TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Engineering

RFID based lending system for the ATI library

Bachelor thesis

Student: Aleks Krause
Student ID: 113182IASB
Supervisor: Mairo Leier
Co-supervisor: Siavoosh Payandeh Azad

Tallinn
2016

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

(*date*) (*signature*)

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

RFID-l põhinev laenutussüsteem ATI raamatukogule

Bakalaureuse lõputöö

Õpilane: Aleks Krause
Õpilase ID: 113182IASB
Juhendaja: Mairo Leier
Kaasjuhendaja: Siavoosh Payandeh Azad

Tallinn
2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(*kuupäev*) (*allkiri*)

Abstract

The objective of this thesis is to use a Raspberry PI model B+ to design a lending system using a RFID reader, a LCD and a MySQL database. The lending system will be taken into use at Tallinn University of Technology's department of Computer Engineering laboratories for lending out electronic modules and scientific equipment.

The new lending system will serve as a replacement for the paper form lending system, where employees do the lending of the equipment to the students. The need for a new system came from a growing number of equipment elligible for lending and from the need of a more up to date and precise lending system.

The system's users are kept in a database. The University's employees are the lending system's administrators. The administrators can register themselves into the system. The administrators are in charge of unlocking the device for lending. Once the system has been unlocked, then students who have RFID tags can register to the system by using a web interface. If the student has succesfully registered, then the student may borrow or return already lent out equipment from the ATI laboratories.

The thesis describes the implementation of the system using the Raspberry Pi, how RFID works, how the hardware is connected and how the python code is structured. The hardware is used to get RFID and button inputs from the user for use with the LCD Python menu. The Python LCD menu is connected to a local MySQL database, which is also described in this project. Finally a web interface is created to let students add themselves into the database.

The result of this project is a functional prototye, its database and user interface, with its hardware and software design descriptions.

The thesis is in English and contains 47 pages of text, 6 chapters, 27 figures, 3 tables.

Annotatsioon

Selle lõputöö eesmärk on luua RFID põhine laenutussüsteem Tallinna Tehnikaülikoolile kasutades Raspberry Pi mudel b+, LCD ekraani ja nuppe. Laenutussüsteemi on mõeldud kasutamiseks Tallinna Tehnikaülikooli Arvutitehnika teaduskonna laborites, et laenutada välja laboris olevat varustust.

Uus laenutamissüsteem hakkab asendama eksisteerivat laenutamisprotsessi, kus varustuse laenutaja andmed kirjutatakse paberil ankeedile. Vajadus uue laenutamissüsteemi jaoks tuli seoses aina kasvava laenutatava varustuse hulgaga ning mugavama ja täpsema lahenduse sooviga.

Süsteemis olevad kasutajad on olema andmebaasis. Laenutamissüsteemi administreerivad ülikooli töötajad. Administraatorid saavad end ise süsteemi registreerida. Laenutamissüsteem on lukus ning avatav ainult administraatori RFID kaardiga. Kui süsteem on avatud, siis õpilased, kellel on RFID kiip/kaart saavad end registreerida süsteemi laenutajateks, kasutades selleks loodud veebilehekülge. Kui õpilane on edukalt end registreerinud, siis võib ta laenata või tagastada RFID kleebisega märgistatud varustust.

Lõputöö kirjeldab, kuidas Raspberry Pi-d saab kasutada ning kuidas RFID töötab. Kirjeldatakse riistvara ülesehitust. Kirjeldatakse Pythoni koodi struktuuri, mille ülesanne on LCD menüü kaudu kasutaja käsklusi täita. Samuti kirjeldatakse MySQL andmebaasi struktuuri. Kasutajate registreerimiseks loodi veebileht PHP-s. Kirjeldatakse veebilehe ülesehitust.

Selle lõputöö tulemuseks on funktsioneeriv prototüüp, koos andmebaasi ja kasutajaliidesega, ning selle riistvara ja tarkvara ehituse kirjeldusega.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 47 leheküljel, 6 peatükki, 27 joonist, 3 tabelit.

Table of abbrevations and terms

RFID	***Radio-frequency identification***
RPi	***Raspberry Pi b+***
NFC	***Near-field communication***
LCD	***Liquid Crystal Display***
TUT	***Tallinn University of Technology***
GPIO	***General Purpose Input/Output***
MSP430	***MSP-EXP430G2 launchpad***
DLP7970	***DLP7970ABP: TRF7970 RFID/NFC reader***
PICC	***Proxmity Integrated Circuit Card***
VICC	***Vicinity Integrated Circuit Card***
GIT	***Has no acronym*** *- A widely used version control system for software development.*

List of figures

List of tables

Table of contents

1. Introduction

Tallinn University of Technology lends out scientific equipment to their students. They have a database where all the equipment elligible for lending is held on record. Despite this the lending is done with paper forms. The main objective of this thesis is to extend the existing lending process with an automated “Lending System”, which uses RFID .

Another objective is to speed up the lending process because filling paper forms takes time and mistakes are more likely to happen. At the begining of the semester when students may want to lend out something within a limited amount of time is when these mistakes are very prone to happen.

The lending system is using RFID authentication. The equipment is tagged with RFID and has its information inserted into the database. Emplyees of the University would use their RFID employee cards for system administration or lending. Students would use RFID cards cards given by the faculty members or cards of the University or cards of their own for lending out equipment or returning it. Personal cards could include such as the ISIC stuednt cards.

The system is based on a Raspberry Pi model b+ (RPi) and a RFID module from Texas Instruments. Raspberry Pi is a low-cost hand-held size computer, which was first introduced to make computer science and electronics more wide-spread around the world. RPi is used for this project, since it can easily be connected to electronic circuits and has a very good interface for developing software to control anything connected to it. Together with some other electronic components the RPi together with a MySQL database will be used to build a miniature lending system.

The lending system automates lending out equipment from the Computer Engineering laboratories and will be put to work as soon the system testing period is over.

1.1 Problem statement

The problem is to disregard paper forms and start using a more precise and faster lending system. How can we use modern technology to create a lending system without it costing too much?

For this purpose there is a need to identify all equipment and identify who is borrowing or returning equipment. Also there is a need to know to whom the equipment was lent out to and when.

1.2 Organizing

This thesis is divided into 6 parts. The first part of the thesis is to make the introductions. Chapter 2, is to describe the technology that the Lending System runs on. The system is also compared to some other similar projects.

The next chapter is to get a descriptive overview of how the hardware was connected and what it is used for. This part also describes the firmware, the RFID reader, runs on.

In chapter 4, the python scripts that are made for the hardware are described. Then an overview of the SQL database created and the tables to go with it is given . After this, the way the SQL database is connected to the Python Menu is described. Lastly in this chapter, the web interface, which is created for new student lenders to have a way to register themselves as users of the Lending System, is described.

In the fifth chapter an overview of the LCD user Menu, which is put together using all the scripts created in chapter 4, is given. The Menu is then broken down into use cases with descriptions.

The final chapter contains the conclusion of the thesis and descriptions for possible future developments and further work.

2. Lending system

2.1 Why a lending system

Lending or renting books, movies, or even cars has become something people nowadays are used to doing. Some people would say that from a business's perspective it is the dream, since you don't have to do much and get paid for it. As written in the famous book "Rich dad, poor dad" by R.Kiyosaki, about how he made his fortune and teaches other people how to make theirs. Even he made his first money by lending out comic books and talks about the beauty of it because he hired someone else to do the lending for him and didn't have to do anything himself [8]. Nowadays we do not even want to pay someone to do the lending for us and want to even save from having to pay salaries. For this an automated lending system is perfect since no employees are required.

When designing a lending system, it should be as low cost as possible. This is one of the reasons why the raspberry Pi is perfect for this project. Obtainable for as low as 44.95€ from Estonian stores [16]. Combined with a cheap 16*2 LCD screen, a few pushbuttons and a RFID/NFC reader, it will be all that is needed to build a lending system. All of the components can be bought for a combined price of under 120 € .

Of course that is not the only reason for building a lending system. Once you start having too many components in your lending storage, it will become hard to keep track of where and to whom anything has been lent to. In this case using paper forms will also do the trick, but would require an employee to be present at all times. In the case of a lending system all records would be kept in a database and can easily be monitored by anyone who wants to log in to view the records. The system built in this thesis will have an automated lending system. It will also have the option for users to register as lenders by themselves, without the need of an employee for administration.

Moreover, we should examine comfort reasons. Automated returning systems are being used nowadays in most libraries. For example, Tallinn University of Technology's library uses a returning system which is operational, even when the library is not open[27]. This makes

returning much easier for the library users and employess. Meaning returning can be done around the clock.

Speed will also become a subject. The user only has to swipe his own RFID tag and the equipment's tag to lend something. And only the equipment's tag for returning. This is much faster than using paper forms for such transactions.

Finally, in many businesses RFID tags are used for security reasons. If all items kept in a shop are tagged and the business has barriers meant for RFID detection, then stealing becomes much more difficult. This will not be in the scope of our project, but is definetely something to consider for further development of this project for later.

2.2 RFID

A RFID system consists of two parts, the RFID tag and the reader [9]. The reader can also be called a writer, since it can also be used to write information to tags in some cases, but that will be beyond the scope of this project. The reader transmits a check-up signal to the tag. If the check finds a tag, then the tag powers on its microchip from its power source or gets power from the reader and sends a signal back to the reader [11].

A RFID tag is an object that can be placed on an object so that it can be identified and in some cases tracked.

The tags can be classified by their power source.

- Active tags work by using a power source. This makes them reliant on the lifespan of the battery they are running on. This is the main reason they are not used by consumers, since they would require constant care. They also cost more, which makes them more dificult to sell to wider audiences. They can however be communicated to from distances ranging up to 4miles [23].
- Passive tags do not operate on power and are thus not reliant on any form of battery. In the case of a passive tag, the tag is powered on by the reader itself, by either magnetic induction (near-field coupling) or by electromagnetic wave capture (far-field coupling) [11]. Thanks to not relying on a battery they are much smaller than active tags. Due to not having a battery however, they require a much stronger signal from the reader. Depending on the frequency that the tag uses, the range of the reader's field also

varies. The frequencies are 128kHz, 13.56MHz, 915MHz or 2.45MHz [11]. Lower frequency systems operate using magnetic induction, higher frequency systems operate using electromagnetic wavecapture [11]. The tags used in our project will be 13.56MHz passive tags.

The tag consists of a microchip and an antenna. The antenna is used to receive a signal from the reader. The microchip then activates and sends out all the information contained on it back to the antenna. The antenna then transmits the information back to the reader [11].

Near Field Communication (NFC) builds upon RFID technology by acting as a two way communications system, whereas RFID only works one way [13].

2.2.1 How passive RFID tags work

The reader used in this project is for 13.56MHz tags[20]. The firmware which will be loaded onto the reader displays the tag information and also the type of the tag which has been scanned. The types supported with the reader are ISO15693, ISO14443 type A and B. The types are written on the reader with corresponding LED-s.

All three standards have a data error rate of 0.1% and their carrier frequency is 13.56 MHz.

ISO 14443 can be grouped up into four parts[6]:

- Physical characteristics. The tag must be in an encasement the size of a credit card[29].
- Radio frequency power and signal interface. This part defines the frequency, data-rate, modulation and bit coding that the contactless chip uses[29]
- Initialization and anticollision. This part defines how the reader acts in the case of multiple cards being in the reader's field[29].
- Transmission protocol. This part defines the data format and transmission protocols between the tag and the reader[29]

For a system to comply with ISO14443 it must meet the requirement for all four parts.

ISO 14443 standard tags are split into 2 most known types [20]:

- Type A: These work by receiving bursts of power from the reader. Once in the proximity of the reader [20].
- Type B: These work by getting continuous power from the card reader. They handle bit rates of 847 kb per second.[20]

The second type of tag we will be using is a ISO15693 tag, which has a greater range than the ISO14443. Due to this the necessary magnetic field can be less than that of the ISO14443 which needs 1,5 to 7,5 A/m. The ISO15693 cards only require 150mA/m to 5 A/m, giving them a range of about 1 meter on some readers[28]

2.3 Similar systems

RFID based authentication or lending systems are nothing new in today's world, where everyone is constantly getting information from the internet. It means that such systems already exist. What are some such systems and what are their differences with the one being made in this project.

1. **An attendance system using an Arduino RFID tag reader, a raspberry Pi, a NUM lock keypad and an LCD screen** [5]. The system uses a MySQL database for storing the users needed to verify attendance. This system is well constructed and serves as a good example of how the raspberry Pi can be used to construct everyday work related necessities, such as time and attendance systems. The difference with our system is that this is an attendance system. Eventhough the hardware is similar, the purpose of the project is something different. The hardware is also similar, not identical, as this system uses a keyboard compared to our system which uses 3 pushbuttons.

2. **RFID based Library Management System** [15]. A lending system on sale for 75€ on sale on the embeddedkits website. This project works on a custom electronics board and is not based on a Raspberry Pi. The project has no case and is a raw electronics circuit. It probably does not have any means of expanding the system, since the hardware is limited and set. The difference between this system and the Lending System built in this thesis is the hardware and expanding capabilities. The system does not have an ethernet cable, meaning connecting to a remote database, would not be possible with this system.

On the other side of the scope are bigger systems, which are used all over the world for library management. The biggest of them all is Bibliotecha[2]. Bloomberg Business describes the system – "*It offers kiosks for self service; return and sorting systems; RFID security products that give information and alert on historic logs to libraries; and staff*

solutions ranging from RFID programming stations to handheld devices and intelligent shelving"[3].

Bibliotecha is a multi-million dollar global company. It is definitely much more advanced than our system, but it is also a lot more expensive, which is its main difference with our system. It has everything this system could become with further development, but would cost much more.

3. Connecting the hardware to the Raspberry PI

In this chapter we will be talking about the necessary hardware components for building our lending system and how to interface them with the RPi. The necessary hardware components are:

- Texas instruments MSP430G2 launchboard
- DLP design Inc. DLP7970 NFC/RFID boosterpack
- 3x pushbuttons
- ADM1602K-NSW-FBS/3.3V 16*2 LCD screen

Since the RPi uses General purpose input/output(GPIO) pins, then some of our hardware can be fairly easily connected to it by soldering them on to these pins. For the MSP430G2 and the DLP7970 we will be using a USB port.
The RPi is going to be enclosed in a protective case made out of plexi-glass. The pushbuttons will be inseted in to the plexi-glass for better durability. The RFID reader will be put into a small box.

3.1 Introduction of the Rasperry Pi

The Raspberry Pi is a very small device comparable to a small mobile phone in size. Even though it is quite small, it can still connect to a lot of devices without problem.

The RPi has standard home Desktop PC cable sockets. It uses an HDMI cable for connecting to a screen. It has 4 USB ports for connecting to a keyboard, mouse or an RFID reader[17]. It also implements an ethernet cable for connecting to the internet or a local network[17]. Lastly, 5 Volts of power comes from a micro-USB cable, which can easily be obtained from a phone charger per say[17].

What makes the Pi great for electronics are its General Purpose Input/Output(GPIO) pins. These pins are meant to be used for all purposes, meaning you can use any GPIO pin for interfacing a button or connecting to a screen. In total it has 40 pins, 26 of which are GPIO pins. [13].

In this project we will be interfacing the three pushbuttons and the 16*2 LCD screen with the GPIO pins.

3.1.1 The electronic board on the Raspberry

In order to connect the components to the RPi, we will be requiring an electronic board. The electronic board will be attached on top of the RPi by adding a 2 headers on top of the 40 pins of the Raspberry.The male pins of the headers will be put through our electronics board and will be soldered on to it. By doing this we can solder the LCD and the buttons easily on top of the RPi, without having to use any wires.

3.2 The LCD

The LCD which will be used is an ADM1602K-NSW-FBS/3.3V LCD display(3.15" x 1,425")[21] by sparkfun electronics.

This display utilizes a common HD44780 parallel chipset[7]. This display is meant to run on 3.3 V, although a 5V display is also available.

Since the RPi has a 3.3V pin on it, most of the wiring will be done without the need of any resistors or capacitors. The only part of the electronic circuit where a resistor will be required is between the LCD contrast pin and ground. If no resistor were to be added, the LCD would run on maximum contrast and make it almost impossible to see anything from it. For this project a 2,7kOhm resistor was used for optimal contrast.

The LCD will require a minimum of 10 pins to function correctly with the RPi, we will be using 12 with the added 2 pins for the backlight.. In the scope of this project, the LCD is interfaced in 4-bit mode. The reason for 4-bit mode is to keep the circuit as simple as possible, as opposed to 8-bit mode, which would require 4 more pins, for more info see *Table 1*.

Any GPIO pins would do for connecting to the LCD, but we will be using the ones that are situated on the left row of the two rows of GPIO pins running on the Raspberry's board. The connections are shown on *Figure **1*** and mapped on *Table 1*. This will save up space for connecting the buttons to the other row later on..
The LCD will be connected to a header, which will have its male connectors be through the board.

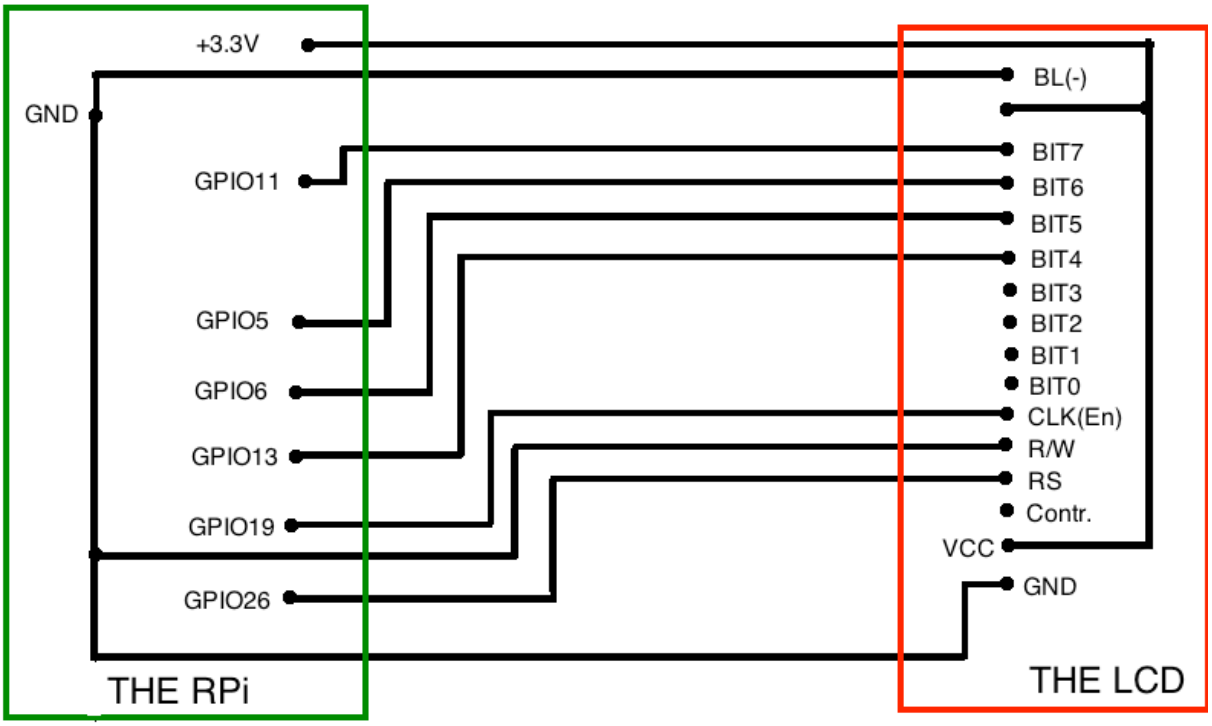

Figure 1, LCD/Raspberry wiring

Table 1, LCD – Raspberry GPIO/LCD soldering[5]

LCD PIN	**Raspberry Pi (Physical no. - GPIO no.)**
1. GND	39. Ground
2. VCC (+3.3V)	1. 3.3V
3. Contrast	->2.7kOhm resistor ->1. Ground
4. Register Select (RS)	37. GPIO 26
5. Read/Write (R/W).	39. Ground
6. Clock (Enable).	35.GPIO 19
7. Bit 0 (Not used in 4-bit operation)	-
8. Bit 1 (Not used in 4-bit operation)	-
9. Bit 2 (Not used in 4-bit operation)	-
10. Bit 3 (Not used in 4-bit operation)	-
11. Bit 4	33. GPIO13
12. Bit 5	31. GPIO 6
13. Bit 6	29. GPIO 5
14. Bit 7	23. GPIO 11
15. Backlight Anode	1. 3.3V
16. Backlight Cathode (-)	9. Ground

Once everything is done then the LCD will power on each time the raspberry is powered on. There will be no text but the backlight is getting constant power and is a good indication that the wiring is done correctly.

3.3 Connecting the pushbuttons

This part of the project describes how to connect the push buttons to the Raspberry Pi b+ .

3.3.1 The logic

In this project 3 pushbuttons will be used for interfacing the menu choices, as shown on *Figure **2***. The buttons will be yellow, white and blue. The buttons will each hold a number, which will be used for interfacing the lending system's menu.

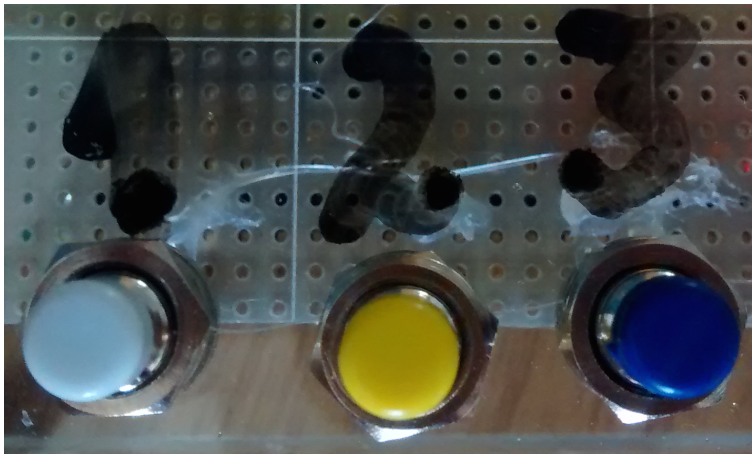

Figure 2, Three pushbuttons

Normally button connecting would be more time consuming, since resistors would be required for edge detection of the signal the psuhbutton sends. This is done by connecting pull-up or pull-down resistors [17]. These resistors are required to keep the GPIO input from going to a floating state, since it does not have any dinstinct voltage level [12]. The pin needs to tell the difference between high and low voltage, which is why the resistors are needed. Otherwise the pin would incorrectly detect states, because of electrical noise[12].

However, in the case of the Raspberry these resistors are already built in and need to be activated with software. Therefore, the pushbuttons can be connected straight to the RPi [12].

Firstly to connect the pushbuttons three GPIO pins and three power outputs (GND or 3.3V) need to be chosen. In this project GPIO 15, 8, 20 and three ground pins were used as shown

on *Table **2***. Mainly because they are conveniently on the other side of the GPIO pin lineup than the LCD pins, on the left. Also, because the chosen GPIO pins are all one pin away from ground on the RPi's GPIO layout, which is exactly the same width as the button pins are apart, meaning they can be connected straight to the pins.

Table 2, Pushbuttons

Button no.	**Rapberry pin 1**	**Raspberry pin 2**
1	Ground	GPIO 20
2	GPIO 8	Ground
3	Ground	GPIO 15

The resistor will be situated between Ground and the input pin. Since we are using Ground, all our buttons will need to be using pull-up resistors [17].

3.4 Connecting the DLP7970 NFC/RFID boosterpack to the MSP430 launchpad

3.4.1 Connecting the launchpad and the boosterpack

Hardware used in this chapter:

- MSP-EXP430G2 launchpad, meant for use with Texas Instruments boosterpacks, as shown on *Figure **3***.

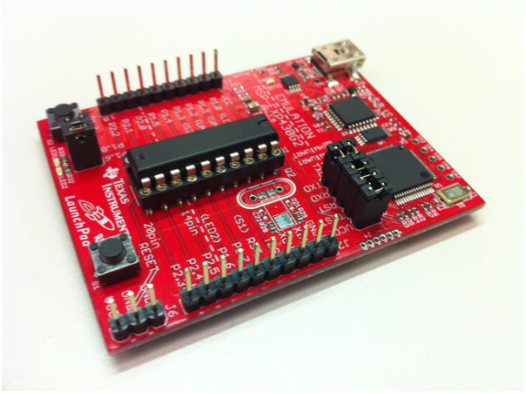

Figure 3, TI Launchpad [24]

- DLP7970ABP: TRF7970 RFID/NFC reader [25] as shown on *Figure 4*.

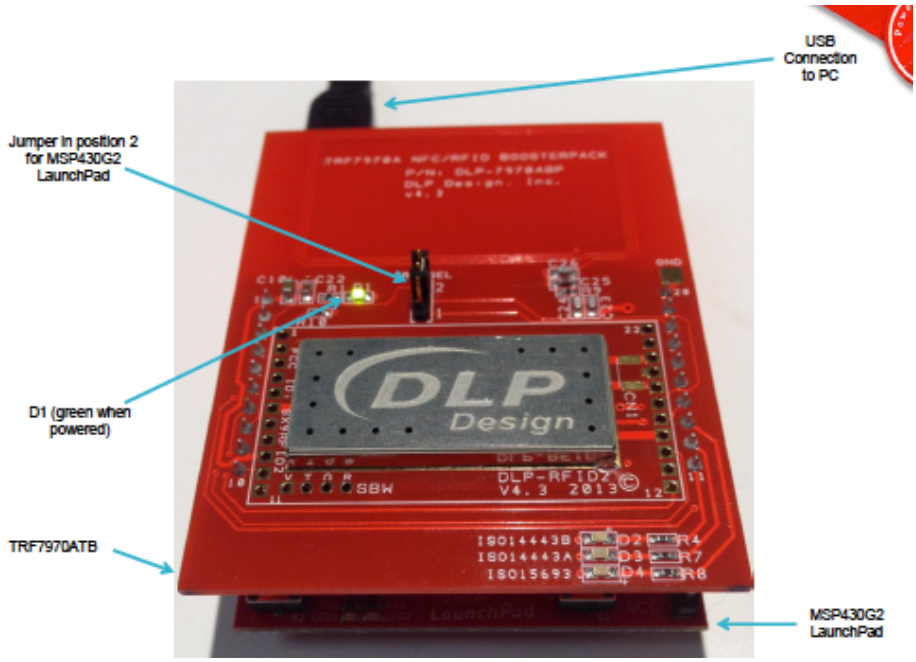

Figure 4, The RFID/NFC reader [26]

At first the RFID/NFC reader was chosen for this project. The initial plan was to use the reader by connecting it straight to the GPIO pins. It turned out that the reader only works with a launchpad, which would run the firmware required for the reader to work. The next step to getting the hardware working was obtaining the MSP-EXP430G2.

The MSP-EXP430G2 can be defined as written In the description on the Texas Instruments homepage: *"The MSP-EXP430G2 LaunchPad Development Kit is an easy-to-use microcontroller development board for the low-power and low-cost MSP430G2x MCUs. It has on-board emulation for programming and debugging and features a 14/20-pin DIP socket, on-board buttons and LEDs & BoosterPack Plug-in Module pinouts that support a wide range of modules for added functionality such as wireless, displays & more*"[24].

The DLP-7970ABP is an add-on board designed to fit TI's MCU LaunchPads that incorporates DLP Design's DLP-RFID2 RFID module.
Neither of the two components can operate without the right firmware, so the launchboard will need to be interfaced to know what to do with the information coming from the boosterpack. The MSP430 needs to know which hardware mode it needs to use. This needs to be defined in the hardware by putting the jumpers on top of the launchpad in the correct position.

The DLP7970 has a jumper on it, which needs to be put to position 2 for the firmware to work, as shown on *Figure 4*. Once the launchpad and boosterpack are connected, then the launchpad has a standard micro USB to USB cable with it, which needs to be connected to a Windows PC in order to upload the right firmware onto it. If all is connected correctly, then a green light will turn on on the DLP7970[26] , as shown on *Figure 4*.

3.4.2 Uploading the firmware to the launchpad

The MSP430 is a multi-purpose launchpad, which is now connected to the DLP7970. For the multifunctional MSP430 to know how to communicate to the boosterpack, it will require the correct firmware. The firmware can be found on the Texas Instruments web page[21] and will be available with the full instructions on the DLP7970 on the extra CD containing all the files used in this project. A windows PC is required to upload the firmware since the flashing process uses a program called UniFlash (Windows) to upload the firmware to the MSP430.

3.4.3 Launchpad Firmware

Once the correct firmware has been uploaded to the launchboard, it will begin transmitting RFID information to the virtualCOM port it is connected to if any ISO 15693, 14443 A or B RFID tags are present in the reader's field. Any terminal can connect to the Boosterpack. The only thing that needs to be made sure, is that the following settings are correct:

- Send/receive on the correct communication channel
- Baud Rate: 9600
- Parity: None
- Data Bits: 8
- Stop Bits: 1
- Flow control: Hardware

A corresponding LED will light up in the case of each type of tag, as shown on *Table 3*.

Table 3, RFID reader LED colours

DLP7970 LED	Tag
Blue LED	ISO 14443 type A
Purple LED	ISO 14443 type B
Green LED	ISO 15693

Once the launchboard and boosterpack are ready, they may be connected to one of the Raspberry Pi's USB ports.

The firmware defines how the two components will work as one. The DLP7970 is connected to the MSP430 launchpad using SPI(serial), using a hardware Universal serial communication interface(USCI)[26]. The launchpad is the master device and initiates all communication with the reader.

The anti-collision prodedures (as described in the ISO standards 14443A/B and 15693) are implemented in the microcontroller firmware to help the reader detect and communicate with one proximity integrated circuit card (PICC) or vicinity integrated circuit card (VICC) among several PICCs/VICCs.

The information it will be sending in the case of one ISO 14443 type A tag, will be a tag on one line and the number of tags found on the other , as shown on *Figure 5*.

```
ISO 14443 type A [tag id, RSSI]

tags found: 1
```

Figure 5, 1 tag present

The tag information sent by the reader will be kept in a cache, which only empties when the reader transmits information.

3.4.4 The RFID tags

Employees of the TUT use RFID cards for the door security system in the University. The system works by using the same cards for the administrators. The equipment elligible for lending will have a sticker with a RFID tag put onto it. The students can use their own RFID cards for using the Lending System. Such cards are Tallinn public transport green cards and student ISIC cards.

3.5 The hardware architecture

The hardware architecture is shown on *Figure **6***. The connected pushbuttons and RFID reader will serve as user inputs. The Raspberry Pi will handle all the processing of the information, given to it by the user, with software. For the time of this thesis, it will be using

the information given to it, to either get or write information to its local storage. The information will then, once more be processed and outputted to the LCD for the user.

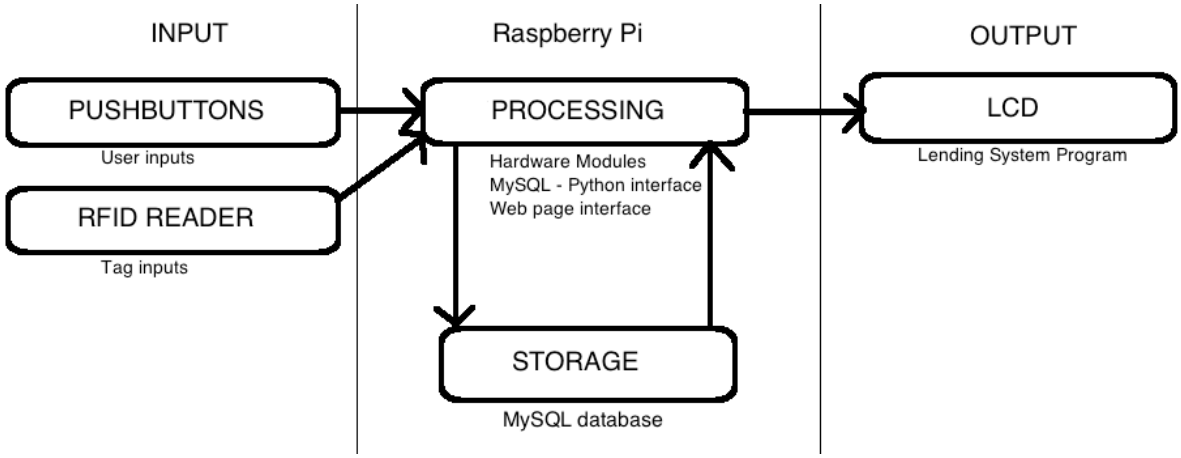

Figure 6, The hardware architecture

The GPIO connections for the whole system are shown on *Figure 7*.

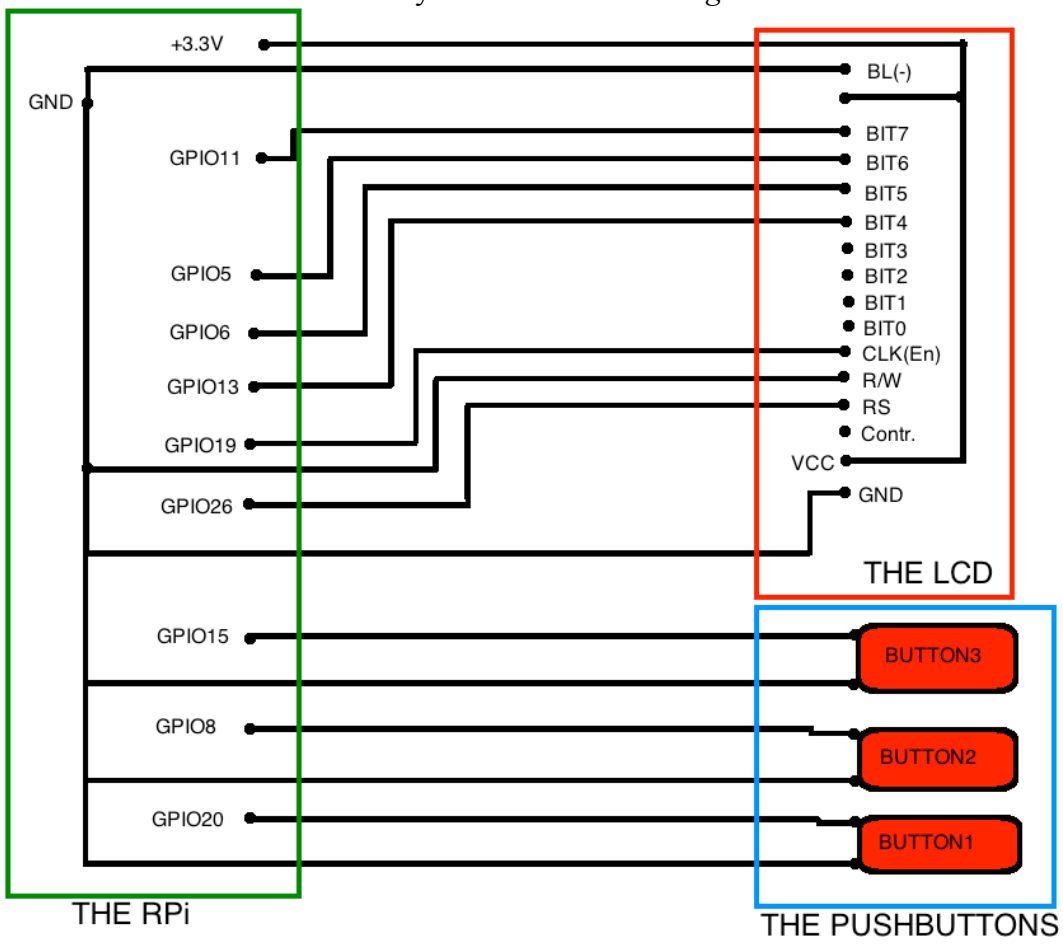

Figure 7, The GPIO connections layout

And lastly, the RFID reader is connected to the device via USB.

3.6 The casing

To make the system more durable, a case was constructed. The case was made of plexi-glass and 4 metal columns. The glass was connected to the RPi by putting bolts through the bolt sockets of the RPi, which were meant for the original RPi case. The LCD is mounted on top of the RPi. The columns have a height on 40 mm, which is the same height as measured from the bottom of the RPi to the top of the LCD. The pushbuttons were inserted into the glass at the top and were numbered. The device is shown on *Figure **8***.

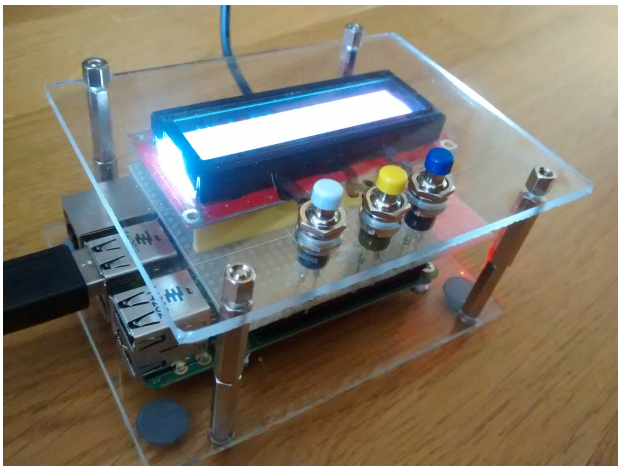

Figure 8, Prototype case

The RFID scanner was put into a box for durability. It is connected to the RPi via a USB cable. The whole system is visible on *Figure **9***.

Figure 9, Complete prototype with the RFID reader

4. Building the Lending System

The Raspbian operating system supports a number of different coding evironments and server options. In the scope of this project we used:

- IDLE for coding Python (Python 2.7) – Python comes preinstalled on the Raspbian operating system and is the language on which most our coding was done for this project.
- MySQL server – The MySQL server was installed for holding all the information about the lending system. PHPMyAdmin was used for server administration. PHPMyAdmin is a graphical tool for MySQL administration. It has the ability to create new tables and create relations between them. Also it can be used to import new SQL data in the form of querries.
- GIT – will be used to download the Adafruit LCD library for working with our 16*2 LCD.
- Nano – Nano is a text editor that comes with UNIX based systems. It is another convenient way for editing code or some of the text files needed in this project.
- Apache web server [18] – Apache allows the Raspberry to host web pages on its IP address.

All the code for the project will be kept in a folder called “Project”. In the documentation all Python modules related to the hardware will be kept there. The project folder will be provided together with this thesis on a CD.

4.1 Software development

4.1.1 Introduction

During the course of this project we will be using IDLE (Integrated DeveLopment Environment) for editing our code.

The project’s main file of operations will be the Menu.py. The Menu.py script gets its functionality from 5 other modules, which we will need to create. They are all shown on *Figure **10*** and all have different colours, depending on what they are going to be used for. The blue colored modules are used for interfacing the hardware connected to the RPi – Simple_reader.py, Button2.py, Adafruit_Char_LCD.py. The yellow module will be used for

all MySQL database handling – SQLinterface.py. The white module will be imported for makng the LCD work more naturally – Time.py.

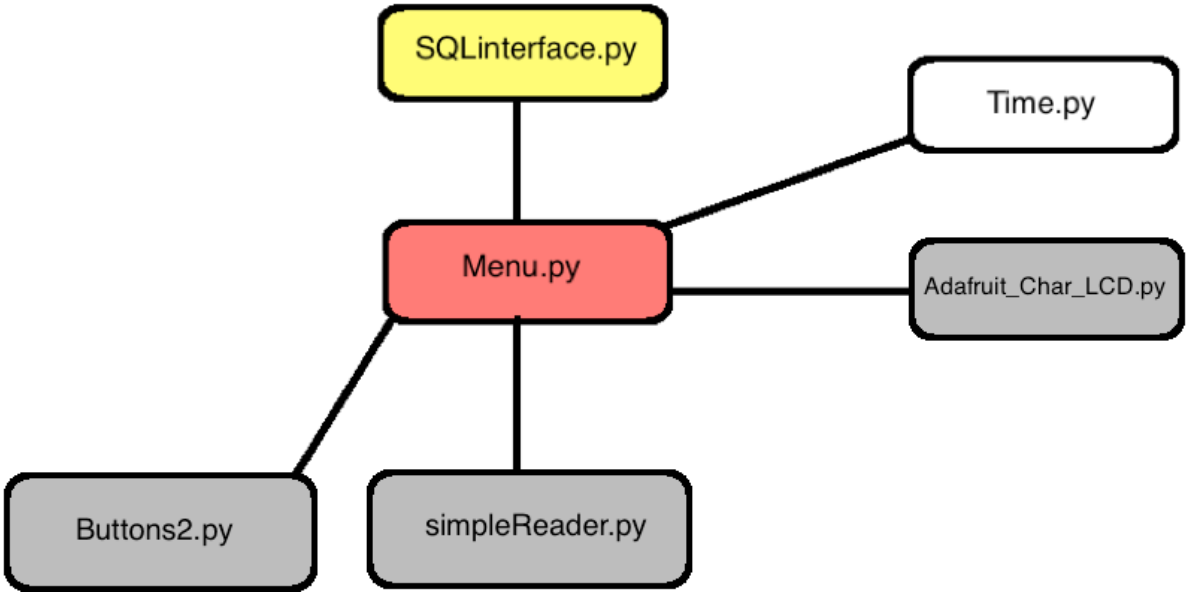

Figure 10, Menu.py modules

4.1.2 System Architecture

The system consists of two parts - the lending system and the option to add users.

The lending system consists of the hardware connected to the Pi, the lending system menu created in the Raspbian operating system and the MySQL database. The MySQL database is a local database on the Raspberry. The hardware and the lending system menu work using Python. The menu works by using Python to querry the MySQL database.

The option to add users works using the hardware connected to the Pi with a web page, which can be run from another computer. The option to add users is displayed in the Python menu on the LCD. The web page is an Apache[18] web server running on the Raspberry. The new student's data will be stored in the lending system's MySQL database.

4.2 Creating the hardware modules

Before creating the code for the Menu.py, modules need to be created for the connected hardware. The modules will need to have inputs from the hardware in the case of the buttons and the RFID reader or will need to give an output to the hardware in the case of the LCD.

4.2.1 User pushbuttons

The buttons are connected to the GPIO pins and are interfaced using the Rpi.GPIO module, since it comes with a set of pre-installed functions. The newest version of Raspbian has this library pre-installed so for this project's scope we will assume that it is installed.

We will create a new Python script and call it Button2.py and import the Rpi.GPIO in the code's header.

After it has been imported, we will create a function called buttonFunction(), which will, when run, always be waiting for a button input until it has got an input from one of the 3 pushbuttons. The pushbuttons will need to be using pull-up resistors, to keep them from being in a floating state. For example, GPIO 20 was defined as an input using a pull-up resistor, as shown on *Figure **11***.

```
def buttonFunction()

GPIO.setup(20, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

Figure 11, Defining a button

Once a button is pressed, it will return an integer value of either 1,2 or 3, depending on which one was pressed. The function uses event detection for as long as it is running. For example, to use event detection on pin 20, we tell the program to look for a falling edge signal, as shown on *Figure **12***.

```
GPIO.add_event_detect(20, GPIO.FALLING)
        while True:
                if GPIO.event_detected(20):
                        rem()
                        return 3
```

Figure 12, Adding event detection

Once the edge has been found, then the program closes all event detection before returning a value, as shown on *Figure **13***.

```
def rem():
        GPIO.remove_event_detect(20)
```

Figure 13, Remove even detection

4.2.2 The RFID reader

Next we connect the reader to the Pi and see which USB port it is sending its data to. In the case of the Raspberry it is /dev/ttyACM0 in the Raspbian operating system.

To get the information from the USB port, a new Python script needs to be created. Firstly, the serial library needs to be downloaded and installed via the terminal for using the serial interface on the Raspberry.

Next we create a python module, called simpleReader.py. The module consists of 1 function, which tries to read a tag. If it gets an error, an exception is given and the module returns a 0. The function uses the readline() function for reading the com-port.

An infinite loop scans for the tags but does not return a value right away. The loop has an "if" statement which searches for the fifth cycle in the loop and for the line in the com port to contain "ISO". If the condition is met, then the text around the tag is discarded and only the tag id is returned, as shown on *Figure **14***.

```
import serial
def tagRead():
    try:
        serial= serial.Serial("/dev/ttyACM0", baudrate=9600)
        …
        while True:
          data=serial.readline()
                n=n+1
                if data[0:3]=="ISO" and n>3:
                    myString1=data.find('[')+1
                    myString2=data.find(',')
                    …
```

```
                return data[myString1:myString2]
    except Exception as e:
        return 0
```

Figure 14, RFID reader code

4.2.3 The LCD module

For the scope of this project Adafruit's LCD library will be used[1]. The Adafruit LCD GIT repository has a wide variety of different Python modules. The only one of interest for this project will be the Adafruit_Char_LCD.py.

The module needs to be modified to use the correct pins for our LCD. So in the Adafruit_CharLCD, the following line, shown on *Figure **15***, needs to be changed.

```
def __init__(self, pin_rs=26, pin_e=19, pins_db=[13, 6, 5, 11], GPIO=None)
```

Figure 15, Defining the correct GPIO pins

Once the pins have been set, the LCD will be ready to be used. We will move the Adafruit_CharLCD.py to our main Project folder.

From the Adafruit_CharLCD library 2 functions will be used:

- lcd.message() – this sends any text put between the brackets to the screen of the LCD. It also uses "\n" as a linebreak.
- lcd.clear() – this function clears the LCD screen when called. This is necessary, since otherwise all symbols would be kept in a buffer and pile up on one another.

4.3 Python Menu Script

This Python script will hold our lending system project. In this chapter all the necessary functions will be defined for building the user interface functions. The menu will consist of 6 main views for the user interface, as shown on *Figure **16***, and are split to levels, through which the user can move back and forth. These Views are defined more thoroughly in chapter 5.

[1] To obtain the module, we fetch the Adafruit Raspberry Pi LCD library from the Adafruit GIT page: `git clone git://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git`

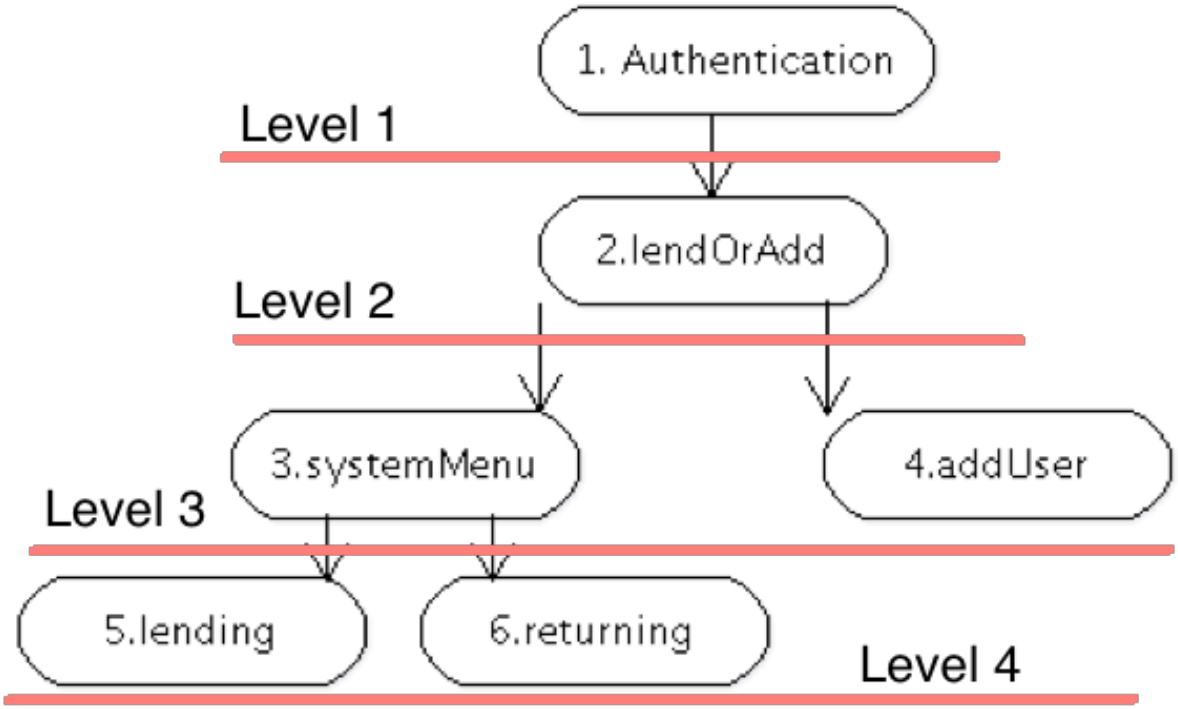

Figure 16, Menu views

4.3.1 Importing the modules

The first thing to do should be to import all the previously defined Python modules into the newly created Menu.py, since this will be our main program and will be using all of their functionality. On top of that, some some more modules will be imported to make the menu a little more functional.

Next the previously downloaded and already prepared Adafruit_CharLCD, the pushbutton operating module “Button2” and the RFID tag reader module “Simple_reader” will need to be imported. On top of the hardware modules, two more modules will be required, time and the SQLinterface.

4.3.2 Python Time module

The time Python module will be used for the messages on the LCD. We will be using the sleep() function to pause the program, whenever the LCD displays a message. Otherwise the program would run too quickly and the message would only be displayed for a fraction of a second. The sleep function can be passed a value in the brackets to tell it, how long it will be required to pause for.

4.3.3 Global variables

Our menu will be using 2 key global variables in this project. The first being menu_actions, which will make moving between views a little easier and will hold 6 different menu areas in it. The second being respID, which will be the person responsible for all transactions done after security authentication has been made. Both of them will be defined at the top of the Menu.py code under the modules.

The respID will have a default value of 0. The menu_actions will need to be defined, for that we will need to create our functions one by one and start writing into menu_actions.

4.4 Switching between Menu functions

4.4.1 The Menu options

As defined above in the global variables, one of the variables is a structure type variable, which will hold all of our menu items with numbers to make moving between menus a little easier, as shown on *Figure 17*.

```
menu_actions = {
    '1': authentication,
    '2': lendOrAdd,
    '3': systemMenu,
    '4': addUser,
    '5': lending,
    '6': returning,
}
```

Figure 17, Global user menus

4.4.2 Switching between Menu functions

In order to move from one menu function to another, a function was created called exec_menu, which works by verifying the number it is given as an input to see if the number is between 1 and 6. If the number falls in this range, then the corresponding menu number is launched.

4.4.3 Moving to the correct function

Moving from one Menu to another needs to be structured. Meaning movement should only be allowed to the menu below you or above you and the third button will always be the button that can be used for quitting back to the second screen and if at the second screen it can be used for quitting. For this purpose a new function called tryAgain was created, which takes the number of the menu you are in at the moment as the input and allows the user to then choose one of the three buttons. According to the menu from which the user is coming from, the user will have a choice of what they want to do. Always having the choice to try again if some sort of RFID scanning is involved.

In the case of menu_actions function number 2 for example, the tryAgain function input is 2. That gives the option of moving forward to 1. addUser, 2. systemMenu or back to 3. authentication, see *Figure **16***.

In the case of a menu where RFID scanning is involved, then pressing button 1 will always launch the same menu again. The second button allows the user to move up by one view. The third option allows quitting to lemdOrAdd or authentication, from where a new administrator may be scanned, who will be responsible for all the new transactions done from that point onwards.

4.5 MySQL database

The use of a MySQL database was a precquisite for this thesis.

Installing the MySQL database[4] functionality gives the opportunity to store all the data needed from future RFID tags, equipment to be lent, student users and faculty administrators. Seeing as viewing raw MySQL data is very difficult, then the best thing to do would be to install a graphical MySQL administrating tool to make administration more simple and to give the faculty members a way to view the databases more graphically.

Database administration is done using PHPMyAdmin[14].

PHPMyadmin is a graphical MySQL administration tool, which, once installed and correctly configured to connect to your database can be used to alter the database from the web browser. Also, it makes working with the Raspberry Pi simpler, since you can now connect to the Pi's PHPMyAdmin interfaced database from another local network computer's browser. If

the Raspberry's IP address were 192.168.0.5, then the address would be: http://192.168.0.5/phpmyadmin.

One of the slowest features of the RPi is its web browsing capabilities. For this reason connecting to it for another computer is a much faster way since the raspberry does not need to do any of the graphical work needed to open the web page.

4.5.1 Importing the schema

PHPMyAdmin provides a nice graphical interface for working with the MySQL database and can do a lot of the SQL querries for the user. After logging in to PHPMyAdmin a screen of all the databases, that it is connected to, will be opened, as shown on *Figure **18***.

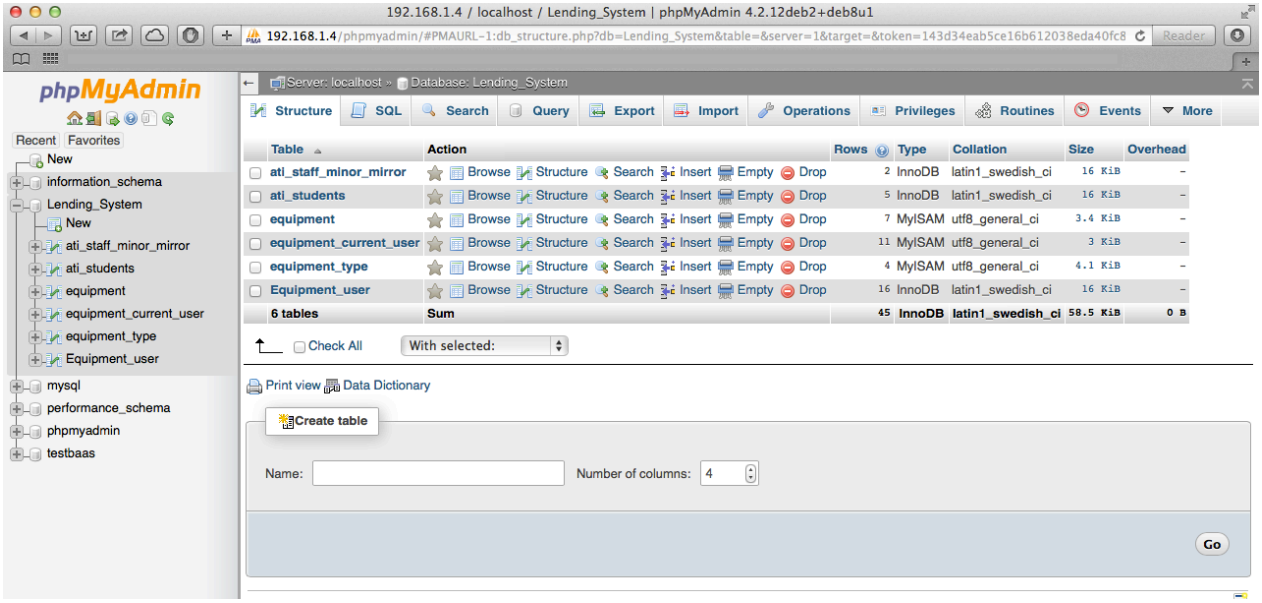

Figure 18, The graphical interface of PHPMyAdmin

Since TUT provided the database schema for this project, it will need to be imported (CD). To import the schema, a new database was created for this project. To simplify the project, it was called "Lending_System".

4.6 Lending system database

One of the requirements of this thesis was that the database run on the same kind of schema as the one already used in TUT. For this purpose 4 MySQL databases were imported. Their structures will be available on the CD provided with this thesis.

This database will for the duration of this thesis be simulated in a localhost server.

In order to idenify the user responsible for lending out equipment, the lending system program needs to check the RFID tag of the person attempting to operate the lending system. This requires the person responsible for lending to be a member of the faculty and be a member of the database “ati_staff_minor_mirror”. Every faculty member that can use the database is represented in the database at the moment. The structure of this database will remain unaltered, this is the way requested by the university. The ID of the user in the “ati_staff_minor_mirror” is its foreign key in the “Equipment_user” table.

All students able to use the lending system are kept in the “ati_students” table. The table has basic information about the users and is has its primary key ID as a foreign key in Equipment_user.

The “equipment_type” holds the equipment’s type. It was provided to better understand just how many foreign keys every database holds in the TUT. Thanks to this it was decided to only use the essential parts for creating this database.

The “equipment” table holds all the information available for every equipment item available for lending. It had 18 columns in its table to begin with. A 19-th column was added in this project to hold the value of the RFID tag used for its corresponding equipment piece. Every item in this list is given an ID as a primary key, which will be used as a foreign key in the “Equipment_user” table.

Finally the “Equipment_user” table holds the information of the lending done in the project. It consists of:

- ID, which is the primary key and is always auto-incremented for every new lent out item.
- equipment_ID, the foreign key from the equipment table.
- user_ID, which is the student’s ID who lent the equipment. It is a foreign key from ati_students.
- responsible_ID, the person who lent out the information. A foreign key from ati_staff_minor_mirror.
- is_active, is a Y or N value. This indicates whether or not this equipment is still lent out or not.
- date_issued, holds the date and time of the time this equipment was issued for lending.
- date_returned, holds the date and time of when the equipment was returned.

- returned_to, holds the ID of the faculty member to whom the equipment was returned to.

A view of the relations between the tables of the database is provided in *Figure **19***.

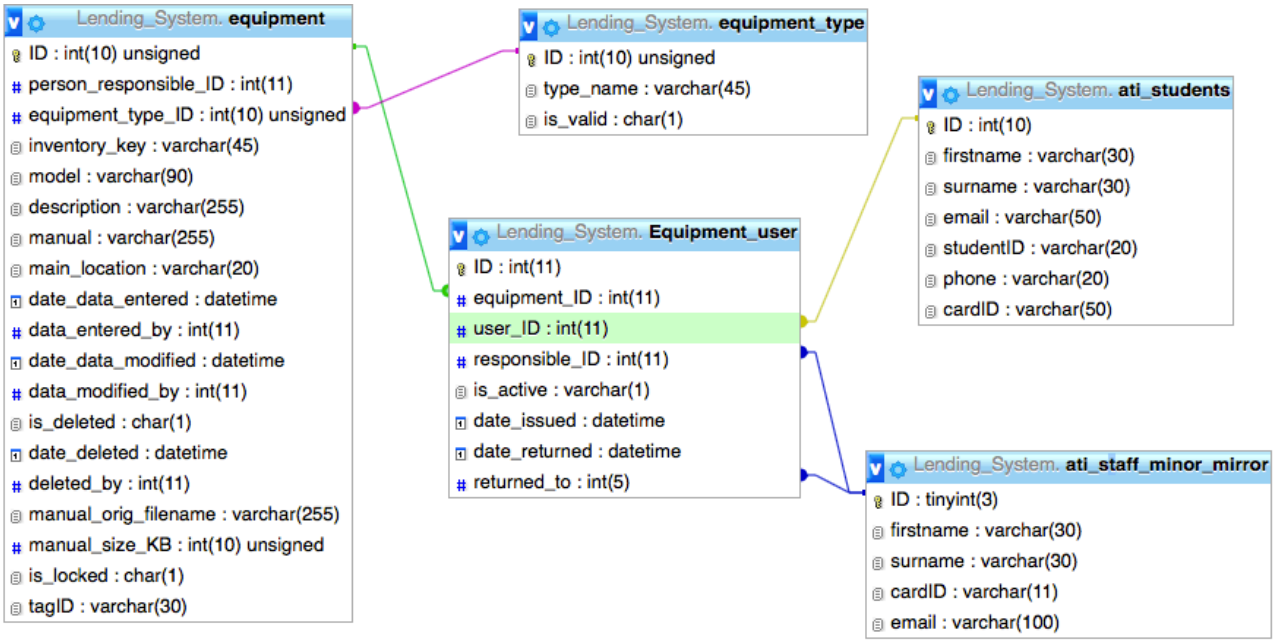

Figure 19, The Lending system database's relationship view

4.6.1 Python script - SQLinterface.py

This module does all communicating with the MySQL database with python. It works using the MySQLdb module. It has 8 functions – checkConnectivity, checkIfExist, getID, writeID, modifyRow, checkRow, checkRowLend, returnVarious.

SQLinterface.py works by logging in to the MySQL database. After which firstly according to the function it can either fetch data from the database as a check for something. Or secondly, it can add data to the database.

The interface works by using a cursor to execute SQL statements. It will then use the previously defined connection to the database (available on the CD) to execute the statement. An example of how the lending information is written to the Equipment_user table is given on *Figure **20***.

```
def writeID(respID, stdntID, eqID):

  con=mdb.connect('localhost','root','root','Lending_System');

    with con:

        cur=con.cursor()

        cur.execute("INSERT INTO Equipment_user(equipment_ID,
user_ID, responsible_ID, is_active, date_issued) VALUES (%s,
%s, %s,'Y', NOW())", (eqID, stdntID, respID))
```

Figure 20, SQLinterface.py

4.7 Web interface for adding users

The web page works by using an Apache[18] web server on the RPi. Once installed, it allows adding files to its folder for access over the network. In this project 2 files will be used index2.php and insert.php.

index2.php will work by loading a web form which asks for the user's information(First name, Surname, e-mail, phone number, student ID number and an RFID tag number). The web page works together with the lending system. The lending system has a function to add users. When launched a tag will need to be scanned for the new user. This tag is stored into a text file called **tag.tx**t with Python. It will remain there for 2 minutes, after which another Python script called tagNullifier.py erases it by checking if its size has become bigger than 0. During this time the scanned tag will appear in the sixth form field each time, the web page is loaded, as shown on *Figure **21***. If the cardID field is empty, then a new tag needs to be scanned.

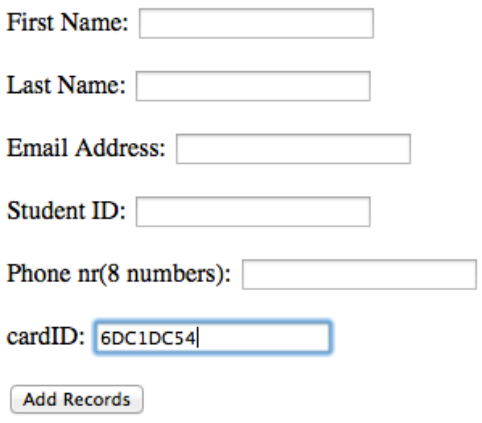

Figure 21, Web form for adding users

If all the fields are filled out, then the “Add Records” button checks the inputs using javascript. If all the fields match the criteria checks of the web page then the data is sent to insert.php.
The other function, insert.php, connects to the local MySQL database and passes on the information which was inputted into the fields into the ati_students database.

4.8 Error checks

All functions, which will be used RFID scanning, depend on the RFID reader connectivity and on a connection to the MySQL database. If neither is working, then the program would crash. To prevent this, corresponding error checks need to be made inside the functions.

4.8.1 SQL connectivity

Prior to beginning operation with all the menu user interface functions, a test should always be run to check for SQL connectivity. Since the lending system is supposed to connect to the school’s database in the future, then this check is mandatory, since the connection can always go down. This function will be called in every function that checks the database, right after an RFID scan has occured. Should the SQL server be down, then the system waits for 20 seconds pior to trying to establish a new connection.

This function uses the SQLinterface module to check the SQL version of the database and if it gets an answer from the database then it will return 1 or else a 0. This value is passed back to the Menu.checkSQL function, which in the case of a 1 will keep the function from which it was called from going. Elsely keep retrying the SQL connection.

4.8.2 RFID reader connectivity

The RFID reader Python module Button2.py had an error handling exception in it in case an error came up. In that case it returns a 0 to the calling function. If a 0 is returned, then the LCD displays a message of an error with the RFID reader and asks the user what they would like to do next, depending on the function they are using (see chapter 4.4.3 for more).

In case an error is thrown during authentication, then authentication is run again after 10 seconds.

When the program restarts, the RFID connection should be re-established by trying to reconnect the RFID reader to the USB port.

4.9 The main program

Since main will be the first function to be exectued, then it is imperative, that the first view is launched from it. For this authentication will be used, which authenticates the person responsible for all the lending done from that point forward, until the next authentication.

4.10 Launching the program

The Lending system is meant to run without the need for a display to launch it from the Raspbian operating system. For this purpose a crontab job was created, which launches a shell executable (.sh) file called Launcher.sh every time at startup.

Launcher.sh consists of two Python scripts, which will be executed to run at startup. The scripts are Menu.py and tagNullifier.py.

Menu.py is our main program and tagNullifier.py is used for cleaning out tag.txt, which holds the information on the new user's tag to be added to ati_students, in case someone wants to add themselves as a user.

5. The Menu

The menu works by putting together all the checks, hardware modules and MySQL querries from the previous chapters. This part of the project describes how the Menu functions, which make up the user interface, work.

5.1 Authentication

A use case of the function is shown on *Figure 22*.

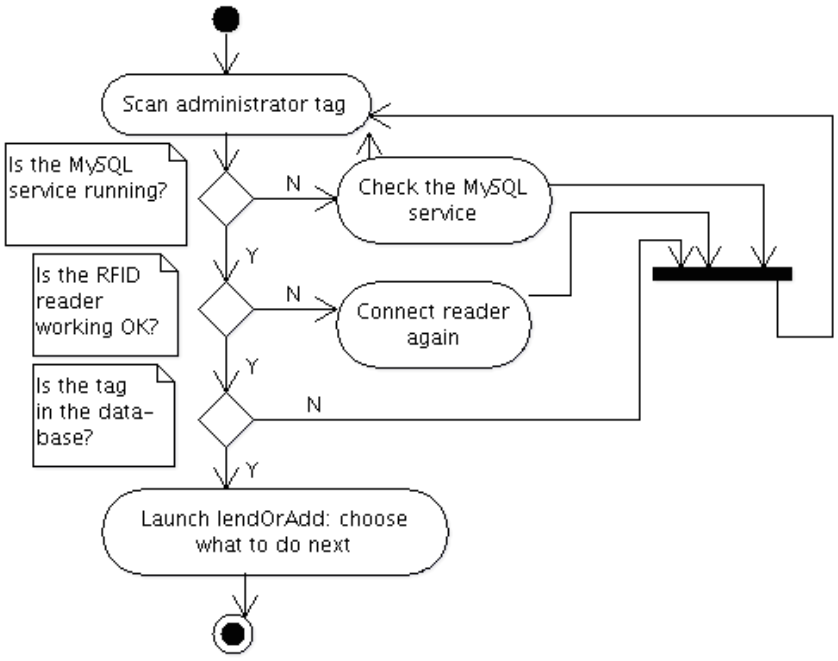

Figure 22, Authentication function Use case diagram

In the previous chapters, the main program was defined to start and launch the authentication function. This function will be responsible for unlocking the device for usage. Unless authentication is passed, the device will not be open for using. Authentication works by first presenting the version of the lending system on the LCD screen and then asking the user for their administrative tag.

The tag will be scanned using our previously defined scanner module. Next a check will be done for a MySQL connection and if the reader is working properly. If all is working and the

scanning was done, then the scanned tag will be compared to the ones stored in Lending_system's ati_staff_minor_mirror table. If a match is found, then next the corresponding tag holder's ID is taken from the database. This ID is then used as a global variable(respID) for as long as the program is active for all Lending system transactions. This ID is changed only when authentication is launched again and a new responsible ID is scanned. After this the function moves to the lendOrAdd function.

If the user is not in the database, then the loop begins again, and asks for the person's administrative tag.

5.2 The lendOrAdd() Function

This part of the program works as sort of a gateway for picking the next thing the user would like to do.

The function is launched right after succesful authentication and gives the user the option of choosing what they would like to do next. Once the choices are displayed on the LCD screen, the function waits for a button input from where the user chooses if they would like to add users or use the lending system.

A use case of the function is shown on *Figure **23***.

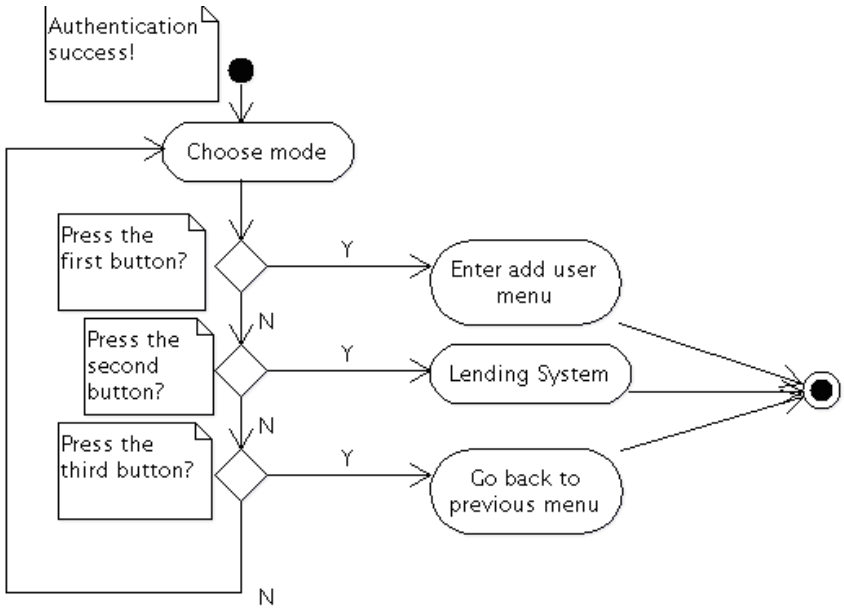

Figure 23, lendOrAdd function use case diagram

5.3 Adding users

Since using PHPMyAdmin can become tiring for doing all the adding, another task was formed for creating a way for users to sign up to the system all by themselves. A simple web form was created for this purpose on a PHP web page (see section 4.7).

A use case of the function can be found on *Figure **24***.

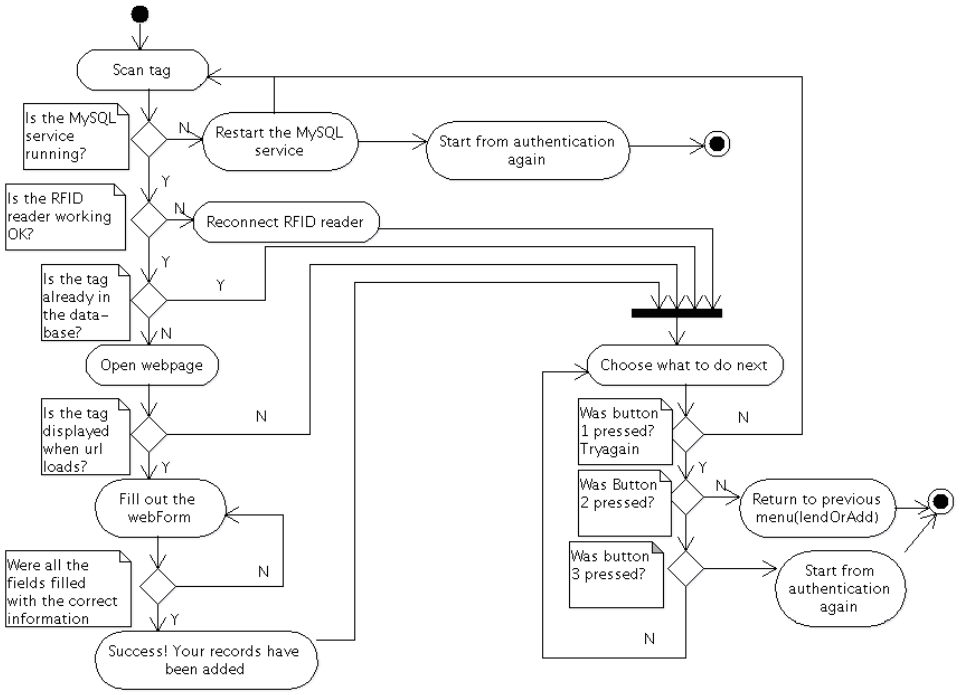

Figure 24, addUser function use case diagram

Upon launch, the addUser python function tells the user to scan the tag to be added. Then the RFID reader waits for the RFID tag. Once the tag is scanned, then the function checks for MySQL connectivity and RFID reader connectivity. If all is functioning, then the tag is compared to the ones already in the database, to check that it does not already exist. If it does not, then the tag is written to the **tag.txt** file. After a successful scan, the user is given a choice, if they would like to scan another tag. If the tag does exists in the database, then the LCD displays an error message about the tag already being in the database and the user is asked, if they would like to try again.

As already stated in chapter 4.7, if the tag is scanned succesfully, then the user will have 2 minutes to open the web interface and input his information to the web form.

Since the Raspberry is running a web server, then all of this can be done remotely in the same Local Area Network by logging into the RPi with another computer's browser. Let's say the Raspberry's IP is 192.168.0.5, for example, then the web page's address would be -

```
http://192.168.0.5/index2.php
```

If all the information is inputed into the fields correctly, and "add records" is clicked, then the browser displays, that the records have been added succesfully.

After 120 seconds, the text file is overwritten by an empty text file.. If the time is up, then the tag will disappear from the form field after reloading the page.

5.4 The systemMenu Function

The lending system is the submenu of lendOrAdd, and is called systemMenu in the code. If launched, the user will see the display asking them if they would like to lend or return equipment or if they would like to return to the previous menu. After this, the program waits for a button input from one of the 3 pushbuttons.

A use case of the function can be found on *Figure **25***.

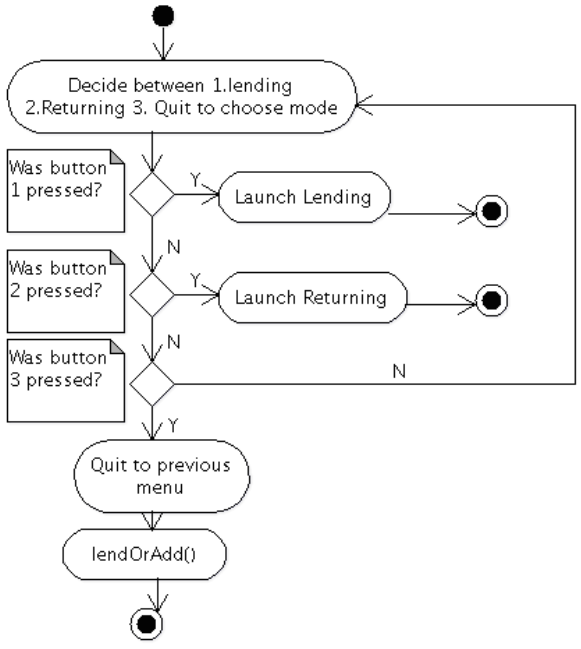

Figure 25, systemMenu function use case diagram

5.4.1 Lending

Lending is done by telling the user to scan an equipment RFID tag on the RFID scanner. After which the reader and SQL connection are checked. If everything is working and the tag is scanned, then the tag is compared to the ones in the database. In case of a match, the LCD propmts the user to scan their own student tag. Once the tag is scanned the same kind of checks are made as during the equipment scan. If the user tag exists, then the display shows the equipment ID and the person's first name it was lent to. After this the user ID, equipment ID and the global ID for the person reponsible(respID) are all written into the Equipment_user along with the date and time at that moment and a 'Y' into the is_active column.

Each time a check does not pass, the user is prompted to try again, if they would like. If all checks pass, then the user sees a message that all was succesful on the LCD and is again asked, if they would like to try again.

A use case of the Lending function can be found on *Figure **26***.

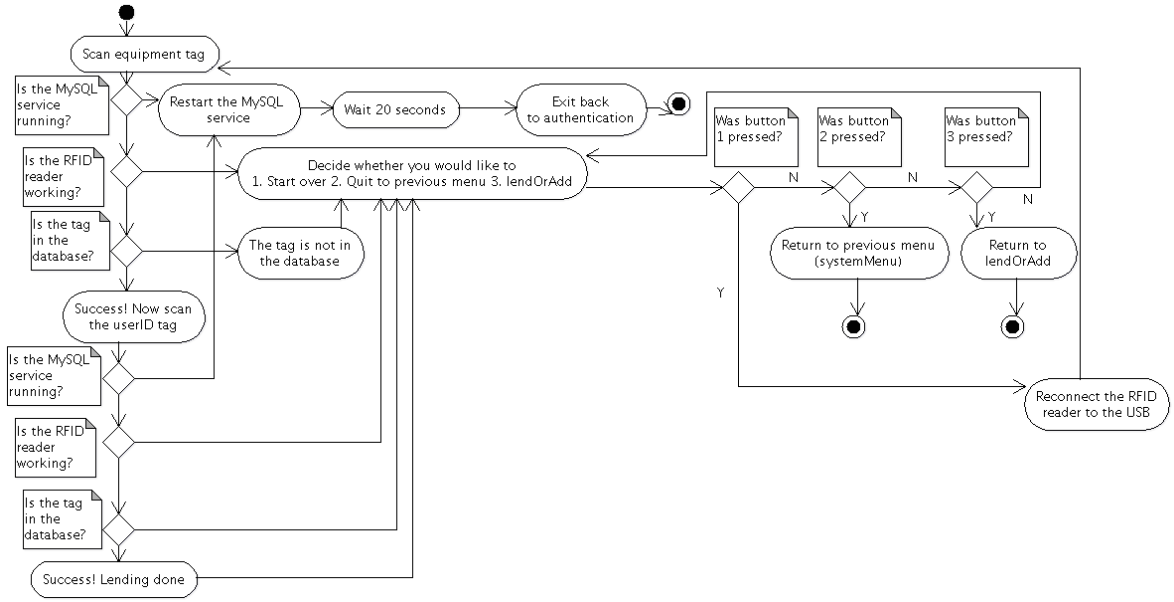

Figure 26, Lending function use case diagram

5.4.2 Returning

Returning works by asking the user to present the tag on the equipment for returning. The user presents the tag. Next, the sql connection and RFID reader connections are checked. If all is correct, then the tag is compared to the ones in the Equipment_user database. If something is wrong with the RFID reader, then the LCD will tell the user to check the connection of the

RFID reader. If something is wrong with the MySQL connection, then the user is told that the system will restart.

A check is made to see if the tag is in the database and has in fact been lent out. If the tag has not been lent out, then the system displays an error message.

If all checks are passed, then the LCD displays equipment tag which was returned. The function then finds the corresponding line in the Equipment_user database and writes in the values

- Date and time returned
- Returned_to, by using the global variable respID
- is_active: 'N' – to understand that the equipment is now returned.

A use case of the Returning function can be found on *Figure 27*.

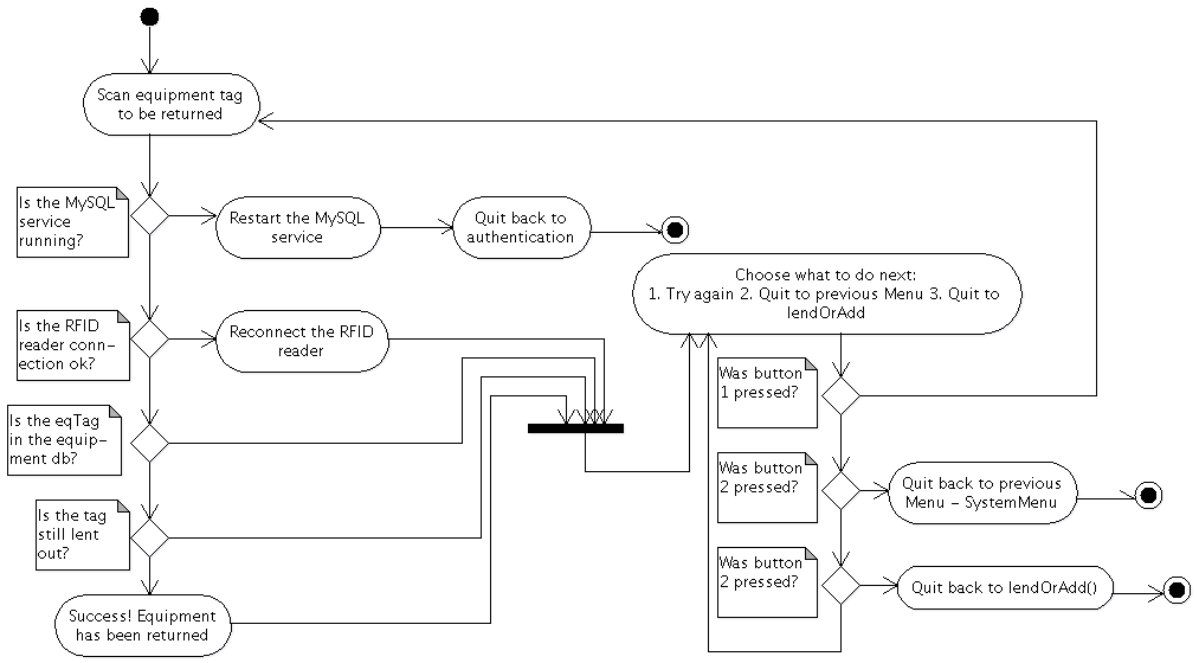

Figure 27, Returning function use case diagram

6. Conclusions and future development options

The objective of the thesis was to create a lending system for the department of Computer Engineering of Tallinn University of Technology.

The result of the thesis project is a working prototype, capable of working with its own local database. The device was not connected to the TUT's database yet, but will be connected after the testing period with a limited number of testmodules is over. The project definetely improved my skills in programming with Python, HTML, PHP, Javascript and MySQL.

The device allows users to lend and return equipment from the TUT. It allows registering a user by using a web page and database administration with PHPMyAdmin.

The RPi is connected to hardware - 3 pushbuttons and 1 LCD screen are connected to the GPIO pins of the device. A USB RFID/NFC reader is connected to a USB port of the RPi. An enclosing case was created for the RPi and another for the RFID/NFC reader.

During the course of software development 5 python scripts were created and 1 was downloaded from GIT. A database was created to store the lending system's data. With PHPMyAdmin 4 tables were imported to emulate the database at the department of Computer Engineering's lab. Also one extra table was created for holding the information of the transactions. A web page was created for adding users to the Lending System's database.

The hardware was put to work using Python scripts. The main program is situated in Menu.py, which uses the functionality of Button2.py, simpleReader.py, SQLInterface.py and Adafruit_CharLCD.py. The web page works via an Apache web server and is accesible from a computer in the same local area network. One more Python script runs indipendantly of the program to check for the web page's input file. The code for all the files is available on the CD provided with this thesis.

Future developments would be to connect the device to the TUT's database. If the students do not have their own RFID tags, then the University could consider handing out RFID tags. The system could be replicated to work in other departments of the school as a library system for similar equipment or other such items.

References

1. Ahuja S., Potti P. An Introduction to RFID Technology. *Communications and Network*, Vol. 2 No. 3, 2010, pp. 183-186. [Online]. Scientific Research Publishing Inc. (14.12.2015). (An article from the online database).
2. Bibliotheca homepage. [WWW] (04.01.2015) http://www.bibliotheca.com
3. Company Overview of Bibliotheca RFID Library Systems AG. [WWW] http://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=32281724 (01.01.2016) (An article from the web).
4. IBEX. MySQL – Raspberry Pi Projects. [WWW] (An article from the web) (27.11.2015) http://www.raspberry-projects.com/pi/software_utilities/mysq
5. Instructables. Attendance System using Raspberry Pi. [WWW]. (05.12.2015) http://www.instructables.com/id/Attendance-system-using-Raspberry-Pi-and-NFC-Tag-r/
6. ISO14443 standards. [Online]. Atmel Corporation (14.12.2015) (An article from their online database). http://www.atmel.com/images/doc2056.pdf
7. Hitachi HD44780 LCD controller. [WWW] https://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller –(20.12.2015) (An article from the web)
8. Kiyosaki, R. T., & Lechter, S. L. (2000). Rich dad, poor dad: *What the rich teach their kids about money-- that the poor and middle class do not!*. New York: Warner Business Books. (A book)
9. Kiyotaka F. Implementation of a RFID-based System for Library Management. *IGI Global*. [Online]. (2015). (An article from the online database
10. Kleback, M. "Raspberry Pi GPIO Pins and Python". Make. [WWW] http://makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/ (21.12.2015) (An article from the web).
11. Kumar, P., Reinitz, H.W., Simunovic, J., Sandeep, K.P. and Franzon, P.D. (2009), Overview of RFID Technology and Its Applications in the Food Industry. *Journal of Food Science*, 74: R101–R106. [Online]. Journal of food science(14.12.2015) (An article from the online database).
12. ModMyPi | Tutorial: Tactile Switch. [WWW] http://www.modmypi.com/blog/tutorial-tactile-switch (21.12.2015)(An article from the web).
13. Nosowitz, D. "EVERYTHING YOU NEED TO KNOW ABOUT NEAR FIELD COMMUNICATION". Popular Science. [WWW](05.01.2015). http://www.popsci.com/gadgets/article/2011-02/near-field-communication-helping-your-smartphone-replace-your-wallet-2010
14. PiMyLifeUp. Raspberry Pi MYSQL & PHPMyAdmin Tutorial. [WWW]. http://pimylifeup.com/raspberry-pi-mysql-phpmyadmin/
15. RFID based Library system. Embedded kits co. [WWW] http://embeddedkits.co.in/index.php?route=product/product&path=35&product_id=563 (05.12.2015) (Product information)
16. Raspberry Pi 2 model B. Oomipood.ee. [WWW] http://www.oomipood.ee/product/2461029/raspberry-pi-2-mudel-b-moodul-1gb&s=raspberry%20b

17. Raspberry Pi Foundation. Raspberry Pi Model B+. [WWW] https://www.raspberrypi.org/products/model-b-plus/ (28.11.2015) (An article from the web)
18. RASPBERRY PI FOUNDATION. Setting up an Apache Web Server on a Raspberry Pi. [WWW] https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md
19. RASPBERRY PI FOUNDATION, (2012). What is a Raspberry Pi? [WWW] https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (27.11.2015)(An article from the web)
20. Wolfgang R., Wolfgang E.(2010). Smart Card Handbook, 4th edition. Germany: Giesecke & Devrient GmbH.
21. Sparkfun electronics inc. Basic 16x2 Character LCD. [WWW] https://www.sparkfun.com/products/709 (06.12.2015) (The specs of the LCD).
22. Sparkfun electronics. Switch Basics. [WWW] https://learn.sparkfun.com/tutorials/switch-basics (21.12.2015) (An article from the web).
23. Swedberg C., (2014). Iotera develops tag with over 4 mile read range. [WWW]. http://www.rfidjournal.com/articles/view?11374 (24.12.2015) (An article from the web)
24. Texas instruments. MSP430 Launchpad Value line Development kit. [WWW] http://www.ti.com/tool/msp-exp430g2 (04.12.2015) (Product information).
25. Texas Instruments. NFC Transceiver Booster Pack. [WWW] http://www.ti.com/tool/DLP-7970ABP (04.12.2015) (The product information).
26. Texas Instruments(2012). RFID Boosterpack. [WWW] http://www.ti.com/lit/ml/slab068/slab068.pdf (04.12.2015) (instructions for the TRF7970).
27. Tallinn University of Technology contact information. [WWW] https://www.ttu.ee/asutused/raamatukogu/kontakt-2/lahtiolekuajad-2/ (28.12.2015) (opening hours of the library)
28. Tran W., Potnis V., Gates J., Howell M.. Using ISO 15693 Compliant RFID tags in an Inventory Control System. IEEE. [Online] (An article from their online database) https://www.ieee.org/education_careers/education/standards/using_iso_15693_compliant_rfid_tags.pdf
29. Wehr, J (2003). Contactless card standards: making sense of 10536, 14443 and 15693. SecureIDNews(01.05.2013) (An article from the online magazine).