Suzanna Wentzel - s1850512 Milo Cesar - s1829688  
Jesper Simon - s1820338 Danique Lummen - s1853155  
Chris Witteveen - s1847821 Apostolis Christoulias - s1833383

**Design Report for**

**Door Authorization**

**in Care Centre “Liemerije”**

31 October 2016

**1. Problem Definition**

**1.1 Problem Scope**

In care centre “Liemerije”, clients and nursing staff have trouble with getting into different rooms with different keys. Finding the right key for a certain door takes time which in the case of the nursing staff can be better spent in helping the clients. The finding of the keys and the opening of the doors in the way used now is very inconvenient and thus this is a problem that needs to be solved.

Using this security system we want to prevent gaining useful information by eavesdropping on the connection after the UID has been scanned (which might lead to unauthorized copying of UIDs) and make the access to different rooms easier for the clients.

**1.2 Technical Review**

**Detailed background:**

Care centre ‘Liemerije’ is a care centre for the elderly people. Most clients have either dementia or have big health issues that causes them to no longer be able to live at home without constant care. In order for the clients to get into their own room, or into the general living room for example, they need to find a member of the nursing staff who has a set of keys so this person can open the doors for them. This takes time, especially when the nursing staff is busy with caring for other clients.

**Prior art:**

The prior art used is using physical keys. These keys only fit to one lock (there do exist keys that fit to multiple locks), mostly those locks work with pins, which fall into place when the right key is inserted, and then the door can be opened when the heck is pulled down. Using these keys is slow, the nurses have to find the right key for the right lock. Also, a lot of keys are needed.

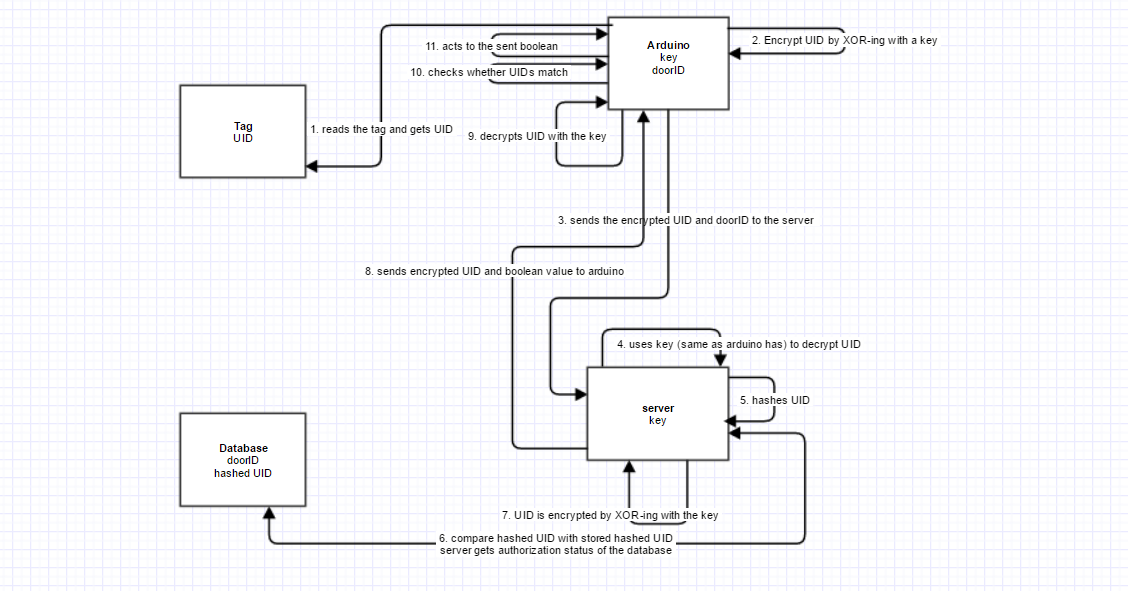
**1.3 Design Requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Design requirement** | **Importance** | **Units** | **Marginal value** | **Ideal value** |
| Unlocks after 3 seconds | **9** | Seconds | 3 | lowest as possible |
| The door remains unlocked for 15 seconds, if the door is not opened | **8** | Seconds | 15 | 15 |
| The door locks itself if the user closes the door | **3** | True/False (Boolean) | 1/0 | 1 |
| The door doesn’t open for unpermitted tags | **1** | True/False  (Boolean) | 1/0 | 1 |
| The door does open for permitted tag | **2** | True/False  (Boolean) | 1/0 | 1 |
| Roles, persons and doors can be added or deleted to the database | **6** | True/False  (Boolean) | 1/0 | 1 |
| All doors can be opened at once in case of emergency | **4** | True/False  (Boolean) | 1/0 | 1 |
| All doors can be closed at once in case of emergency | **5** | True/False  (Boolean) | 1/0 | 1 |
| Permissions for persons or roles can be added or deleted to the database | **7** | True/False  (Boolean) | 1/0 | 1 |

**2. Design Description**

**2.1 Overview**

The design gives or denies access to certain specific doors in the facility. It does this by using key cards and RFID readers.

**2.2 Detailed Description**This system uses different blocks. Information goes from the tag to the arduino, from the arduino to the server, from the server to the database, from the database to the server and from the server to the arduino. Here is described what happens at which block and why.

**Arduino**When the arduino first makes connection over the serial port it sends an acknowledgement to the server to make sure it’s door ID is valid or else to add it’s door ID to the database.

When the tag is presented to the arduino, the arduino reads its UID. The arduino sends its door ID and an encrypted version of the UID. The UID and the door ID need to be sent encrypted, because otherwise, someone could intercept it and open doors without permission. The arduino now sends this secured message to the server.

**Server**When the server receives any message at all it will first respond to that message with an acknowledgement to let the arduino know it successfully received it request.

The server gets the secured message and decrypts it.Then the server hashes the UID. The server now compares the hashed UID with the hashed UID in the database and gets the authorization status of the key (as a boolean). The server encrypts the UID again, by XOR-ing the UID with the key. The server creates a message with the encrypted UID and the boolean value. This secured message is now send to the original arduino.

**Arduino**The arduino gets the secured message (with UID and boolean value) from the server and decrypts it. The arduino checks if the original sent UID is equal to the just received UID and handles in regards to the boolean value stored in the value key pair. True will result the door to open and False will give a negative feedback.

**2.3 Use**

When the card reader is out of use it will show a solid red light.

To use the card reader, first the user holds their personal key card the RFID reader. The RFID reader will then read out the user’s personal ID, where the server then compares this with the ID’s which are authorized to enter the area which lies behind the specific door.

If the card is successfully read then a signal will be sent which then outputs an audio; namely a single beep.

If the person is authorized to enter a certain area, this will be confirmed by a long beep and a green light. While this beep lasts the door is unlocked and can be opened, if the door is not opened after the sound has ended the door will lock itself again. If the is opened by the user this will lock again when the door is closed. When the door is closed the light will turn red again.

If the person is not authorized to enter a certain area, this will be confirmed by two short beeps; in this instance the light does not change. The door will not be unlocked.

In case of an emergency, for example an electricity fail, power generators will be activated to cover the power that is required in order to keep the system running without a problem. If there is a fire at the building,the system will automatically open all the doors so the people can leave the building safely and without further hassle.

In case of an emergency, for example an electricity fail, power generators will be activated to cover the power that is required in order to keep the system running without a problem. If there is a fire at the building,the system will automatically open all the doors so the people can leave the building safely and without further hassle.

**3. Evaluation**

**3.1 Overview**

Our approach to testing our system was by testing a prototype. We had different design requirements, and thus different ways of testing those different requirements. In the table below, one can see how we tested which requirement. There is one note to this evaluation, we tested this without the connection from the arduino to the door, because a door is not in our resources. We tested all of our requirements with the fact that if the arduino gives a sign, the door will open, and therefore the arduino represents the door.

Table with requirements and how they were tested:

|  |  |  |
| --- | --- | --- |
| **Design requirement** | **Target value** | **Test method** |
| The door doesn’t open for unpermitted tags | 1 (True) | Hold unpermitted tag for RFID and see what the response is from arduino |
| The door does open for permitted tag | 1 (True) | Hold permitted tag for RFID and see what the response is from arduino |
| The door locks itself if the user closes the door | 1 (True) | Could be tested if the arduino was connected to a door, but we are not on that stage yet, so we cannot test this. |
| All doors can be opened at once in case of emergency | 1 (True) | This couldn’t be tested because we haven’t implemented this option in our system. If it would have been implemented, it could have been tested by clicking on the button “Open all doors” in the GUI and see if the arduino responds positive. |
| All doors can be closed at once in case of emergency | 1 (True) | This couldn’t be tested because we haven’t implemented this option in our system. If it would have been implemented, it could have been tested by clicking on the button “Close all doors” in the GUI and see if the arduino responds positive. |
| Roles, persons and doors can be added or deleted to the database | 1 (True) | Add and delete persons, doors and roles via the GUI and see whether the database has changed positively. |
| Permissions for persons or roles can be added or deleted to the database | 1 (True) | Add and delete permissions for persons or roles via the GUI and see whether the database has changed. |
| The door remains unlocked for 15 seconds, if the door is not opened | 15 seconds | Scan permitted tag and see whether arduino gives a signal for opening, and 15 seconds later for closing. |
| Unlocks after 3 seconds | Soonest as possible | Scan the tag and time the time waiting for the door to be opened. |

**3.2 Prototype**

The prototype which is designed is based upon the eventual systems; like a béta-version. In this prototype the most important design requirements are to be tested. First of all the basic idea of using key-cards and a scanner gets tested and the results can be seen by the flashing lights we also include in our system. So if a card is accepted by the scanner the green light is on and when a card is denied the red lamp lights up. The flashing lights are also followed by a buzzing sound for each situation. Our prototype contains an arduino uno mechanism, an RC522 RFID scanner, 2 flashing lights and a buzzer, 2 key-cards and 2 tags.

**3.3 Testing and Results**

The door doesn’t open for unpermitted tags:  
This requirement was most important of all requirements, because if the doors would have opened with unpermitted tags, our system would not be valid. Intruders could then easily get in rooms of clients, or clients could end up in the wrong room.

This requirement was evaluated by holding an unpermitted tag for the RFID-scanner and seeing what the response was from the arduino.   
The response we got from the arduino were two short beeps, meaning that the door would not open. This is a good thing, because we can now say that our system meets the most important requirement.

The door does open for permitted tag:

This requirement was important, because if the door would not open while scanning a permitted tag, clients or nurses or other person who would have been authorized, cannot get in their room, which obviously is a big problem.   
This requirement was evaluated by holding a permitted tag for the RFID-scanner and see what the response is from arduino.  
The response from the arduino was a long beep, which means that the door would open. This is again a good response, and this is also meeting the requirements.

The door locks itself if the user closes the door:

This requirement was important, because if the door would not lock if someone closes it, everyone could enter that door.  
However, this requirement could not be tested, because we have not connected the arduino to a door yet. So for this requirement, there are no results, which is a point we will mention in the paragraph “Next steps”.

All doors can be opened at once in case of emergency:

This requirement was important because when an emergency occurs, clients should be able to get evacuated to get into safety. If this could not happen, a lot of delay occurs, which could lead to dangerous situations.   
This couldn’t be tested because we haven’t implemented this option in our system. If it would have been implemented, it could have been tested by clicking on the button “Open all doors” in the GUI and see if the arduino responds positive. But right now, we have no results, because we have not tested it. This is a point which will be mentioned in the paragraphs “Assessment” and the “Next step”.

All doors can be closed at once in case of emergency:

This requirement was of importance because if some sort of emergency occurs which leads to the need of closing all doors, it should be doable. Let’s say someone is missing or if there has been an emergency which wanted to open all doors, it would be nice if all doors can be closed again.   
However, this couldn’t be tested, because we haven’t implemented this option in our system. If it would have been implemented, it could have been tested by clicking on the button “Close all doors” in the GUI and see if the arduino responds positively. Again, right now we have no results, because we have not tested it. Therefore this is also a point which will be mentioned in the paragraphs “Assessment” and the “Next step”.

Roles, persons and doors can be added or deleted to the database:

This requirement was also very important, because if there are new clients, or clients are leaving, or there will arise new roles, or there will be an expansion of the building, those things have to be added to the database.   
This can be tested by adding and deleting persons, doors and roles via the GUI and see whether the database has changed positively.  
We tested this, and it worked. This is a good thing, because our system now meets this requirement.

Permissions for persons or roles can be added or deleted to the database:

It is important to be able to remove, add or change the permissions of a user through the database, because if a role changes its permissions or if a specific person needs to get in different rooms, his or her permissions need to be changed.

This was tested by adding and deleting permissions for persons or roles via the GUI and see whether the database has changed.

This was done, and everything worked. Again this is a good thing, because now our system meets this requirement.

The door remains unlocked for 15 seconds, if the door is not opened:

This requirement is important for the usage of the system by the client, it is based on making the usage of the system easier so the user will not have to scan the door for a second time if he/she does not manage to open the door right after the first scan.  
This was tested by scanning a permitted tag and see whether arduino gives a signal for opening, and 15 seconds later for closing. Again, it works, which is nice because we met another requirement with our system.

Unlocks after 3 seconds:

This is another requirement based on making the system easier to use, so instead of waiting for a long time for a door to open, 3 seconds is a good amount of time.

This can be tested by scanning the permitted keycard. If the arduino gives a green sign in 3 seconds that means that the system is working.

This worked, which means this requirement is also done with our system.

**3.4 Assessment**

Just like every other project, this project has its strengths and weaknesses. The best characteristics of this project is that it is simple to use and able to recognize which person has the right permissions to open a door, something that means that it also provides security. It also meets most of the design requirements so it is safe to say that it is a well-designed system. The requirements that it does not meet are the ones that have to do with the connection between the arduino and the door. This happens because we did not connect our system with a door to see the actual results, but it is safe to assume that there will not be any problems with it since the arduino works perfectly. Another weak spot of this system is the card. The card does not allow us to write any information on it something that could add even more security. By investing into better and more developed cards, the designer will be able to write the information that is needed in each card and therefore it will be even more difficult for someone to bypass this system. Also, in case of an emergency, we want to have an option in our system that will allow all the doors to open in once. This option is currently not functional, something that should be ready before the system is ready for use.

**3.5 Next steps**

Since the project is completed, tested and working, it is safe for the companies to go ahead and use it. If there is something that can be improved in the project, this would be to make the buttons “Close all doors” and “Open all doors” work. This is something that the system designer will work on in order to provide the company with a well structured and fully functional security system. Also, since this project is in the project-stages, when a company decides to order this security system, the designer should make sure that a connection between the arduino and the door will be made. What will be succeeded by that will be that the door will open whenever the arduino gives a green light and it will remain closed whenever it gives a red light.

**Appendices**

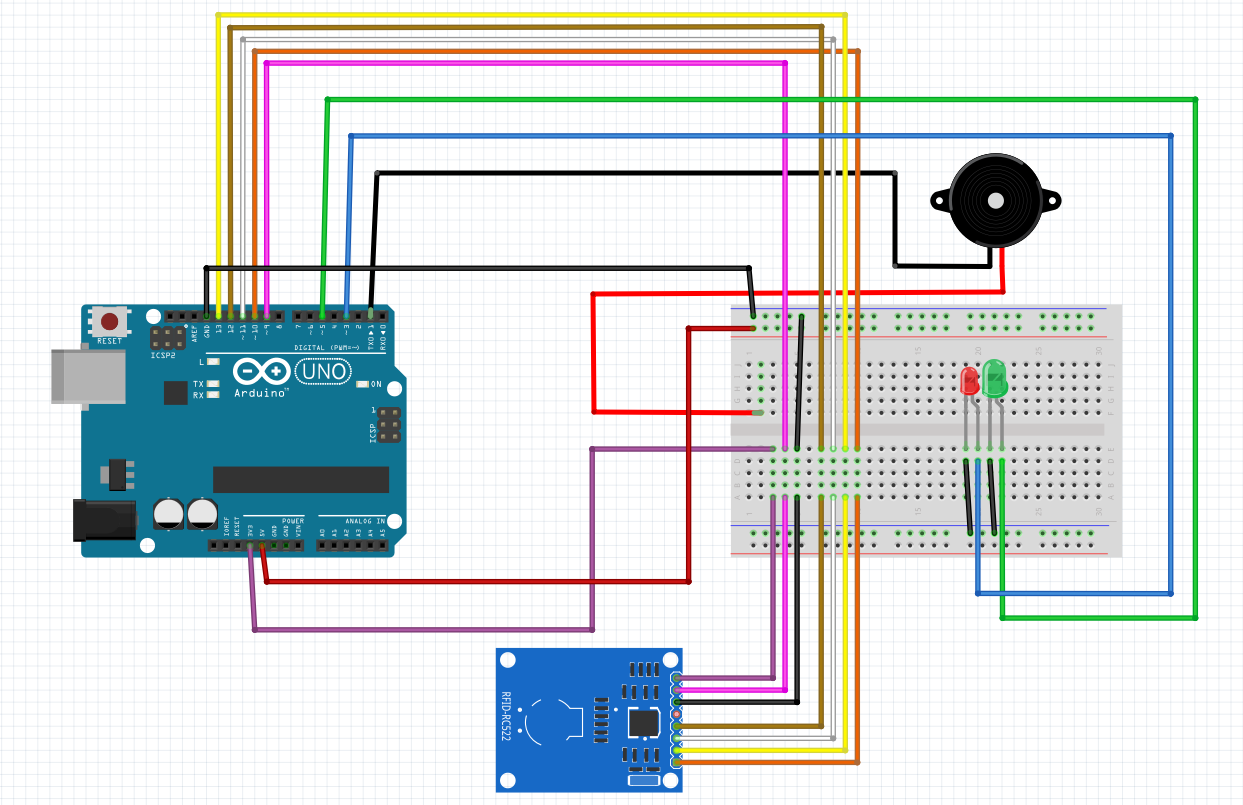
There are a couple of appendices. There is a list of appendices, a table with results and an image of the connections from the arduino.

List of the materials that we used:

* 2 Keycards
* 2 tags
* 1 arduino uno
* 1 RC522-RFID scanner
* 1 green/1 red light
* 1 buzzer

Results we got from the testing of the requirements “Roles, persons and doors can be added or deleted to the database” and “Permissions for persons or roles can be added or deleted to the database”.

|  |  |  |
| --- | --- | --- |
| **Action** | **Target value** | **Value** |
| Add a person to the database via the GUI | Accomplished | Accomplished |
| Add a role to the database via the GUI | Accomplished | Accomplished |
| Add a door to the database via the GUI | Accomplished | Accomplished |
| Add a permission for a person to the database via the GUI | Accomplished | Accomplished |
| Add a permission for a role to the database via the GUI | Accomplished | Accomplished |
| Delete a person from the database via the GUI | Accomplished | Accomplished |
| Delete a role from the database via the GUI | Accomplished | Accomplished |
| Delete a door from the database via the GUI | Accomplished | Accomplished |
| Delete a permission of a person from the database via the GUI | Accomplished | Accomplished |
| Delete a permission for a role from the database via the GUI | Accomplished | Accomplished |

Image of the connection from the elements (buzzer, RFID scanner and leds) to the arduino: 

**References**

* Gliffy.com
* Fritzing.org