock()
tag.write()
tag.write_access_pwd()
tag.write_id()
tag.write_kill_pwd()
tag.write_tid()
tag.write_user_data()
>>> tag.read_id()
ok tag_id=0x0000000000000000A525015

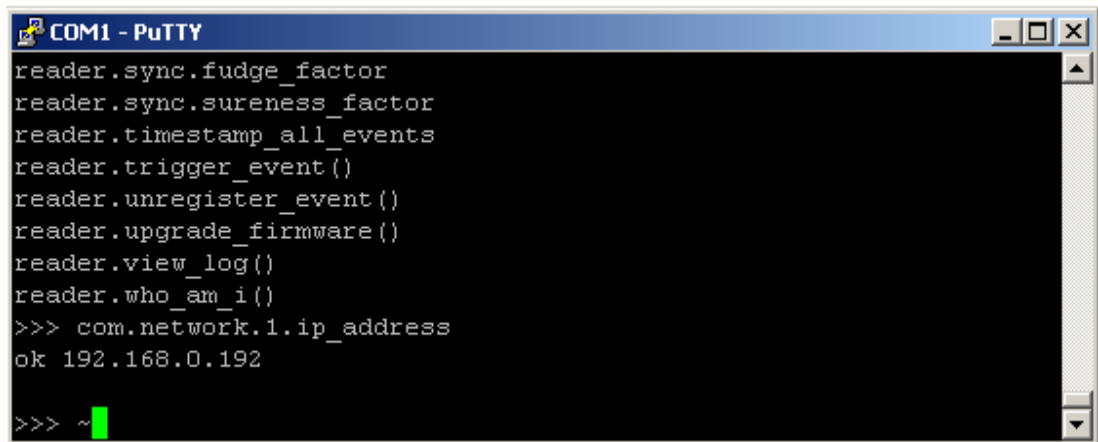
>>> █

```

Figure 21: Responses from the reader when sending command “tag_read_id()”

The “Tag_id” shown in figure 21, was unique. It was easy to check an IPv4 address of the reader by using the “com.network.1.ip_address” command. If the reader re-

sponded with string “ok” and an IP address, it meant that reader was able to work with a LAN connection. In figure 22, the reader IP address is “192.168.0.192”.



```
COM1 - PuTTY
reader.sync.fudge_factor
reader.sync.sureness_factor
reader.timestamp_all_events
reader.trigger_event()
reader.unregister_event()
reader.upgrade_firmware()
reader.view_log()
reader.who_am_i()
>>> com.network.1.ip_address
ok 192.168.0.192
>>> ~
```

Figure 22: Reader’s responses from sending “com.network.1.ip_address”

Figure 22 shows the IP address of the reader and other information of the reader such as a firmware, an event, and a view log. In order to see details of the firmware, the event and the view log, sets of firmware, event and view log commands had to be used.

TCP/IP connection

Another way to connect the reader with the computer was using an IPv4 address in case the serial connect could not work. Disadvantage of the LAN connection was that the IPv4 address and ports must be known in advance; however, this drawback could be solved by using the third party software such as the advance IP scanner which is free, easy-to-use and harmless to the computer. The scanner software could be used by using the “scan” button to scan all equipment working in the same LAN with the computer of the user. After scanning, the software returned an IPv4 address, a MAC address and other information relating to a LAN. Through the reader MAC address, the reader IPv4 address, which was used for the TCP/IP connection, was recognized. When the TCP/IP connection was successful, one command line interface, shown in figure 23, opened.

```

192.168.0.192 - PuTTY
Using username "cliuser".
event.connection id = 19
>>> Caught exception in event processing thread.

```

Figure 23: Connection to reader through an IPv4 address

Figure 23 shows that the connection between the reader and the computer was set up successfully. The reader gave "cliuser", a default username, which has the same privilege as a guess user. In order to manage the reader, the user needs to use command to change a guess account into an admin account because all configuration values are unchangeable under a guess account. The process of checking "tag_id" could be carried out in the same way as the process of checking "tag_id" in the serial connection.

4.2 Creating a reader profile

Creating a reader profile was important because it stored the reader's configuration values. Depending on user's purposes, profiles can have the same or different configuration values. Two methods of creating the profile are using the firmware interface and using CLI commands requiring knowledge of the reader's commands. The reader uses a profile name convention; therefore, when creating a new profile, a user needs to follow the instruction of the reader in the Sirit reference guide showing that the name profile must consist of a character "A-Z", "a-z", "0-9", "-" or "_". Commands for creating and checking the current active profile are "reader.profile.save(name_of_profile)" and "reader.profile.active" respectively. By using the firmware interface, the user chooses "manage profile" in the basic configuration in order to create a new profile, which is shown in figure 24.



Figure 24: Creating a new profile by using the web interface

Figure 24 shows the reader interface used to create a new profile. Furthermore, the reader allows users to delete the current active profile and restore the factory profile by sending commands in the CLI or using the firmware interface. Although the reader stores many profiles, only one profile, which can be activated by using the “activate” button in the firmware interface or using the “reader.profile.save” command in the CLI, is used at a specific time.

In the project, the profile “tw” having all suitable configuration values was created by using the firmware interface. Antennas, protocols and a communication were set up for verifying that the connection was successful and the “tag_id” could be read; so all values for these fields were identical to default values. Later, the configuration values of antennas, protocols and a communication were edited for harmonizing with the final application when the project was shown at the RFID lab show rom.

4.3 Software design

Software design, which is usually represented by UML diagrams, was the important stage in constructing the smart fridge system. Based on UML diagrams, the final application was constructed more easily. All diagrams were created by the Astah profes-

sional tool, which is a commercial product providing many useful features, such as easy-to-use, friendly user interface, and many programming languages, and system compatibility. "use case" diagrams defining functionalities which a user can manipulate when he/she uses the system were firstly created. Main components in the "use case" diagram are actors with direct interactions to the system, actions which actors do and the system. One user can do many actions to the system and one action could be done by many users. Six "use case" diagrams including "adding product use case", "taking away product use case", "editing product use case", "purging database use case", "adding favourite food use case", and "adding favourite dishes use case", were created in the project. All details of specific cases were described in section 3.1. After finishing "use case" diagrams, activity diagrams describing activities which the system reacts to when a user acts one specific action must be created. The activity diagram includes the starting activity, actions and the ending activity. Five activity diagrams comprising "adding product activity", "taking away activity", "editing activity", "adding favourite food and dish activity" and "purging database activity" were constructed. All details of these activity diagrams were explained in section 3.2.

4.4 Software preparation

The smart fridge project used the PHP programming language and MySQL for the final application so Eclipse and Xampp were installed in C drive. Eclipse, which is open-source software, helps programmers construct applications more easily by providing a ready-made platform comprised of a framework, tools, runtime for building and deploying. In the project, the Eclipse portable version, which supports the PHP programming language, was used for simplifying the project complexity. In order to change some values of Xampp configuration, the "http.conf" file, which was located at "C:\xampp\apache\conf", had to be edited. In the "http.conf" file, the line starting with "DocumentRoot" had to be modified as "DocumentRoot "C:/Documents and Settings/RFIDLAB/My Documents/workspace"" showing the workspace location which Eclipse used as the main workspace to store the whole project. Apache in Xampp was tested by using a Firefox browser with "localhost" string in the URL and "index.html" created in the workspace to ensure the successful configuring. Server services including the Apache service had to be ac-

tivated. Two ways of turning on server services include using the Xampp control panel and using commands in the CLI. In the project, the "Start apache_start" command was applied in the CLI. If the browser responded with the content written in the "index" file, the Apache and the workspace were ready to use. Otherwise, all steps of the Apache configuring must be carried out again. In order to make the final application interface more flexible, JQuery, which is a ready-made and free JavaScript library, was used. The JQuery library could be downloaded from the official JQuery website and directly added to the project in the workspace. The project could not work without the database because of its importance and useful features. The database, which is MySQL server, is included in Xampp by default, so the database could be directly used without the consideration of configuring. MySQL services could be turned on by using the Xampp control panel or using the "Start mysql_start" command in the CLI. After the database "rfid" was created by the "CREATE DATABASE rfid" command, the "SHOWS DATABASES" command was used to test an existence of the "rfid" database. The "rfid" database was shown on the screen if all commands were successful. Many tables including "add_new_product", "favourite_food", "rfid_store" were created by the "CREATE TABLE" command in the database in order to store the project data. [17]

4.5 End-user application

The final application written in HTML, PHP, MySQL, CSS and JavaScript was one of the most important stages in constructing the smart fridge because the end-user application was shown in the web interface which a user could use directly to interact with the system. In the end-user application, many files, which had different functions, were created.

4.5.1 Building a PHP file for connecting a PC with a reader

Firstly, connecting the computer with the RFID reader through the TCP/IP connection was established by using PHP socket commands: "\$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)", "socket_connect(\$socket, \$destination_ip, \$destination_port)". The "\$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)" command including three parameters such as "AF_INET" for the IPv4 Internet, "SOCK_STREAM"

for the socket type, and "SOL_TCP" for the TCP protocol which provided full-duplex, reliable, connection-based byte streams was used to create and return socket resources. The "socket_connect(\$socket, \$destination_ip, \$destination_port)" command was used for socket connecting. When connecting successfully, other commands, which are shown in figure 25, were used for logging in and getting data from the socket.

```

$output = "reader.login(login=admin,pwd=readeradmin)"."\r\n";
if(!socket_send($socket, $output, strlen($output),0))
{
    echo("Initialisation failed: " . socket_last_error());
    socket_close($socket);
    exit();
}
$input = socket_read($socket,50008);
|

```

Figure 25: Sending commands to reader by PHP

Figure 25 represents commands which were taken from the RFID manual and were applied in the "main.php" file for retrieving data from the socket. The "socket_read" command including the socket and the connection port was used for receiving responses from the reader.

4.5.2 Running the file as script in the background

The reader responded to a user immediately when the user did some actions in the application; therefore, the "main.php" file for getting "tag_id" at antennas and updating the database had to be run as script in the background. Two ways of running files as script in background include using the PHP "exec" command or using the "cmd.exe" in Windows to run the file. However, the "cmd.exe" command was used because the main web page running infinity loop would be automatically refreshed for updating the newest information. If the PHP "exec" command had been used in main web page file, the "exec" command would have recalled. As a consequence, antennas would stop for some seconds due to refreshing time delay. The "C:\Program Files\PHP\php.exe -f "C:\Documents and Settings\RFIDLAB\My Documents\workspace\Rfid_Reader\main.php" > "C:\Documents and Set-

tings\RFIDLAB\Desktop\test.txt" command includes three parts: the location of the PHP executable file, the location of the file run in the background, and the output file for testing. It was used to run the "main.php" file which combined two main steps.

4.5.3 Building the user interface of the smart fridge

The user interface of the smart fridge was shown with two frames including the left frame for showing all products information, favourite food, dishes, and the shopping list, and the right frame for editing products by using the "<FRAME src="frame1.php">" tag, the "<FRAME src="frame2.php">" tag and the "<FRAMESET cols="50%,50%">" tag. The system main view is shown in figure 26.

The image displays two side-by-side screenshots of a web application interface titled "Welcome to intelligent fridge".

Left Frame (Product Information):

- Header: "Welcome to intelligent fridge", "Wednesday, 14.43, 12. September.2012"
- Section: "Product" with a table showing items like "bean", "cucumber", and "marshmallow".
- Section: "New product's ID" with input fields for various identifiers.
- Section: "Food List Recommendation" with a "show/hide" button.
- Table with columns "DISH" and "MISSING INGREDIENT". Rows include "incolnshiresausage", "omelette", and "sausagesandmashe dpotato", each with a "missing ingredient added to shop list" button.
- Section: "Food needed to buy" with a "show/hide" button.
- Section: "Products" with sub-sections:
 - "Missing product from favourite food list" with a table containing "cake", "califlower", "chicken".
 - "Missing ingredient to make recommended food" with a table containing "bread", "egg", "potato", "sausage".
- Buttons: "add favourite dishes", "add favourite food", "clear_ant1", "clear_ant2", "clear favourite food", "clear all missing ingrs".

Right Frame (Product Management):

- Header: "Welcome to intelligent fridge", "Wednesday, 14.43, 12. September.2012"
- Form fields: "Product:" (dropdown), "Quantity:" (input), "Unit:" (dropdown), "Expire day:" (calendar), "Open day:" (calendar), "Description:" (text area).
- Buttons: "add or remove product in product list", "submit", "delete_item", "show all items".
- Footer: "LinkedIn", "RFID-LAB FINLAND on Facebook", "RFID Lab Finland".

Figure 26: The system main view

Figure 26 describes the system main view including two frames: the left frame and the right frame. The left frame is used for showing the product information, and functions of the fridge. When a function in the left frame is activated, the system will make a process to call other pages corresponding with the function. As a consequence, the right frame is updated with corresponding pages.

Welcome to intelligent fridge

Wednesday, 14:35 , 12. September 2012

Product			
milk	bean	cucumber	marshmallow

New product's ID	
0x00000000000000000000A5250	0x300833B2DDD906C000000000
0x3034994608000040000000008	

Food List Recommendation show/hide

DISH	MISSING INGREDIENT
Lincolnshiresausage	missing ingredient added to shop list
Omelette	missing ingredient added to shop list
Sausagesandmashedpotato	missing ingredient added to shop list

Food needed to buy show/hide

Products				
Missing product from favourite food list				
beer	cake	califlower	chicken	
Missing ingredient to make recommended food				
pork	bread	egg	potato	sausage

add favourite dishes
add favourite food

clear_ant1
clear_ant2
clear favourite food
clear all missing ingr

Figure 27: Overall view of the fridge.

Figure 27 represents the left frame with the product id, the product name, the shopping list and the current time which was built in the "time.php" file including the "date_default_timezone_set('Europe/Helsinki')" function for setting the Helsinki time zone and the "getdate()" function for returning an array storing the current time. The part of the "time.php" file is shown in listing 28.

```

<?php
    date_default_timezone_set('Europe/Helsinki');
    echo "<div align='center'>
    <h2> Welcome to smart fridge \n </h2>";
    echo '<br>';
    $today=getdate();
    echo $today['weekday'].",
    ".$today['hours'].":".$today['minutes']. '
        . ' ,
    '.$today['mday']. '.'.$today['month']. '.'.$today['year']
    ;
    echo '</div>';
?>

```

Listing 28: PHP code in the "time.php" file

Listing 28 shows the method of setting the time zone and retrieving the current time including current minute, hour, day, month, and year. The "time.php" file was added to the frame by the "include('time.php')" method.

The "Food list recommendation" table and the "food needed to buy" list have "show/hide" buttons, which were used for showing or hiding the content of the "food list recommendation" table and the "food needed to buy" list when buttons were activated. Buttons operation-handled system was built by the JavaScript function described in listing 29.

```

<script type="text/javascript">
    function show_hide(id) {
        var v = document.getElementById(id);
        if(v.style.display == 'block')
            v.style.display = 'none';
        else
            v.style.display = 'block';
    }
</script>

```

Listing 29: The "Show_hide" function in JavaScript

Listing 29 represents how the "show_hide" function was written in JavaScript. The button function included an invisible mode and a visible mode. The "Food list recommendation" table had buttons for adding the missing products to the "food needed to buy" list. When the button was used, the smart system added necessary products by updating the database with the parameter "updating values product_name from table rfid_store" inside the MySQL function. The "Product_name" value and the "rfid_store" table can be changed by other values and tables due to specific situations.

All buttons in the bottom of the page described in figure 27 were used for purging the database information by using the "deleting from table" parameter inside the "mysql_query()" function. Depending on user's purpose, a specific button could be used. When a user uses all buttons, the whole system is restarted.

In the project, the tooltip feature, which was built by the JavaScript tooltip library, was added. When the user puts mouse on the product name, all information of the product such as "tag_id", "product's name", "expiration date" will be shown in the tooltip.

The right frame was used for showing the "editing" form, the "adding" form, or the "option" form. A specific form was loaded depending on which function was activated. The frame is shown in figure 30.

Welcome to intelligent fridge

Wednesday, 14:43 , 12.September.2012

Product:

Quantity:

Unit:

Expire day: 12 September 2012

Open day : 12 September 2012

Description:

[show all items](#)

Welcome to intelligent fridge

Wednesday, 14:54 , 12.September.2012

Favourite food

please add number of your favourite food

[show all items](#)

Figure 30: The right frame of the screen

Figure 30 shows the right frame which is opened when a user uses the "add favourite food" button, the "add favourite dishes" button. The top of figure

30 shows the “edit” form and some buttons which the user can use to add and update data in the database. The “edit” form is described in listing 31.

```
<form action='<?php $_SERVER['PHP_SELF']?>' method='post'>
    <input type='text' name='num' id='num' />
    <input type='submit' name='submit_num' value='submit' id='submit_num' />
</form>
```

Listing 31: A form in PHP

Listing 31 shows how to build one PHP form. The form has “action” parameter defining the address of the page which the form uses to submit data to. In this case, the action was set with the “\$_SERVER['PHP_SELF']” PHP’s global variable which would submit the information from the form to the same page. In figure 30, the button “now” and the small calendar icon are used to add date to the form. When the user clicks the icon between a date string “12 September 2012” and the button “now”, one calendar table is shown on the screen. The user can choose a specific date from the calendar table. The calendar was built in the PHP language and stored in the calendar library containing the calendar class, images and CSS. In the project, the calendar library written by “TJ @triconsole” was used and modified for being suitable with the project. The bottom of figure 30 shows other form which is used for adding a number of favourite foods shown. For example, when a user needs to add three favourite foods, he/she submits the “please add number of favourite food” form with number three filled in. As a consequence, the form shown in figure 32 extends with other fields.

Welcome to intelligent fridge

Wednesday, 14:55 , 12.September.2012

Favourite food

Please add number of your favourite food

Food name	Quantity	Unit
<input type="text"/>	<input type="text"/>	<input type="text"/>
▼		kg ▼
<input type="text"/>	<input type="text"/>	<input type="text"/>
▼		kg ▼
<input type="text"/>	<input type="text"/>	<input type="text"/>
▼		kg ▼

Figure 32: The favourite food form

In figure 32, three different types of food are shown in the form. A user could choose values from boxes or type by him/herself. All added information was updated by using the `mysql_query()` function with the `updating` parameter.

If the user drags mouse to a specific product in the `food list recommend`, the product data is shown in the tooltip. If he/she clicks a specific dish, the right frame of the screen is opened with recipes of the dish. Figure 33, which shows the page in the right frame, is an example when the user chooses Lincolnshire sausage in the `food list recommendation` table in the left frame.

Welcome to intelligent fridge

Wednesday, 14:55 , 12. September 2012

Lincolnshire sausage

ingredient :pork,bread



Procedure to make Lincolnshiresausage

Lincolnshire sausages are a distinctive variety of pork sausage developed in and associated with the English county of Lincolnshire. A widely available variety at most UK butchers and supermarkets, the sausage is commonly dominated by the herb sage, rather than the more peppery flavour balance found in other regional English sausages such as the Cumberland sausage. Other herbs such as parsley and thyme are often used, although these are not authentically considered Lincolnshire sausages. Lincolnshire sausages are also characterised by their open, chunky texture the result of the constituent pork being coarsely ground rather than minced. Lincolnshire sausage: [Download video from this page](#)  

pork mixed with binders, seasonings and preservative. Traditionally the dominant seasoning flavour has always been that of the herb sage, but some recipes include other herbs, such as parsley or thyme, and flavourings such as onion. Efforts to standardise and control the manufacture of Lincolnshire sausages have resulted in a proposed ingredients list to which future manufacturers of Lincolnshire sausages may have to adhere

Lincolnshire sausage

Figure 33: A recipe of a dish

Figure 33 shows a recipe of Lincolnshire sausage. Recipes are different depending on which dish a user wants to choose. Besides, there was one "search for other recipes and directions" button for searching other recipes for the same dish. If the user clicks the "Search for other recipes and directions" button, the system will load the other page from the other cooking site which has hundreds of recipes for making dishes. Besides, the right frame can load the "checkTag.php" file which has a form including four options: "open product", "remove", "up-

date”, “item waiting” when a product comes to antennas. These options, which were built in PHP and MySQL in order to update the status of the product, are shown in listing 34.

```
<form>
    <input type="radio" name="choose" value="open product" /> open product<br />
    <input type="radio" name="choose" value="remove" /> remove <br />
    <input type="radio" name="choose" value="update" /> update<br />
    <input type="radio" name="choose" value="item waiting" /> item waiting <br />
</form>
```

Listing 34: A option form in PHP

Listing 34 describes a radio form in PHP. The function “mysql_query()” with different parameters such as “update”, “delete” was used for updating new data . Next, CSS, which helps programmer in adding, for example, colors, dimensions, fonts, loading image, and modifying visibility, was created and added. All templates, fonts, images, ideas provided by RFID Lab Finland were added to CSS.

4.6 Testing and locating the system

Testing and debugging were carried out in order to ensure that the system could work in a stable way. Every stage of the system was tested after constructing; however, when combining all stages together, some errors might occur. Firstly, all data in the database had to be purged by activating purging data buttons which were mentioned in section 4.5.3. One example product had to be used at the antenna which was located far away from the fridge for ensuring that the system could read the data from the new product. If the product information was immediately shown at the user interface, the reader and its antenna worked in real time. Next, the product had to be put inside the fridge in order to test the antenna which was attached to the fridge. After that, the product inside the fridge was taken out to the antenna for checking the “checkTag.php” page, antennas and the database. All buttons had to be used and

all forms had to be filled and submitted for testing. After testing, the antenna configuration had to be modified. In the project, all calculations were done by a specialist of Electrica OY because the smart fridge project had to be in harmony with other projects which used the same RFID technology.

Finally, the process of setting the system location in the RFID lab show room was carried out. The project needed two antennas: one antenna had to be attached to the fridge and the other antenna had to be located about two to three meters far from the fridge. The screen, which is a monitor or a touchscreen, was located near the fridge. A location of the reader and its antennas can be changed due to user's wishes.

5 Results and discussion

5.1 Problems and solutions

The smart fridge project was built during five months with many difficulties. In some stages, the project did not run smoothly. The project was about to be eliminated due to the long working hours spent for the project and its difficulties. However, the project was successfully finished with the help of some professionals from RFID lab Finland and Electrica OY. Finally, the project fulfilled all the requirements from RFID lab Finland and the project has been on show in the RFID lab show room.

First of all, exploring the reader firmware was challenging. In the project, the reader used the obsolete firmware version, so all information written in the Sirit Infinity 510 reader guide was unsuitable. The interface shown in the guide was different from one shown in the computer when the reader firmware was accessed in a browser. Upgrading the firmware was carried out for solving the problem. However, after upgrading, the project still had some unexpected issues among the reader, its firmware and manual. Although the firmware and the manual were got from Sirit, the reader could not support some operation modes and commands in the manual. The idea of replacing another reader was considered but it was difficult to purchase a new RFID reader and its antennas. Therefore, the project was tested again with other antennas and new computers. After testing, it was found that a reason causing problems was compatibility between the reader and its firmware. For example, the "autonomous" mode and the "polled" mode were supported, while the "active" mode which combined the "polled" mode and "standby" mode could not be applied to the reader.

It was easy to let antennas to work generally; nonetheless, it was a challenge to force antennas to work in specific situations because the configuration of antennas depends on conducted power, attenuation, cable loss, gain, gain units, and computed conducted power. Many mathematic combinations were calculated to figure out the frequency of antennas; however the project did not work smoothly because all formula used in calculations still had some small errors. In order to solve this problem, the project was tested with different values; as the result, the suitable configuration values were found.

The project was later put into a server; however, it was firstly built as a demo, so Xampp was used. In order to use Apache in Xampp, some configuration values in the "http.conf" file had to be modified. For example, some configuration values in the "DocumentRoot" line in the "http.conf" file had to be modified. The line "DocumentRoot" is used to specify the workspace location. Apache has many configuration features such as security, and a virtual host which can be modified due to the user's purposes. However, the project just used the Apache server without any security concerns because the project was a demo located in the RFID lab show room. Later, when the application is uploaded into a server for online using, security issues will be considered. All problems with Apache in Xampp might be disappear but replaced with other problems happening in the server. Another difficulty when dealing with the Apache server was error reporting. Apache offers the error and warning reporting, which is useful for checking errors but in some cases, it is annoying. In the project, the error and warning reporting was set in the default mode; therefore, the reporting was always unintentionally shown on the main screen. This issue was solved by modifying the Apache configuration file or by applying some PHP commands. The method of using the PHP `ini_set('display_errors', 1)` command was used for solving the error reporting issue.

The system reacted in real time; so every time when some actions were carried out, the web page had to be reloaded for updating the new information. As a consequence, users had to wait until the whole page was loaded. In order to solve the problem of the loading time, the method of running the file in the background was used. Running the file in the background was mentioned in section 4.5.2.

When building the end-user application, some difficulties occurred because the application was complicated and written from many programming languages. First of all, there

were differences in the time format between PHP and MySQL. In the project, the data was created by PHP and stored in MySQL; however, when the data was called from MySQL, it showed some strange characters. In order to solve this problem, converting methods were created. The `mysqldate = date('Y-m-d', $phpdate)` method was used to convert dates from PHP to MySQL and the `strtotime($mysqldate)` method was for converting dates from MySQL to PHP. After that, the `mysql_query("$sql_command")` function had to be considered because this function returned resources of the database. For getting the data from resources, the function `mysql_fetch_array()` had to be used in the PHP "while" loop .

The smart fridge had to be updated immediately when a new tag was coming, which caused some difficulties. At first, the `<meta http-equiv='refresh' content='1; url=http://localhost/Rfid_Reader/edit_open.php'>` tag was used for refreshing the main page in one second or redirecting to other pages in the project. The `<meta>` tag includes a character set for HTML, time in second for activating the command and the full URL. In the project, the `<meta>` tag worked but in some cases, it deactivated some JavaScript functions; therefore, the project main view was built in another way to solve the problem. The method of using two frames on the screen was constructed for making the application interface become user-friendly and dynamic. The way of creating two frames was described in session 4.5.3.

The project used many free and available libraries such as the tooltip library written in JavaScript and the calendar library written in PHP, so some difficulties occurred when applying these libraries to the project. As a result, some ready-made functions built in these libraries could not run smoothly. In order to solve this problem, some new functions were written and some ready-made functions were modified.

All problems were solved; as a consequence, all functions of the smart fridge, which were designed in advance, were successfully constructed. The smart fridge has been on show in the RFID lab show room which has about a thousand customers every year. The project was successful but many new functions of the fridge will still have to be studied in order to improve the project.

5.2 Benefits

Nowadays, many applications based on the RFID technology are used in industry, in public places or in many companies, so the RFID system used at home in daily life is unusual. However, when the project is published, it will prove that the concept of the RFID technology can be applied anywhere. Besides, the project will also prove that the RFID antenna is able to work in tough conditions such as the antenna attached to the fridge worked in the stable way at +10 C.

In the project, the RFID reader and antennas have been continuously used for five months without stopping, so it shows that the RFID reader and its antennas have good quality and stability. In reality, many companies in logistics industry use the RFID technology in their products, services 24 hours per day and 7 days per week. For example, the RFID technology is applied in key system in schools and universities in Finland. Each student has his/her own RFID key which is used for checking the authorization at every door. With the authorized privilege, the user can use his/her key to access to any room anytime. This service is run in 24 hours a day and 7 days a week.

The smart fridge is a multi-functional, easy-to-use and modern system which improves the quality of life and helps users in saving time. The smart fridge's recommendation dish function is an example of helping users save time in thinking and buying products as well as finding suitable recipes for their daily meals. Furthermore, the system protects users' health by informing expired products which are unhealthy. Besides, the system provides the adding favourite food function which helps users ensure that all necessary products are ready in the fridge.

Besides, the system is very flexible because users can choose different readers and antennas as long as they are compatible. Although each of them has different features, they still provide basic functions such as reading "tag_id", or storing "tag_id" in the database. Furthermore, users can combine the RFID reader and antennas with any fridge because the system and the fridge can work independently.

5.3 Drawbacks

Although the system was successfully built, it still had some drawbacks. Firstly, the system itself is a complicated system combining many technologies; therefore, specialists or users with the knowledge of the RFID reader and its antennas had to configure the system as well as fix problems of the system.

Besides, nowadays, passive RFID readers and antennas are mainly produced for industry and organizations, so prices of readers and antennas are expensive. For example, the RFID reader and antennas used in the project have the price of 4000 euros.

5.4 Further development

In the near future, when the project is published successfully, many companies might produce a smart fridge system. As a consequence, the system price will probably be competitive and many services will be offered to buyers who will hopefully be eager to possess one smart fridge system at home with the reasonable price.

The project was built as a demo for the RFID lab show room, so it used the database installed in host machine. It is possible to upload the project into a server which users can access through the Internet, so they can use the project and see products inside the fridge everywhere. Besides, users can use their mobile phones with the Internet connection to connect to the project, which is helpful in buying favourite products and ingredients which are needed for daily meals.

When the project is uploaded into the server, the system security could be improved by checking the user authority and categorizing groups of users. Only users with the authorized privilege can access the system and possess their favourite food. Furthermore, the system can use the server database for recording a history of activities, which will allow users to see who used the system at a specific time and what activities happened.

Checking the food calories function, which is helpful in controlling the calories of a meal, can be added to the system. Users could use this feature to categorize some specific food for each member of family depending on users' wants and their physics. For example, following doctor's instructions, a specific person might need to consume

1800 kcal per day for his/her special training. He/she could create a food list and add to the system which would calculate and return the number of total calories. Every time when he/she uses a product, the system will show how much calories he used and how much is left.

When the system is produced industrially, the current system reader and antennas could be replaced by other readers and antennas for reducing the system cost. Furthermore, the project could be combined with the smart phone technology. Some applications used in smart phones could access the system, so the user can use the system everywhere.

6 Conclusion

The main goal of the project was to build a smart fridge using the RFID technology. The project demonstrated the usefulness of the system in daily life. Users could possess a modern, smart system at home and they could easily use the system without understanding how the technologies built in the system work.

Applying the RFID technology everywhere in society is not impossible due to the success of the system. The project proved that the concept and idea of constructing a smart system combining the RFID technology and the web technology are practical. The RFID technology may be not only applied to the smart fridge but also used in other projects which relate to people's daily lives.

Due to the limitations of the project, the project, which was built as a demo version for the RFID lab show room, showed how to build a smart system using the RFID, web technology. In order to improve the project quality and usefulness, the project should be uploaded into a server which all users can access everywhere anytime. Furthermore, the project could be studied for improvement and it could be combined with the smart phone technology. Users could use their phones with a ready-made application such as Android application, IOS application or Windows phone application to access the smart fridge. Users could use the information provided by the system to buy products online through their mobile phones.

The project also showed that software design plays an important role in constructing one system from basic ideas, and that building a smart fridge was complicated. In the near future, when the RFID, web and smart phone technology are perhaps connected, applying the smart system using these technologies might bring high profits and benefits not only to producers but also to users. The smart fridge project has been on show in the RFID Lab show room which has about 1000 visitors every year.

References

1. The Basics of RFID technology [pdf]. RFID Journal; 2012
URL: <http://www.rfidjournal.com/article/view/1337/1>
Accessed 26 July 2012
2. Active RFID Asset Tracking using WiseTrack [pdf]. WiseTrack; 2010
URL: <http://www.wisetrack.com/rfid.html>
Accessed 5 August 2012
3. What is RFID? [online]. Association for Automatic Identification and Mobility; 2011
URL: http://www.aimglobal.org/technologies/rfid/what_is_rfid.asp
Accessed 10 August 2012
4. Infinity 510 Quick Start Guide [pdf]. Sirit; 2009
5. Infinity 510 Protocol reference guide [pdf]. Sirit; 2009
6. An introduction to Web site Content [online]. Phizzie Design & Productions
URL: <http://www.phizzie.com/content/an-introduction-to-content.htm>
Accessed 12 August 2012
7. Lane D, Williams HE. Web Database Application with PHP and MySQL. 2nd ed. O'Reilly; 2008
8. HTML <body> Tag [online]. W3Schools; 2012
URL: http://www.w3schools.com/tags/tag_body.asp
Accessed 14 August 2012
9. Duckett J. HTML&CSS design and build web sites. John Wiley & Sons; 2011
10. Welling L, Thomson L. MySQL Tutorial. Sams Publishing; 2007
11. The Advantages of PHP [online]. Designer's playground; 2011
URL: <http://www.designersplayground.com/pr/the-advantages-of-php/>
Accessed 18 August 2012
12. Overview of new features in Apache 2.0 [online]. The Apache software Foundation; 2012
URL: http://httpd.apache.org/docs/2.2/new_features_2_0.html
Accessed 18 August 2012
13. Goodman D, Morrison M. JavaScript Bible. 6th ed. Wiley Publishing; 2007
14. The Main Feature of MySQL [online]. MySQL; 2012
URL: <http://dev.mysql.com/doc/refman/5.0/en/features.html>
Accessed 19 August 2012
15. Chapman S. What is JavaScript? [online]; 2012
URL: <http://javascript.about.com/od/reference/p/javascript.htm>

Accessed 20 August 2012

16. Glass K M, Scouarnec Y L, Naramore E, Mailer G, Stolz J, Gerner J. Beginning PHP, Apache, MySQL Web Development. Wiley Publishing; 2004
17. What is Eclipse and Eclipse foundation[online].The Eclipse foundation; 2012
URL: <http://www.eclipse.org/org/>
Accessed 20 August 2012

