**DESIGN PORTFOLIO**

**Alarm System**

**by:**

**R.A. van Wyk 24905798**

**J.J Fick 25867350**

**M.B Odendaal 25340638**

**PROJECT SHIELD**

Submitted in pursuit of the degree

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER AND ELECTRONIC ENGINEERING**

**North-West -University Potchefstroom Campus**

Supervisor: Johann Holm

Potchefstroom

2017

Contents

[1. System Operational Requirements 2](#_Toc494897346)

[1.2 Functional Analysis – Operational Level Architecture and Behaviour 2](#_Toc494897347)

[1.2 Physical Requirements: 3](#_Toc494897348)

[1.3 Interface requirements: 3](#_Toc494897349)

[1.4 Additional Requirements: 4](#_Toc494897350)

[1.4.1 Environmental Requirements: 4](#_Toc494897351)

[1.4.2 Safety requirements 4](#_Toc494897352)

[1.4.4 Usability Requirements: 5](#_Toc494897353)

[2. Excel Project Management Documents 5](#_Toc494897354)

[2.1 Combined Project Documents 5](#_Toc494897355)

[2.2 Risk Mitigation Register 13](#_Toc494897356)

[2.3 Minutes of Meetings 14](#_Toc494897357)

[2.4. Ruben van Wyk 15](#_Toc494897358)

[2.4.1 Management Documents 15](#_Toc494897359)

[2.4.2 Ruben van Wyk Experience Report 16](#_Toc494897360)

[2.5 Johan Fick 16](#_Toc494897361)

[2.6 Burger Odendaal 16](#_Toc494897362)

[3.1. Ruben van Wyk Engineering Report 16](#_Toc494897363)

[3.1.1 Scope of the work 16](#_Toc494897364)

[3.1.2 Assumptions and Constraints 17](#_Toc494897365)

[3.1.3 Specific Engineering Methodology 17](#_Toc494897366)

[3.1.4 Schematic structure 17](#_Toc494897367)

[3.1.5 Physical Measurements 17](#_Toc494897368)

[3.1.6 Health and safety considerations 17](#_Toc494897369)

[3.1.7 Additional Productivity Tools 17](#_Toc494897370)

[3.2. Johan Fick Engineering Report 18](#_Toc494897371)

[3.2.1 Scope of work 18](#_Toc494897372)

[3.2.2 Assumptions and Constraints 18](#_Toc494897373)

[3.2.3 Discipline- Specific Engineering Methodology 18](#_Toc494897374)

[3.2.4 Simulations/ Flowcharts /State diagrams 19](#_Toc494897375)

[3.2.5 Physical Tests/ Measurements (Explanation of test Results) 19](#_Toc494897376)

[3.2.6 Health and safety considerations (Full Life cycle) 19](#_Toc494897377)

[3.2.7 Discussion of tools used to enhance productivity 19](#_Toc494897378)

[3.3. Burger Odendaal Engineering Report 20](#_Toc494897379)

[3.3.1 Scope of work 20](#_Toc494897380)

[3.3.2 Assumptions and Constraints 21](#_Toc494897381)

[3.3.3 Discipline- Specific Engineering Methodology 21](#_Toc494897382)

[3.3.4 Simulations/ Flowcharts /State diagrams 21](#_Toc494897383)

[3.3.5 Physical Tests/ Measurements (Explanation of test Results) 21](#_Toc494897384)

[3.3.6 Health and safety considerations (Full Life cycle) 21](#_Toc494897385)

[3.3.7 Discussion of tools used to enhance productivity 21](#_Toc494897386)

[4. Sub-system Specification Documents 22](#_Toc494897387)

[4.1 Hardware system specifications (Ruben) 22](#_Toc494897388)

[4.1.1 Sub-system Functional Flow 22](#_Toc494897389)

[4.1.2 Interface Definitions 23](#_Toc494897390)

[4.2 Specifications Johan Fick 23](#_Toc494897391)

[4.2.1 Sub-system Functional Flow 23](#_Toc494897392)

[4.2.2 Interface Definitions 23](#_Toc494897393)

[4.2 Specifications Burger Odendaal 23](#_Toc494897394)

[4.2.1 Sub-system Functional Flow 23](#_Toc494897395)

[4.2.2 Interface Definitions 23](#_Toc494897396)

[5. Design documentation 24](#_Toc494897397)

[5.1 System design documentation 24](#_Toc494897398)

[5.1.1 Final System Functional Definition Final requirements 24](#_Toc494897399)

[5.1.2 System Concept Drawings 24](#_Toc494897400)

[5.1.3 System Interface Definitions 25](#_Toc494897401)

[5.1.4 System Integration testing 26](#_Toc494897402)

[5.2 Ruben an Wyk 28](#_Toc494897403)

[5.2.1 28](#_Toc494897404)

[5.2.2 28](#_Toc494897405)

[5.2.3 28](#_Toc494897406)

[5.2.4 28](#_Toc494897407)

[5.2.5 28](#_Toc494897408)

[5.2.6 Design Implementation 28](#_Toc494897409)

[5.2.7 System Test and Evaluation 28](#_Toc494897410)

[5.3 Johan Fick 28](#_Toc494897411)

[5.3.1 28](#_Toc494897412)

[5.3.2 28](#_Toc494897413)

[5.3.3 28](#_Toc494897414)

[5.3.4 28](#_Toc494897415)

[5.3.5 28](#_Toc494897416)

[5.3.6 Design Implementation 28](#_Toc494897417)

[5.3.7 System Test and Evaluation 28](#_Toc494897418)

[5.4 Burger Odendaal 28](#_Toc494897419)

[5.4.1 28](#_Toc494897420)

[5.4.2 28](#_Toc494897421)

[5.4.3 28](#_Toc494897422)

[5.4.4 28](#_Toc494897423)

[5.4.5 28](#_Toc494897424)

[5.4.6 Design Implementation 28](#_Toc494897425)

[5.4.7 System Test and Evaluation 28](#_Toc494897426)

# List of tables

[Table 1:Matrix IDE comparison table 11](#_Toc494897427)

[Table 2:Typical protocols to consider 14](#_Toc494897428)

[Table 3:Risk Mitigation Register 16](#_Toc494897429)

[Table 4:Member A schedule 19](#_Toc494897430)

[Table 5: Passing conditions table 29](#_Toc494897431)

# List of Figures

[Figure 1:System operational architecture. 6](#_Toc494897454)

[Figure 2:System operational flow. 6](#_Toc494897455)

[Figure 3:Security System Definition 9](#_Toc494897456)

[Figure 4:Raspberry Pi 3 Model B 10](#_Toc494897457)

[Figure 5: Broad program categories.: 11](#_Toc494897458)

[Figure 6:Integrated development environment 12](#_Toc494897459)

[Figure 7:GUI login screen 13](#_Toc494897460)

[Figure 8:Typical GUI elements which will be used 14](#_Toc494897461)

[Figure 9:Basic communication system concept 15](file:///C:\Users\mb253\Desktop\2017%20semester%20twee\Third%20year%20project\ELO5_burger.docx#_Toc494897462)

[Figure 10:Unit breakdown schematic 18](#_Toc494897463)

[Figure 11: Uml Diagram 22](#_Toc494897464)

[Figure 12: Subsystem functional analysis 23](#_Toc494897465)

[Figure 13: Subsystem functional architecture. 23](#_Toc494897466)

[Figure 14: Total functional flow of the system 25](#_Toc494897467)

[Figure 15:Functional flow of the hardware subsystem 26](#_Toc494897468)

[Figure 16:Design Drawing A 27](#_Toc494897469)

[Figure 17:Design Drawing B 28](#_Toc494897470)

# System Operational Requirements

## Functional Analysis – Operational Level Architecture and Behaviour

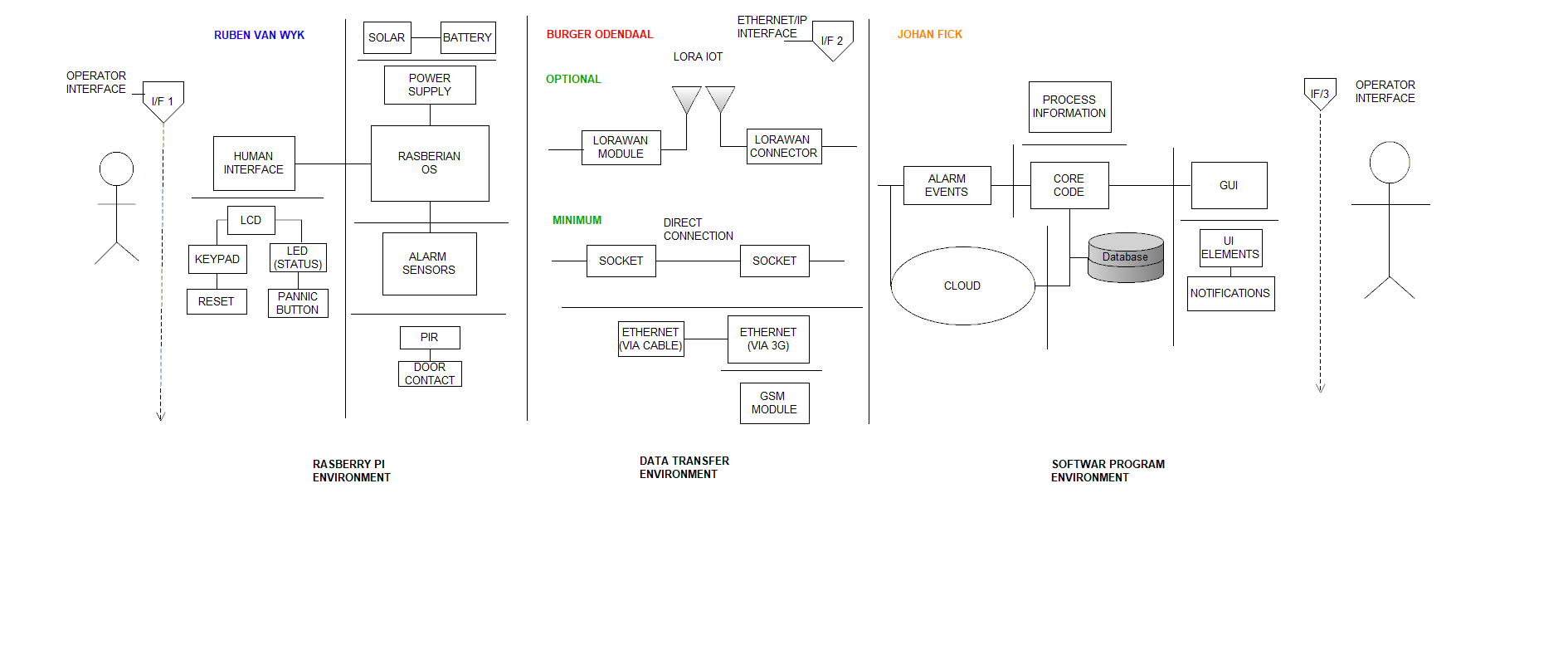


Figure :System operational architecture.

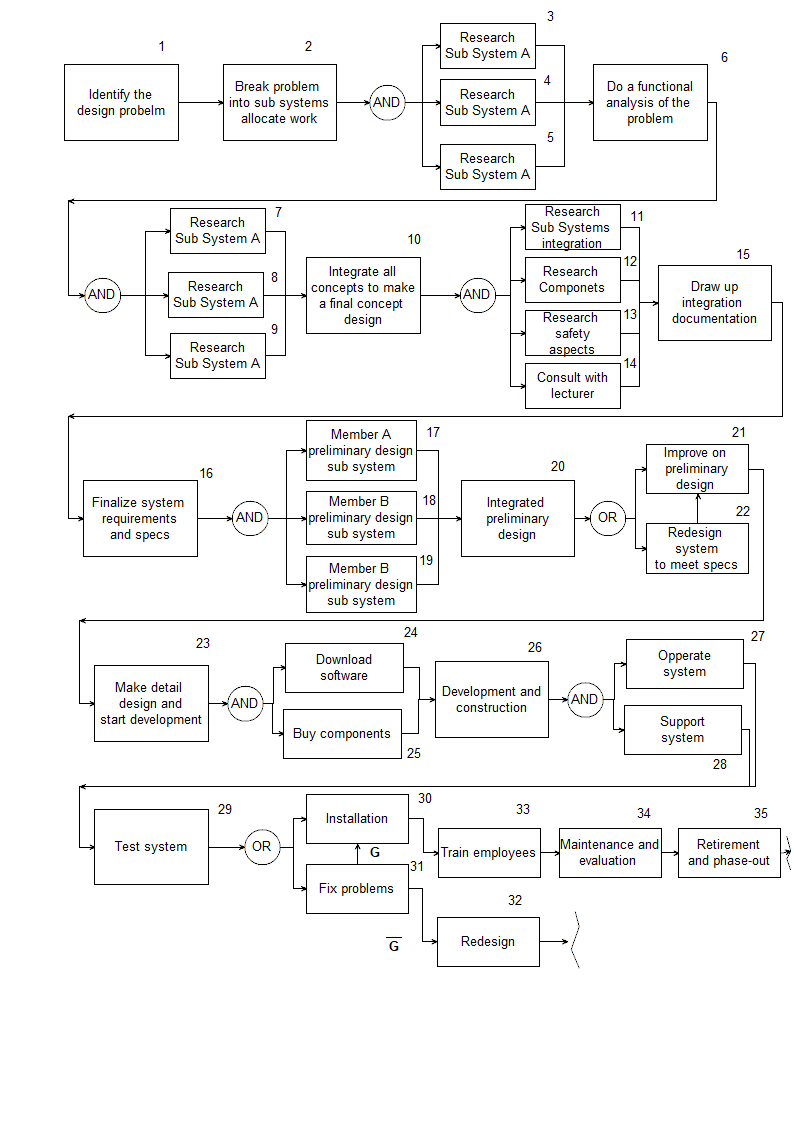


Figure :System operational flow.

## 1.2 Physical Requirements:

* The different sensors, case and all other components of the system must be IP54 rated.
* The alarm panel must be small enough to be fitted to a wall. (Double sided tape will be used to avoid the need of drilling)
* The panel must be designed in such a way that it will allow the cables easily plugin in to the box.
* The panel must be waterproof to ensure that no damage will be caused due to accidental water spillage or rain.

## 1.3 Interface requirements:

* Interfacing with the alarm panel must be easy to understand and to use. Different interfacing elements such as buttons, LED’s and sound can be used.
* All the applicable software and circuit diagrams must be designed, except for the components and the case of the alarm panel.
* The Raspberry Pi 3 will be used for the processing at the panel end.
* The database will be implemented on a computer. Later on the database will be implemented in the cloud. The database will be connected to the security monitoring software where it will provide additional information regarding the specific alarm system.
* The PIR or door contacts will be used to monitor for any movement and will generate a signal which will be sent to the alarm panel and in return to the computer.
* The panel will be power by a POWER BLOCK. The POWER BLOCK will provide a 12 V @ 6 W supply current.
* The alarm panel is operated at 3.3 V but the different components are operating at 12 V. Some type of step down is needed from the 12 V to single board computer.
* Ethernet will be used to transfer data between the SBC and the PC. A 3G modem can be used to achieve wireless communication. IoT and Lora WAN will be implemented if time permits.
* The power supplied to the panel will be connected using screw in wires.
* The height at which the panel will be placed must be such that it is easily accessible by the user. (Not too high or too low)

The panel shouldn’t be installed over existing wires, water, fire hazard or any other interference.

## 1.4 Additional Requirements:

### 1.4.1 Environmental Requirements:

* The different sensors, case and all other components of the system must be IP54 rated.
* ESD must not affect the system to such an extent that it cannot function correctly.
* The panel and other components must be tamper proof to an extent.
* The materials used must be robust and able to endure different environment challenges.

### 1.4.2 Safety requirements

* The system will be grounded to ensure no electrical shocks can occur to an end user
* Wires must be concealed at all time.
* Small children must not be able to access the panel or any sensors.
* The panel will be concealed to ensure that people will not be able to tamper with the internal electronics of the panel.
* Grounding the system will help ensure that the end user will not receive an electrical shock from the device.

#### 1.4.3 Legislative Requirements (SAIDSA bylaw 25)

* Control equipment
  + The time it takes to disarm the system must not be more than 30 sec.
  + Installation of the system must be within 1.5mm from the ceiling
* Signalling equipment
  + A protected area must be used to place the singling equipment in.
  + System or wires may not be placed where it may influence telephone lines.
* Maintenance
  + Check the cables for damage.
  + Check that the sensors are still functional.
  + Check alarm panel for damage.
  + Check that the transmitter light is on.

### 1.4.4 Usability Requirements:

* System must be easy to use with little additional training.
* The control box will be mounted with double sided tape and will be provided.

# Excel Project Management Documents

Please refer to linked excel document below:



## 2.1 Combined Project Documents

#### Member a – ruben van wyk

The purpose of this report to determine the needs and requirements of the hardware department of the alarm system and the human interface that is attached to it. Details of the design will be discussed at a later when the scope of the project is more fully understood.

The hardware will rely on a specific alarm unit that will be sold and should be connected to alarm sensors and door security sensors as well as the network interface (via GSM module). The overall system is as follows:

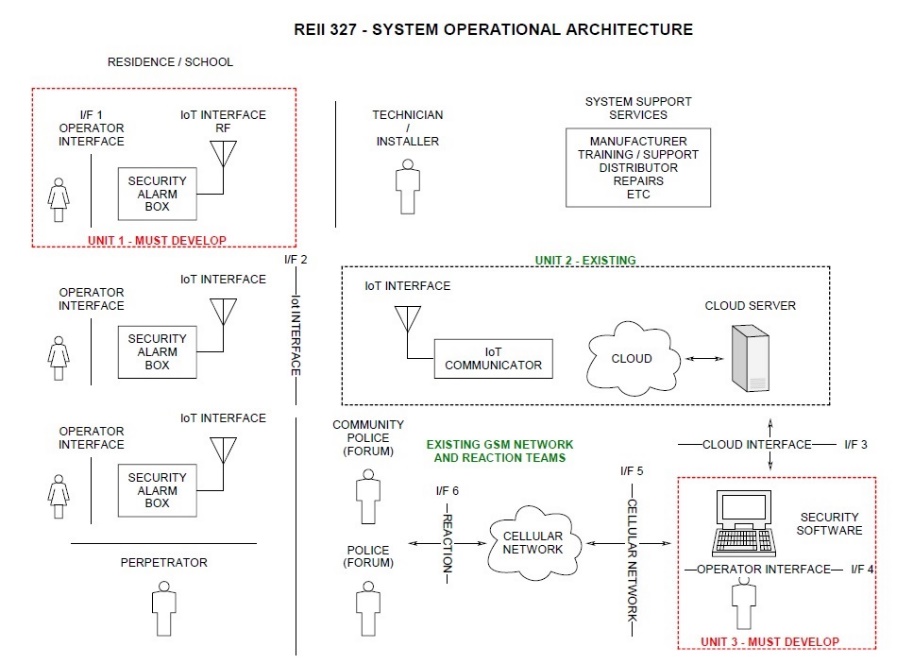


Figure :Security System Definition

Therefore, this report will analyse the left-most part of the schematic, occupying the human interaction and hardware components – everything up the ‘Internet of Things interface’ or I/F2.

##### Functional Requirements

First, let us identify the physical hardware system:

* Alarm module.
* Laser sensory modules.
* Door contact modules.

The Alarm module is the primary piece of equipment and may contain the following:

* Raspberry Pi chipset.
* LCD Panel.
* LED Array.
* Physical Buttons.
* GSM/Network module.
* Battery and battery control module.
* Physical casing.

A full component list will be included later on in this document.

##### Chipset Specifications

The Raspberry Pi is a versatile computing platform that allows for practical embedded programming and ambiguity on a tremendous scale. The device has powerful components coupled with an industry-low price which allows for radical implementations. It has continually led to the advancement and innovation of otherwise novice ideas and gadgeteering enthusiasm into full-blown profitable projects.

The newest version is the Raspberry Pi 3, and we will specifically be looking at the Model B version.

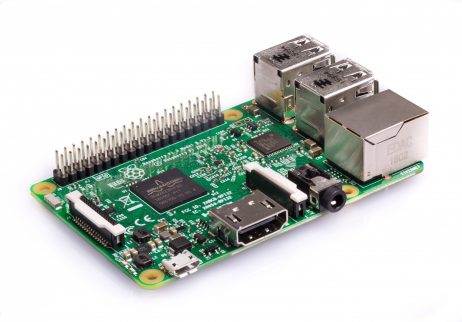


Figure :Raspberry Pi 3 Model B

Features & Benefits of the Pi 3  
•Broadcom BCM2837 chipset running at 1.2 GHz  
•64-bit quad-core ARM Cortex-A53  
•802.11 b/g/n Wireless LAN  
•Bluetooth 4.1 (Classic & Low Energy)  
•Dual core Videocore IV® Multimedia co-processor  
•1 GB LPDDR2 memory  
•Supports all the latest ARM GNU/Linux distributions and Windows 10 IoT  
•microUSB connector for 2.5 A power supply  
•1 x 10/100 Ethernet port  
•1 x HDMI video/audio connector  
•1 x RCA video/audio connector  
•1 x CSI camera connector  
•4 x USB 2.0 ports  
•40 GPIO pins  
•Chip antenna  
•DSI display connector  
•microSD card slot  
•Dimensions: 85 x 56 x 17 mm

##### Programming Model

The Pi will act as the primary controller of the on-site system. We can now consider a programming structure that will allow for further investigation as the project progresses.

Figure : Broad program categories.:

##### Category descriptions:

* **System state display:** Output pins connected to some sort of display device (Monitor, LCD panel or LED array) that can show the user whether the alarm system is enabled or disabled and precisely which input is tracking motion.
* **Operator configuration input:** The operator gets to configure the password of the system, use emergency functionality and disable or enable the alarm.
* **Sensory monitor inputs:** Constantly monitor input from the passive alarm sensory elements to check for motion in the setup area. The inflow of this information is most essential to the system and requires top priority.
* **Network information exchange:** Follow a premeditated protocol to report information of movement or alarm events to the servers and alarm police or the server base station.

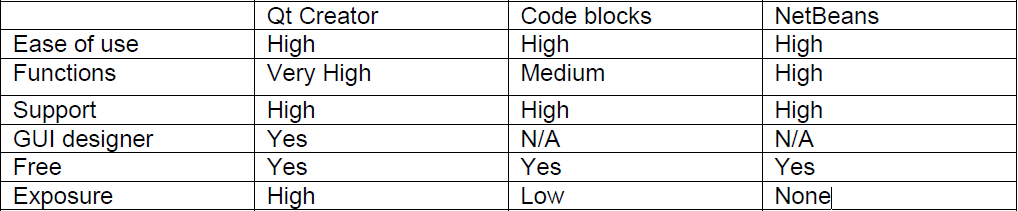
#### Member b – Johan fick

It is very important to choose the correct program which will be used to develop the security software. A wide range of software development programs was considered. The top 3 was used to do a more in depth study on.

The top 3 IDE’s:

* Qt Creator.
* Code blocks.
* NetBeans.

Table :Matrix IDE comparison table



By using the results from the table above, a conclusion could be drawn. Qt creator will be used since it fits the criteria best.

One reason for using this program is due to the level of exposure to it. In a previous module this program was used to quite an extent. Therefore, it will be relatively easy to use and build upon knowledge gained from previous work.

A main driving force is the flexibility of the GUI development. Qt is very powerful when it comes to designing an application interface. This will allow the developer to create a wide range of GUI’s, which can be tested to determine the best suited interface.

There is also a very big community of Qt users. Which means that there are a lot of resources and example programs to choose from.

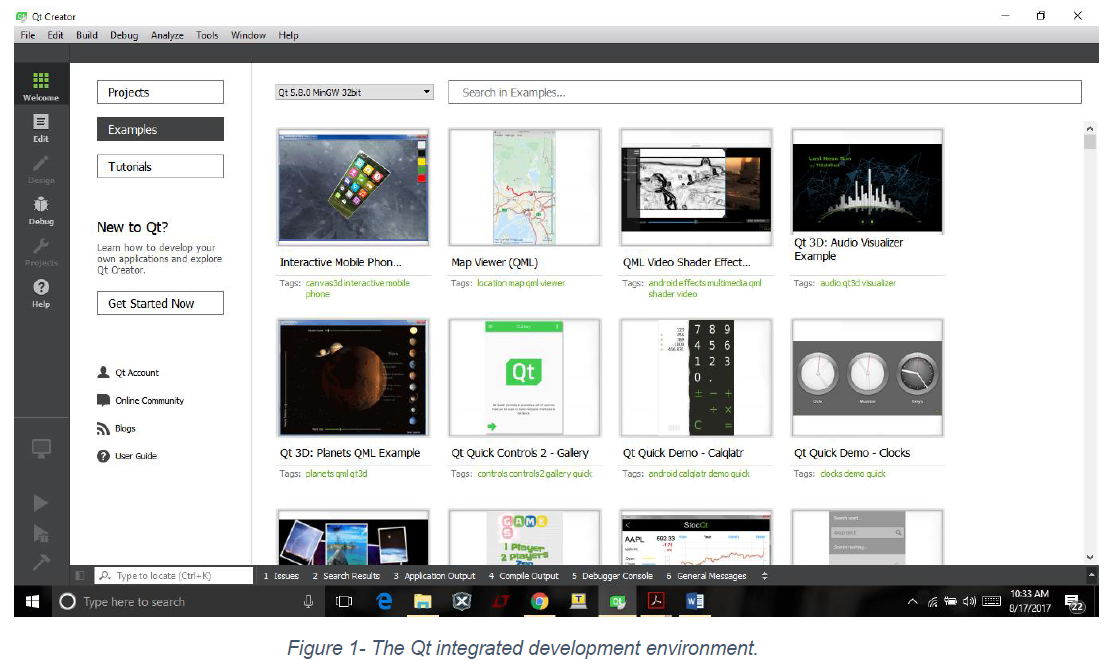


Figure :Integrated development environment

##### Designing the UI.

One of the biggest elements in the development of the security software will be the user interface. This will convey all information regarding the security system to the office. This will allow the security officer to easily track the status of the system.

* **Keep the interface simple.** Avoid using any unnecessary elements in the UI. All labels and interface should be cleared and well labeled.
* **Create consistency and use common UI elements.** Using elements that are standard and consistency. Will allow the user to easily navigate through the interface with ease.
* **Be purposeful in page layout.** When design the layout of the interface place items in a logic manner. This will draw the attention of the user to the most important parts.
* **Strategically use color and texture.** Color is a very good way to draw attention. It can also be used as a status of some objects.
* **Make sure that the system communicates what’s happening.** Assure that the program is conveying the correct information to the user through the UI. Don’t include any unnecessary information.

Refer to figure 2 and 3. These figure display the typical GUI format which will be used for the program. The elements contained in the GUI are very easy to understand and navigate with.

Every time the program launches the employee of the company will have to login in via the login menu. Once access is gained to the program the user will be able to switch between different tabs. Each of these tabs conveying information specific to the tab.

When referring to figure 3. There are some key elements which is crucial to the program design.

A **data basis** will be used to store all the information regarding the alarm systems. The information stored will be device specific. This will allow the user to easily keep track of the status of all devices.

**Emergency protocol**. Each of the different protocols will have different affects. For example, protocol 1 – is typically a false alarm. Therefore, when this alarm occurs the program will log the event but no immediate actions will be executed. Protocol 2 – will be activated when a caution alarm is activated. The owner of this system will be informed requiring conformation before sending a unit. Protocol 3 – will be activated when a high likelihood of a burglary is occurring. A unit will have informed and dispatched to the scene.

The elements seen in these GUI’s aren’t fixed. Therefore, change can be expected. These elements will however for part of the final GUI design.



Figure :GUI login screen

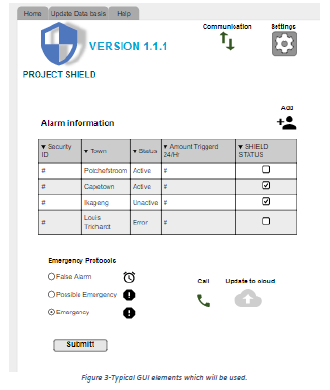


Figure :Typical GUI elements which will be used

#### Member c – burger odendaal

In order to effectively implement reliable communication between the alarm system hardware (Raspberry pi -MCU), control station(PC) and the cloud server the communication system has been divided into three operative interfaces. From the figure visually depicts the interfaces defined as well as two protocols needed.

Table :Typical protocols to consider

|  |  |
| --- | --- |
| **Interface** | **Description** |
| IF1 | MCU send to **GSM module**, receive from **GSM Module** |
| IF2 | **GSM module** send to **MCU**, receive from **MCU** |
| IF3 | GSM module send to **PC**, receive from **PC** |
| IF4 | PC send to **GSM modal**, receive from **GSM modal** |

Figure :Basic communication system concept

RASBERRI PI

GSM Module

PC

IF1

IF3

IF2

IF4

Communications protocal ‘A’

Communications protocal ‘B’

According to figure one two different communications protocols needs to be selected. Protocol ‘A’ for communication between interface 1 and interface 2. Protocol ‘B’ is responsible for communication between interface 3 and interface 4. It is obvious that protocol A and protocol B cannot be the same, since protocol ‘A’ is responsible for establishing communication between hardware at interface 1 and an eventual software response at interface 2. Thus the next natural step is selecting or designing the best communication protocols between each interface pair.

##### Selecting Protocol ‘A’

Considering that the Raspberry pi MCU and the GSM module can both be considered to be embedded devices, a suitable embedded communications protocol need to be selected. Since the Raspberry pi is known to be compatible with multiple communication protocols, the first step is listing all the embedded communication protocols that the standard GSM module comes with. The limitation set to only use serial communication protocols was decided on the grounds that extra hardware will be needed for non-serial communication. Before comparing and selecting an appropriate protocol some information about common serial communication protocols is considered.

##### Protocol background:

Serial is a canopy term referring to all that is time-division-multiplexed. It means that the data sent is spread over time, bit by bit most of the time. [1]

###### UART

UART stands for Universal Asynchronous Receiver Transmitter, UART is well known and one of the most used serial communication protocols. Most MCU have a UART peripheral. It implements a single data line for transmitting and another for receiving data. Thus two lines in total a TX (Transmit line) pin and a RX (receive line). Most frequently 8-bit data is transferred. This consists of a start and stop bit and 8 data bits. The start bit is low level [1].

The low level start bit and high level stop bit mean that there's always a high to low transition to start the communication. This is the main idea behind why UART is used so extensively. There is no voltage level responsible for defining data elements, so 3.3 V or 5 V is acceptable, depending on the operating voltage of the target UART. MCU’s that connected have to agree on the bit-rate. Since only the transition from high has to be synchronized.

For extensive distance communication the UART is not very dependable, that's why it's adapted to a higher voltage. This is where RS323 comes into play [2]. The data format remains the same, RS323 only modifies the voltage level. However, there is more to this, which are not relevant in this concept design. Different line coding schemes comes into play here. The timing dependence is one of the great disadvantages of UART, and the answer is USART. (Universal Synchronous/Asynchronous Receiver Transmitter). In synchronous communication not only data, but also a clock is transmitted. With each bit a clock pulse informs the receiver it should latch the bit. Synchronous protocols either need a higher bandwidth.

###### SPI

SPI (Serial Peripheral Interface) is another very simple serial protocol. A master sends a clock signal, and upon each clock pulse it shifts one bit out to the slave, and one bit in, coming from the slave. Signal names are therefore SCK for clock, MOSI for Master Out Slave In, and MISO for Master In Slave Out. By using SS (Slave Select) signals the master can control more than 1 slave on the bus. There are two ways to connect multiple slave devices to one master, one is mentioned above i.e. using slave select, and other is daisy chaining, it uses fewer hardware pins (select lines), but software gets complicated.

# 2.2 Risk Mitigation Register

Table :Risk Mitigation Register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk No** | **Date** | **Description of Issue / Risk** | **Severity** | **Impact** |
| *1* | *07-Sep* | There is a vast amount of uncertainty in the group towards the end goals of the project and what is expected of us, therefore we have not made significant progress because we have no clear goal which we are working towards. | *High* | *High. Causes disruption in work process.* |
| *2* | *07-Sep* | Due to busy schedule and sheer amount of work we have not been able to fulfil our good intentions and goals and have not been interacting together as a team as we should have. | *Low* | *Causes lag in work flow.* |
| *3* | *07-Sep* | We have not found a clear time to work on our projects in the week. Proposed: Tuesdays 08h00 to 11h00. | *Low* | *Miscommunication and lack of vision for project.* |
| *4* | *07-Sep* | Two of the group members have Linear Algebra which fall on the same time slot as our Project classes | *Medium* | *Causes a break in communication between lecturer and members* |
| *5* | *07-Sep* | Uncertainty about human interface – should a physical alarm module be constructed with full LED screen and buttons or should the Pi be connected to a monitor and mouse setup? Must the Pi use embedded coding or can it use software on top of existing Linux infrastructure? | *High* | *Disrupts hardware design process* |
| *6* | *07-Sep* | We have not yet as a group completed our portfolio, as set out to be finished nearly a week ago. | *Medium* | *Documentation lags behind* |
| *7* | *07-Sep* | *Missing deadlines* | *High* | *Failure to complete the project in time* |
| *8* | *07-Sep* | *Unseen problems in design* | *Meduim* | *Desing flaws* |
| *9* | *07-Sep* | *Portfolio guidelines are needed* | *Low-Meduim* | *Flawed portfolio* |

# 2.3 Minutes of Meetings

|  |  |
| --- | --- |
| **Date** | 15-Aug |
| **Time** | 11:00 |
| **Chairman** | *Ruben van Wyk* |
| **Attendance list:** | *Ruben van Wyk* |
|  | *Johan Fick* |
|  | *Burger Odendaal* |
|  |  |
| **Agenda** |  |
|          Concept design |  |
|          Excel spreadsheet |  |
|          Work allocation |  |
|          Product implementation | |
|          Portfolio: intensive planning | |
|          RF Flow |  |
|          Datasheets must be inserted into portfolio | |
|          NB Design verification through testing steps | |
|          Reconsider all possible options | |
|          Discover the cost of different options for example: | |
|          LEDs, LCD panel, 7-segment etc. | |
|          Radio-control remote, button, more complex interface. | |
|          Find out about monitor-setup vs alarm-button configuration from Prof Holm. | |
|          Fact check die information. | |
|          Identify all pieces with research must be conducted on. | |
|          FIAP Concept Design. | |
|  |  |
| **Actions** |  |
| * RF Flow |  |
| * Work on concept design according to your part. | |
| * Portfolio met planning, begin met RF Flow. | |
| * Do a functional analysis of each step. | |
| * Important for now: complete functional analysis and flows for every part. | |
| * Each member must identify and research different options for their part such as the IDE for example which they shall be using. | |
| * Set goal dates for each piece. | |
| * NB Concept design schematics.  2.4. Ruben van Wyk2.4.1 Management Documents It is this member’s responsibility to develop the Raspberry Pi embedded software and and how the end user and perpetrator interfaces will work.    Figure :Unit breakdown schematic | |

More complete documentation about management will only be available once more of the development of the Raspberry Pi and its associated environment has taken place and choices in development can be more fully motivated.

#### Member schedule

Table :Member A schedule

|  |  |  |
| --- | --- | --- |
| **Dates** | **Description** | **Milestone goal** |
| Prior to 16 September | Completion of analysis and planning | 30% |
| 17 - 23 September | Raspberry Pi setup and IDLE initisilisation | 30% |
| 24 - 30 September | GPIO coding for sensory system | 40% |
| 1 - 7 October | Project week - intensive hardware coding | 60% |
| 8 - 14 October | Completion of perpetrator interface IF2 | 70% |
| 15 - 21 October | Coding and testing of operator interface IF1. Documentation finalisation. | 80% |
| 22 - 27 October | Integration with network interface IoT.Documentation finalisation. | 90% |
| 29 Oct - 4 Nov | Integration with network interface IoT.Documentation finalisation. | 100% |
| 5 - 11 November | Total integration and testing.Documentation finalisation. | 100% |
| 12 - 17 November | Hand in project. | 100% |

### 2.4.2 Ruben van Wyk Experience Report

The design process thus far has been very challenging. This is our first real interaction with the Engineering Process. I think if more reasonable weekly documentation goals could have been set from the start of the semester things would have run more smoothly because now we have been overloaded with masses of work regarding the Project in the two weeks before the recess.

Also I feel there is still a large gap in understanding as to why the engineering process is so cumbersome and large, over documentation seems to be a very big problem at the moment because we are making uninformed predictions about future processes in the design process which we cannot foresee at this time.

At least there currently seems to be a spirit of fervence in our group at the moment. Each member has clearly defined goals and a greater understanding of the workings of the project and their part and how it will relate to the bigger picture and also what channels to follow for development of the project.

## 2.5 Johan Fick

2.5.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Dates** | **Description** | **Milestone goal** | |
| Prior to 16 September | Completion of analysis and planning | 30% |  |
| 17 - 23 September | Development software installation and research | 30% |  |
| 24 - 30 September | Databasis design and GUI design | 40% |  |
| 1 - 7 October | Coding- Main program code to be written | 60% |  |
| 8 - 14 October | Networking- Accessing the webpage and verifying connectivity . | 70% |  |
| 15 - 21 October | Testing of the Databasis, GUI and networking code. | 80% |  |
| 22 - 27 October | Integration with the other members. | 90% |  |
| 29 Oct - 4 Nov | Integration with the other members. (Final). Documentation updating. | 100% |  |
| 5 - 11 November | Total integration and testing.Documentation finalisation. | 100% |  |
| 12 - 17 November | Hand in project. | 100% |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Progress Tracking** |  |  |  |  |
|  |  | **Project Status** | | |
| **Dates** | **% Complete** | **Green** | **Amber** | **Red** |
| Prior to 16 September | 20-25 |  |  | Red |
| 17 - 23 September | 20-25 |  |  | Red |
| 24 - 30 September | 20-25 |  |  | Red |
| 1 - 7 October | 30-35 |  |  | Red |
| 8 - 14 October | 35-40 |  | Amber |  |
| 15 - 21 October | 50-60 |  | Amber |  |
| 22 - 27 October |  |  |  |  |
| 29 Oct - 4 Nov |  |  |  |  |
| 5 - 11 November |  |  |  |  |
| 12 - 17 November |  |  |  |  |

2.5.2

The 3 year design process has been cumbersome. Due to the fact that there are so many aspects which need to be considered and done in the engineering process. Some aspects seemed vague and therefore were left to the end. This caused a build-up in work load. It could help if there were some better objective guidelines. However this can be seen positively since when attempting another engineering design process, the skills learnt here can be applied to avoid this mistake.

One thing that is also cumbersome is the fact of working with software/hardware/concepts that I don’t have any experience in. This causes extra research and time needed to learn and solve mistakes which are accompanied with the above stated.

There is a positive vibe in the group. This seems to help since we are able to encourage each other and drive to progress of the project regardless of the mistakes made. My overall attitude towards the design process and project, is however very positive since each lesson learnt and mistake made is one step closer to becoming a qualified engineer.

## 2.6 Burger Odendaal

### 2.6.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Dates** | **Description** | **Milestone goal** | |
| Prior to 16 September | Completion of analysis and planning | 30% |  |
| 17 - 23 September | Research and language choice considerations and learning of software to be used | 30% |  |
| 24 - 30 September | Final protocol | 40% |  |
| 1 - 7 October | Start basic coding | 60% |  |
| 8 - 14 October | Have a working pipeline | 70% |  |
| 15 - 21 October | Testing of the network pipeline, and adding security | 80% |  |
| 22 - 27 October | Security testing, is pipeline secure enough | 90% |  |
| 29 Oct - 4 Nov | Integration with the other members, and final documentation uploads | 100% |  |
| 5 - 11 November | Complete system integration testing | 100% |  |
| 12 - 17 November | Finalize documentation | 100% |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Progress Tracking** |  |  |  |  |
|  |  | **Project Status** | | |
| **Dates** | **% Complete** | **Green** | **Amber** | **Red** |
| Prior to 16 September | 10- 12 |  |  | Red |
| 17 - 23 September | 10- 12 |  |  | Red |
| 24 - 30 September | 20-25 |  |  | Red |
| 1 - 7 October | 20-25 |  |  | Red |
| 8 - 14 October | 35-40 |  | Amber |  |
| 15 - 21 October | 35-40 |  |  | RED |
| 22 - 27 October |  |  |  |  |
| 29 Oct - 4 Nov |  |  |  |  |
| 5 - 11 November |  |  |  |  |
| 12 - 17 November |  |  |  |  |

6.2.1

With this project it is very difficult to keep up and make progress. The reasons for this being that the member c has never done the work expected of him. Most of the time was spend reading up on possible solution and more effective ways to design the pipeline. Some of this research proved valuable in deciding in which design direction to go. Other times time was wasted reading up on something, which proved to be useless for the particular project. Having little experience in networking, is really slowing down the progress.

Furthermore, due to the somewhat vague direction on the procedure to be followed in the design and documentation of this project there was a great of confusion which ultimately caused work to be delayed. Only being able to start work when there was a good enough understanding of what is expected.

Since I am working with two other members, design progress and coding was delayed since I was waiting for them to make progress, so that they could inform me what I had to send and receive on my pipeline.

Lastly, although we are behind as a team the team is working well together as a team and we are all learning a great amount. We will continue to progress as much as possible towards successfully completing the project by 19 November.

## 3.1. Ruben van Wyk Engineering Report

### 3.1.1 Scope of the work

The scope of the hardware design element is to set up the Raspberry Pi, program it to be handle input from various external element from the end user interface and perpetrator interface and provide some type of output.

Essentially this is a cornerstone of the project as this relates to the physical alarm device. The Pi will act as the primary alarm module and will be connected to various sensors and buttons and LEDs. In this part of the project embedded software is important. The development and research is tremendous as such because the Linux environment is completely new and the members do not have much experience with embedded programming.

This part of the project is the physical forefront of the design and determines what the end-user will ultimately see and experience.

### 3.1.2 Assumptions and Constraints

* We assume that the Pi will be connected to a stable standard power source.
* We assume that the alarm system will be set up on the inside of a home and therefore not be exposed to rain or extreme weather.
* We assume an operable temperature of between freezing point to 30 degrees Celsius. The alarm module can therefore not be situated near a source of heat or steam such as a kettle or oil heater.
* We assume at most part that sensors are connected to the alarm module – there is therefore functional input to the system.
* We assume network connectivity via a GSM module to ensure contact with security forces or integrated system feedback.

### 3.1.3 Specific Engineering Methodology

Embedded programming as a field is known for its unique methodology. Specifics of the system such as interrupt use, interrupt service routines (ISR) and other embedded specific details of the system will only be described at a later stage when more of the design and development is completed. The use of strange foreign environments such as Linux and possible Python also contain specific jargon which will be described in more detail at a later stage.

### 3.1.4 Schematic structure

For this part of the project please refer back to Figure 3 for a state diagram and Figure 5 for a flow chart of the system.

### 3.1.5 Physical Measurements

No physical results have been produced at this time and therefore it is impossible to provide interpretation of physical results at this time.

### 3.1.6 Health and safety considerations

The electronics of this system are harmless but care should be taken to recycle once a systems has received unrepairable damage. The Pi will be put in an enclosure and therefore it is not probable that an external short circuit or exposure to electric current from within the circuit can occur. The system should not be handled by infants or children as exposure violent play may cause damage to the physical system elements.

### 3.1.7 Additional Productivity Tools

Multiple guides and external resources as to the design of an alarm system using a Raspberry Pi is currently being used. All these resources are open source and provide helpful insights as to workings of the embedded environment because these are foreign concepts to the group. These documents are by a handful of authors such as Marc-Olivier Schwartz and Matthew Poole.

## 3.2. Johan Fick Engineering Report

### 3.2.1 Scope of work

* Design the back- end security software program - This program will allow the end user to interface via a GUI. Notifications and other crucial protocols will be built into the software to ensure that quick and accurate response strategies can be made. (This will instruct the operators what to do)
* Design the database – The database will be used to store the information regarding each individual device. Refer to the database diagram for the relevant storing information.
* Design the software needed to allow the database to be connected with the cloud.
* Ensure that the software program is able to interface with the “data transfer environment”.

### 3.2.2 Assumptions and Constraints

* The coding language must be able to integrated with the other parts of the project, such as the data transfer part and the system on chip part of the project.
* The database must be designed in such a way that it will provided maximum information with the least amount of data transfer from the SoC.

### 3.2.3 Discipline- Specific Engineering Methodology

* State diagrams will be drawn to help with the code process.
* UML diagrams will be drawn to increase the speed at which can be programmed.
* The database will be designed using MySQL Workbench. This will help increase the overall product quality.
* Subsystem analysis will be done to identify all the components of the subsystem.

### 3.2.4 Simulations/ Flowcharts /State diagrams

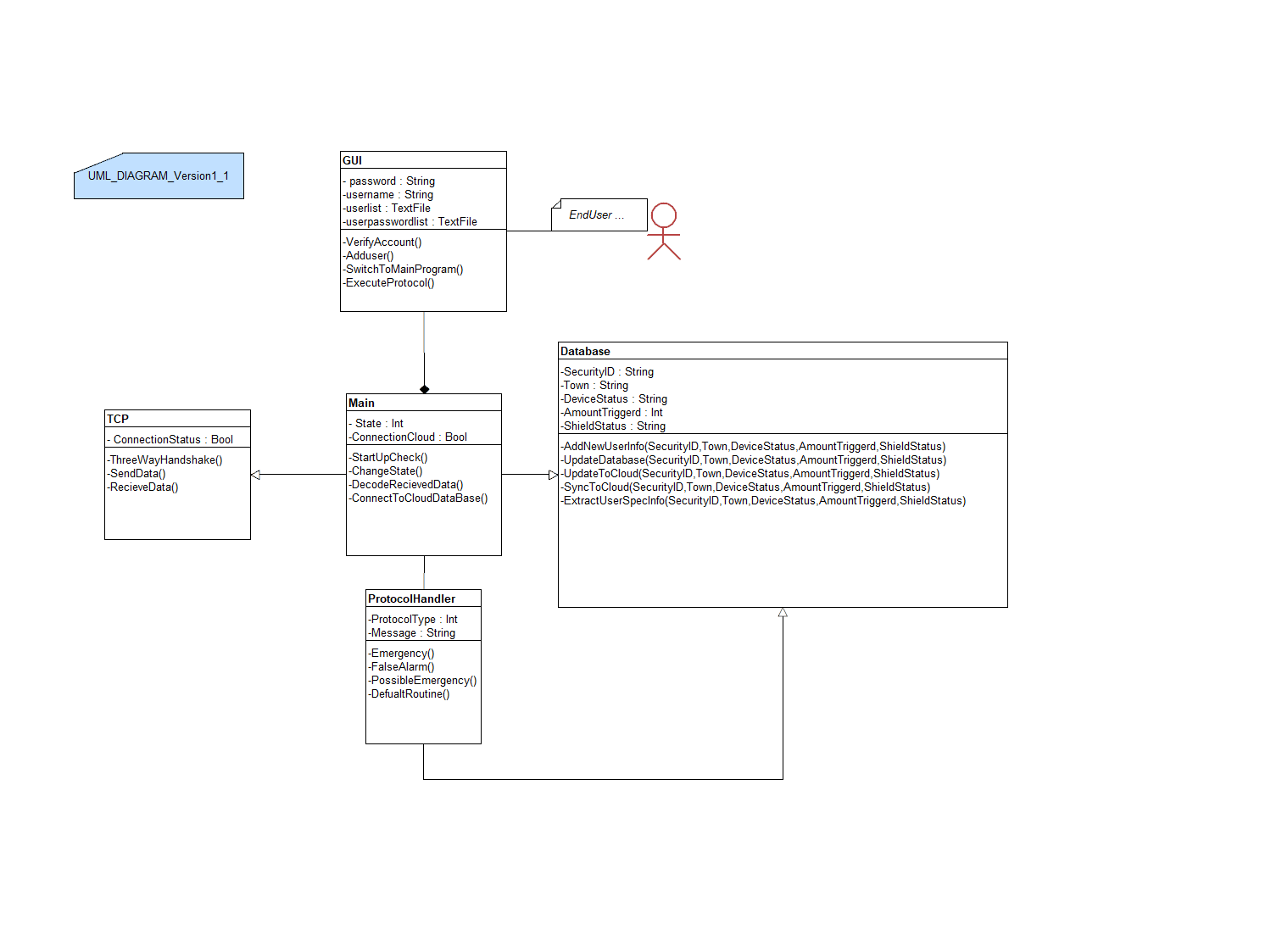
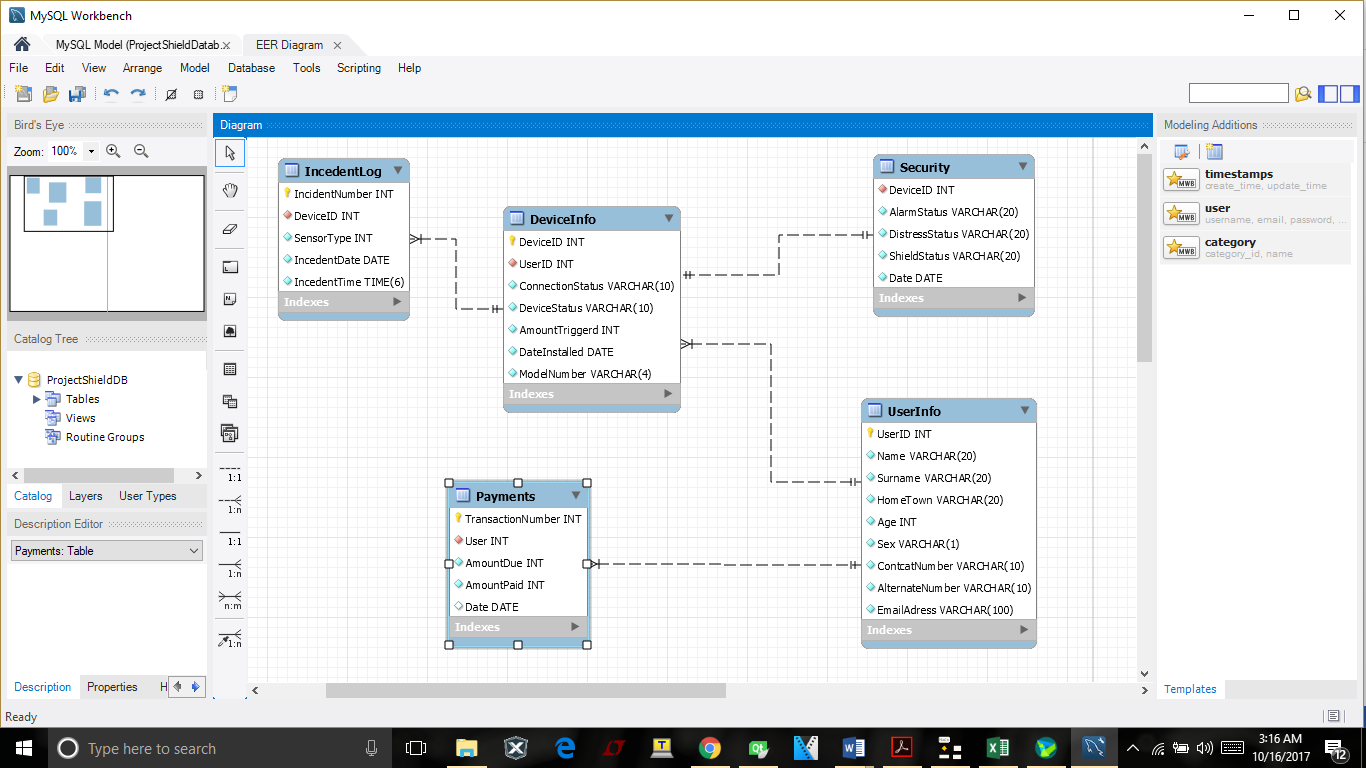
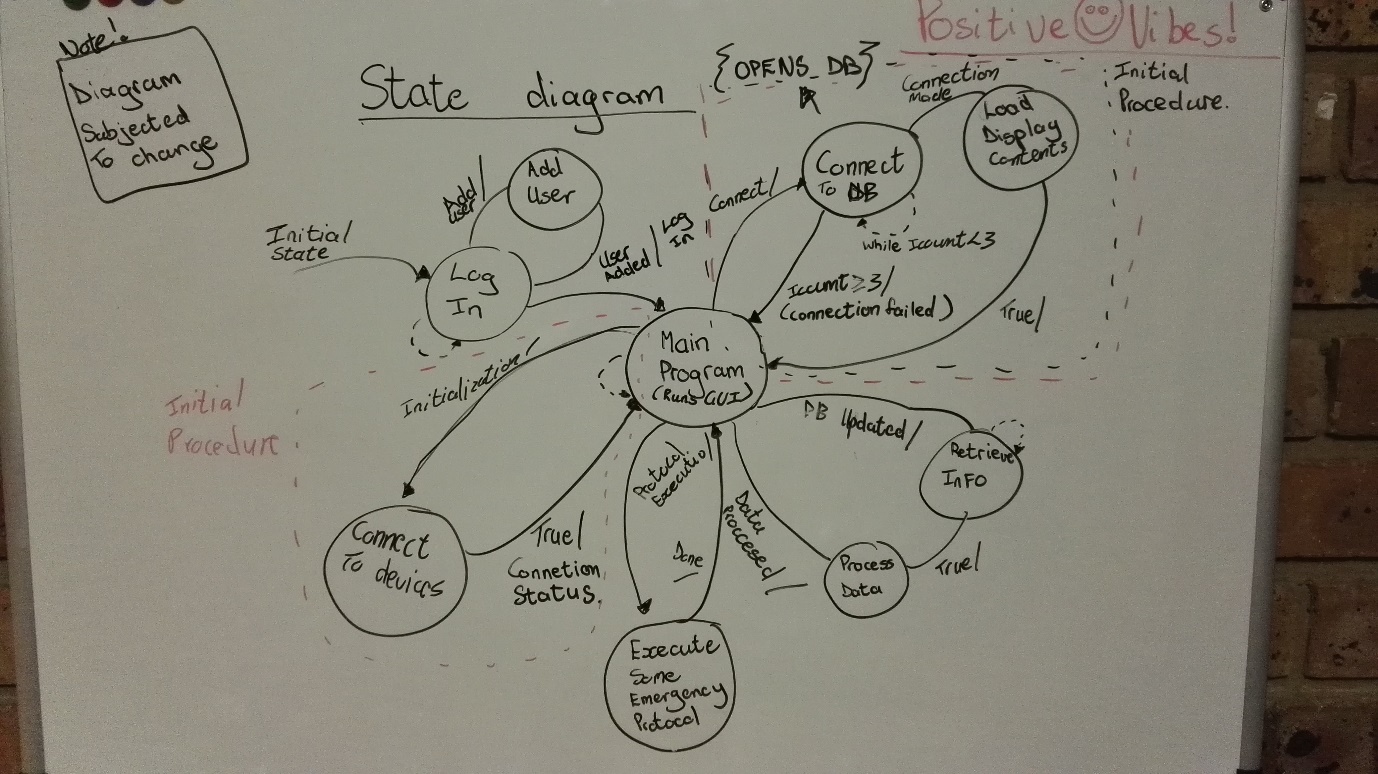


Figure : Uml Diagram





### 3.2.5 Physical Tests/ Measurements (Explanation of test Results)

* (Too be updated)

### 3.2.6 Health and safety considerations (Full Life cycle)

* (Too be updated)

### 3.2.7 Discussion of tools used to enhance productivity

* (Too be updated)
* Sub-system Specification Documents

## 3.3. Burger Odendaal Engineering Report

### 3.3.1 Scope of work

* Designing of the network pipeline between the MCU (Physical alarm hardware) and the control station. Since it is an alarm system there should at least be some basic security built-in the communication pipeline. A TCP SSL secure socket layer is an example as a basic encryption algorithm which will be implemented. Furthermore, a logical algorithm will have to ensure that a break in the connection line is realized in due time.
* It is this member’s responsibility to ensure that the control pc and MCU (alarm hardware part) can effectively communicate in a half-duplex mode.

### 3.3.2 Assumptions and Constraints

* Incorporating scalability into the design of the network is not a priority, since at this stage there is only one control PC, and one alarm system.
* There should however be a basic implementation for controlling more than one alarm system with one control pc, however not to the extent where the amount of control stations to one computer becomes so much that the throughput of network is too small effectively work as it should.

### 3.3.3 Discipline- Specific Engineering Methodology

* Flow charts and UML diagrams will be used to help the logical thinking process of coding the network pipeline.
* Since a TCP socket connection is a very standard, example code of the QT c++ API will be used as a guide to get started. This example will be modified and altered according to the project’s needs.
* Wireshark, network monitoring software will be actively used to ensure that the final pipeline is secure and encrypted.
* Throughout the coding process Telnet will be used for debugging and other one sided testing procedures.
* Since member c does not have access to the MCU (RASBERRY PI) a Raspberry PI emulator will first be used to test the network socket on the MCU’s side of the pipline.

### 3.3.4 Simulations/ Flowcharts /State diagrams

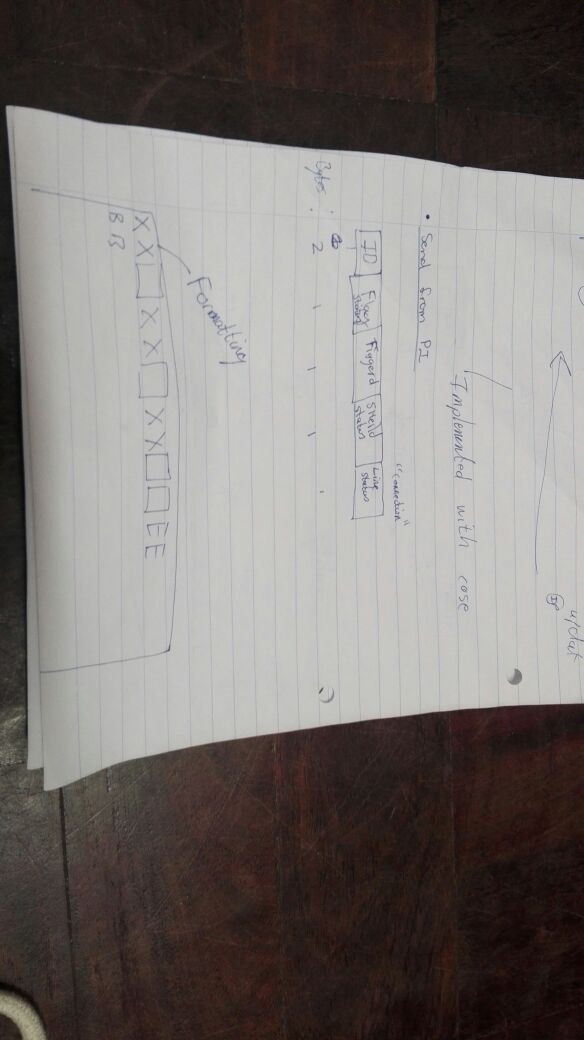


Figure : Format Send from PI

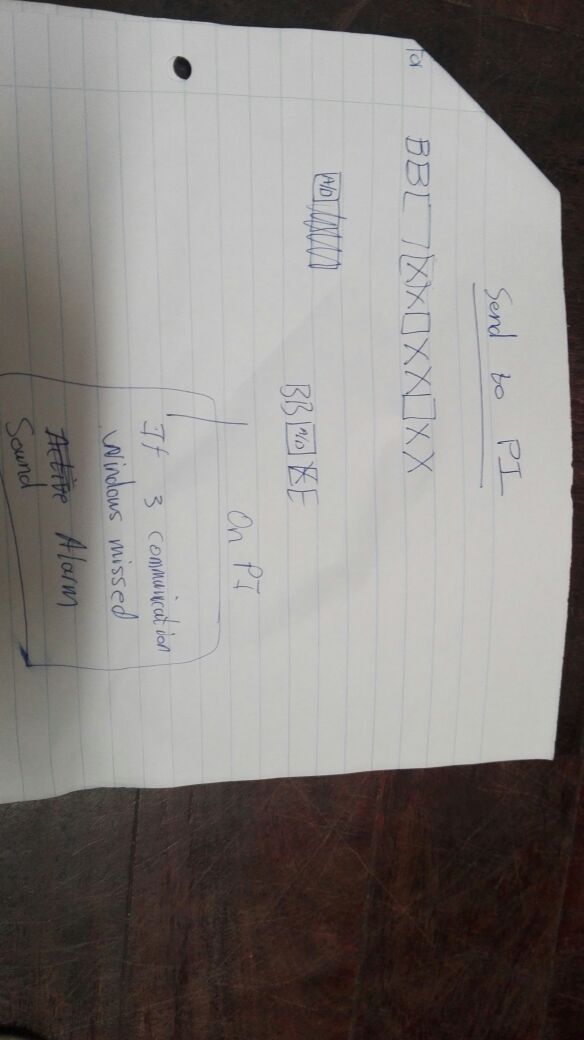


Figure :Format Receive from PI

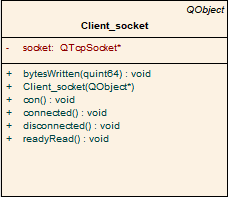


Figure :Socket class UML diagram

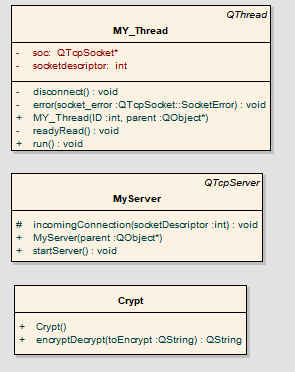


Figure :TCP Multithreaded Server classes UML diagrams

### 3.3.5 Physical Tests/ Measurements (Explanation of test Results)

* (Too be updated)

### 3.3.6 Health and safety considerations (Full Life cycle)

* (Too be updated)

### 3.3.7 Discussion of tools used to enhance productivity

* (Too be updated)

# 4. Sub-system Specification Documents

## 4.1 Hardware system specifications (Ruben)

### 4.1.1 Sub-system Functional Flow

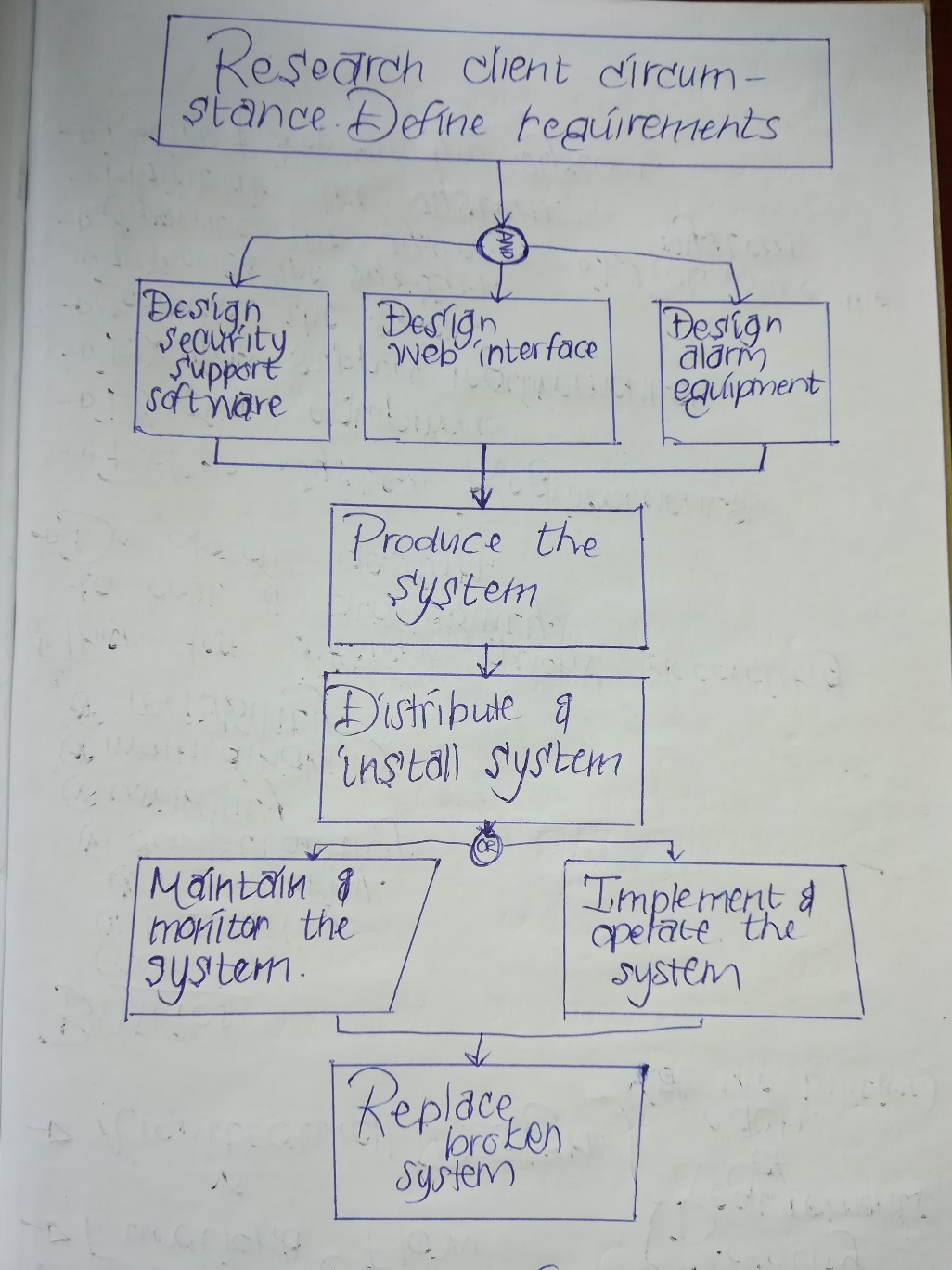


Figure : Total functional flow of the system

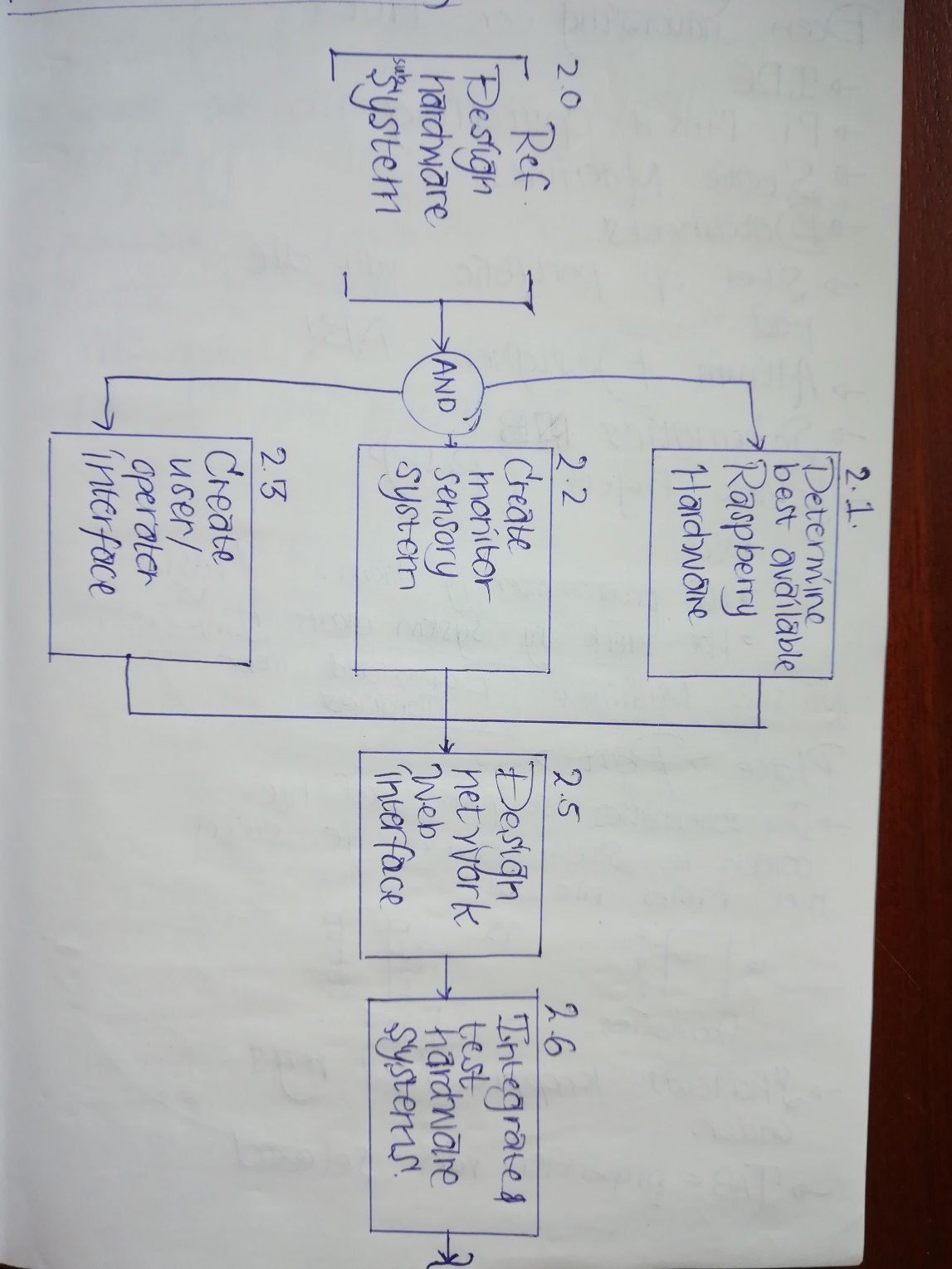


Figure :Functional flow of the hardware subsystem

### 4.1.2 Interface Definitions

* **IF1 User Interface** – This interface is the primary way in which the user will interact with the system. This will include buttons and some form of display output (LED’s or LCD screen) that will allow the user to disable and enable the system, send an emergency SOS signal and access other such standard alarm functionality.
* **IF2 Perpetrator Interface** – This interface includes the sensory array that will pick up when a perpetrator has entered the house. This is the primary input of the system and the most functional objective of the entire system since essentially we just want to know when someone is in our house that should not be there.

## 4.2 Specifications Johan Fick

### 4.2.1 Sub-system Functional Flow

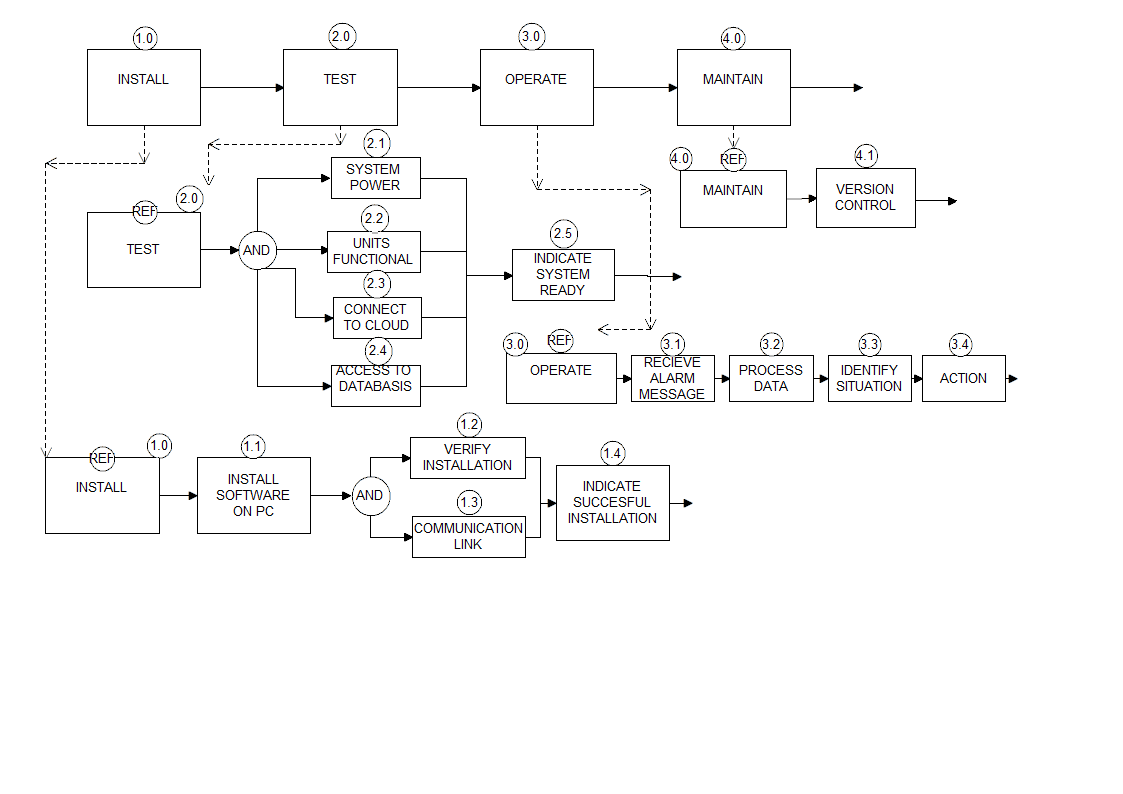


Figure -Subsystem functional analysis

### 4.2.2 Interface Definitions

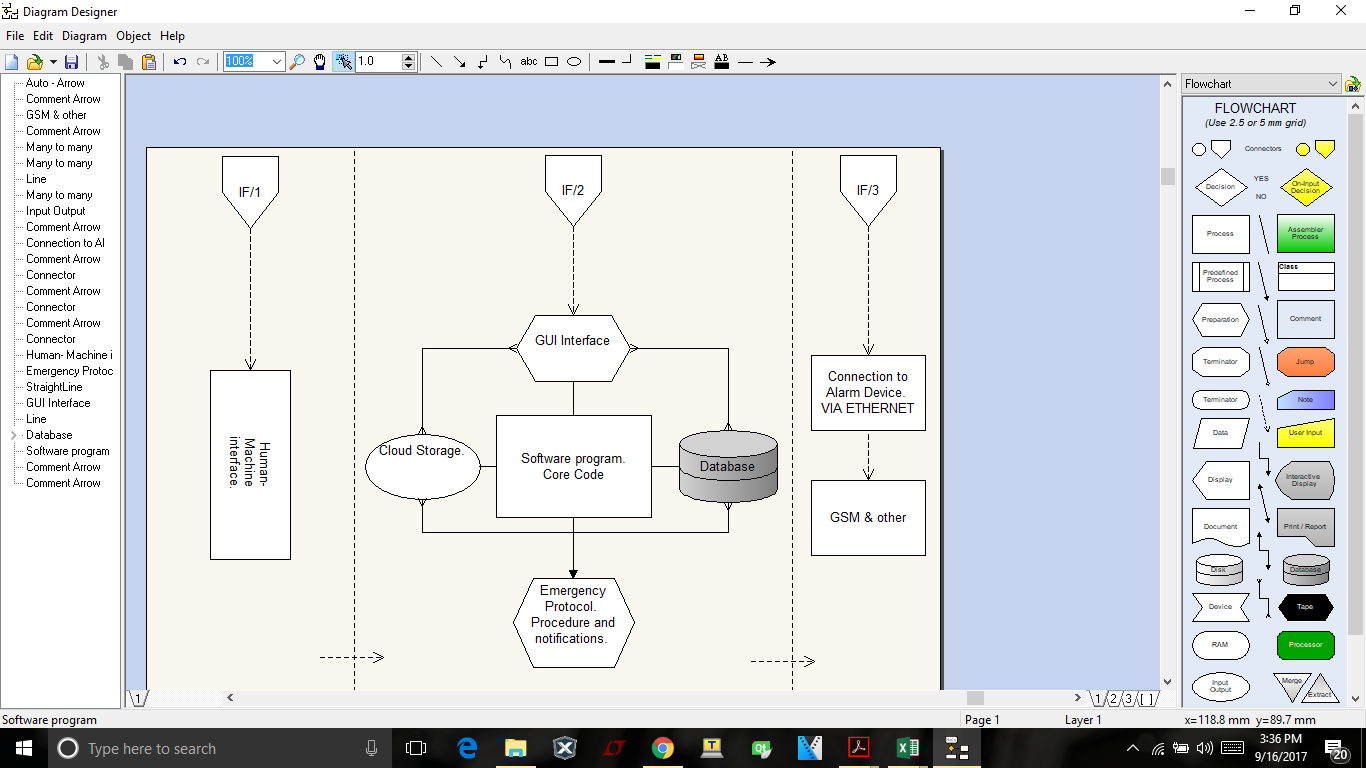


Figure -Subsystem functional architecture.

* **IF1 Human machine interface** – This interface will allow a user to interface to the software program. The program will be hosted on a computer and with the aid of a screen, speakers, mouse and a keyboard. The user will be able to make adjustments to the program.
* **IF2 Software** – This interface will be used to interface the different elements of the software program. It will allow the different elements such as gui, database ect. To communicate with each other and will allow overall program functionality.
* **IF3 Cloud** – This interface will allow the database to be attached to the cloud. Therefore ensuring data safety.

## 4.2 Specifications Burger Odendaal

### 4.2.1 Sub-system Functional Flow

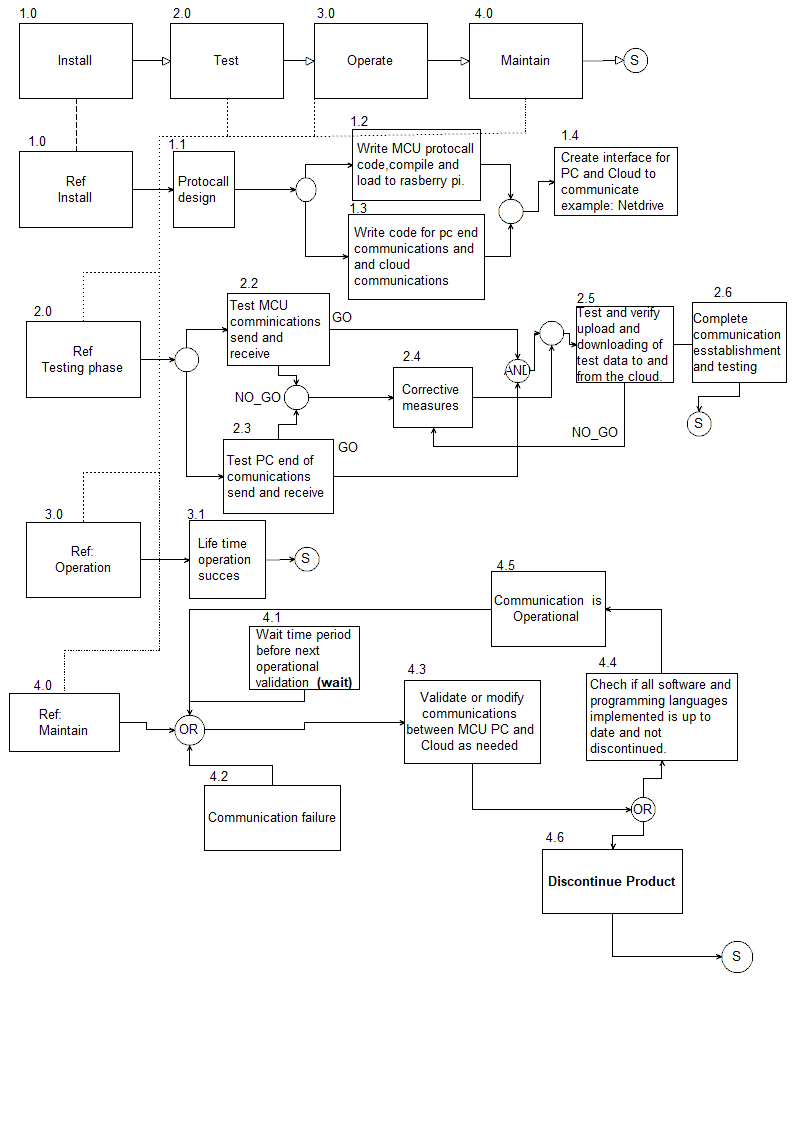
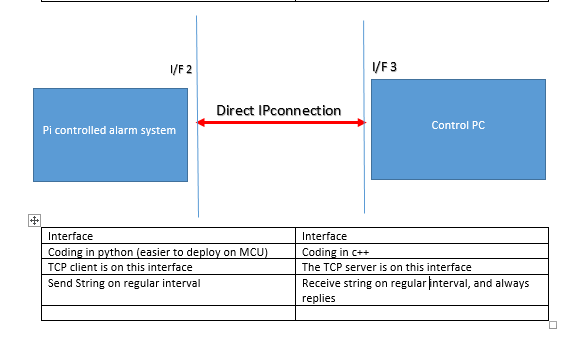


Figure : Sub functional analysis of network pipeline

### 4.2.2 Interface Definitions



* **I/F2-** Interface between the MCU (Raspberry PI) and the network pipeline. This interface sends data on a regular time interval, and receives a reply each time.
* **I/F 3-** Interface between the control PC (containing TCP server) and the network pipeline. The TCP server actively listens on a specified port for a connection to be made. Once a message is received it immediately responds with a message from the control PC. The connection is closed immediately as little as possible open connections at one time.

# 5. Design documentation

## 5.1 System design documentation

### 5.1.1 Final System Functional Definition Final requirements

* Alarm system should consist of four motion sensors and at least one door contact.
* At the very least a direct IP connection should be able to inform the control station that an event has occurred.
* Furthermore, then the control system has to be able to send a signal through the ip connection, allowing the alarm enabled and disabled
* The control system must be able to the following log alarm information to a cloud database:

1. ID
2. Town
3. Status of alarm
4. Amount triggered (amount of times that the alarm was triggered)
5. SHIELD status.

* This information must be send to the database via the control PC on 7 second intervals.
* If more than three intervals do not arrive at the control pc then a break in connection must be logged, and reported.
* An encrypted SSL TCP socket connection must be used to connect the control computer to the physical alarm system. This will minimize the chance of a cyber-attack on the alarm system.

### 5.1.2 System Concept Drawings

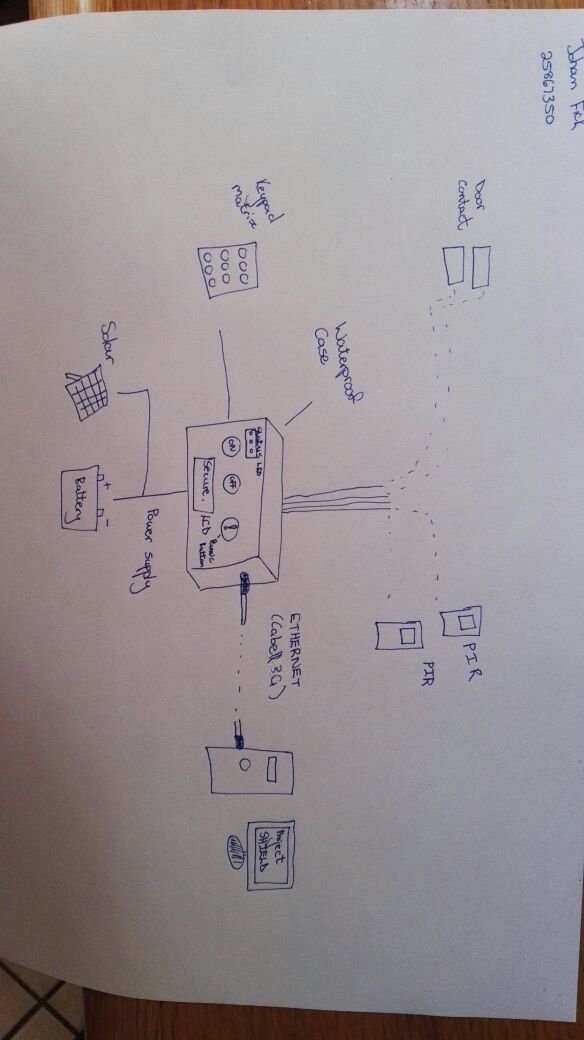


Figure :Design Drawing A

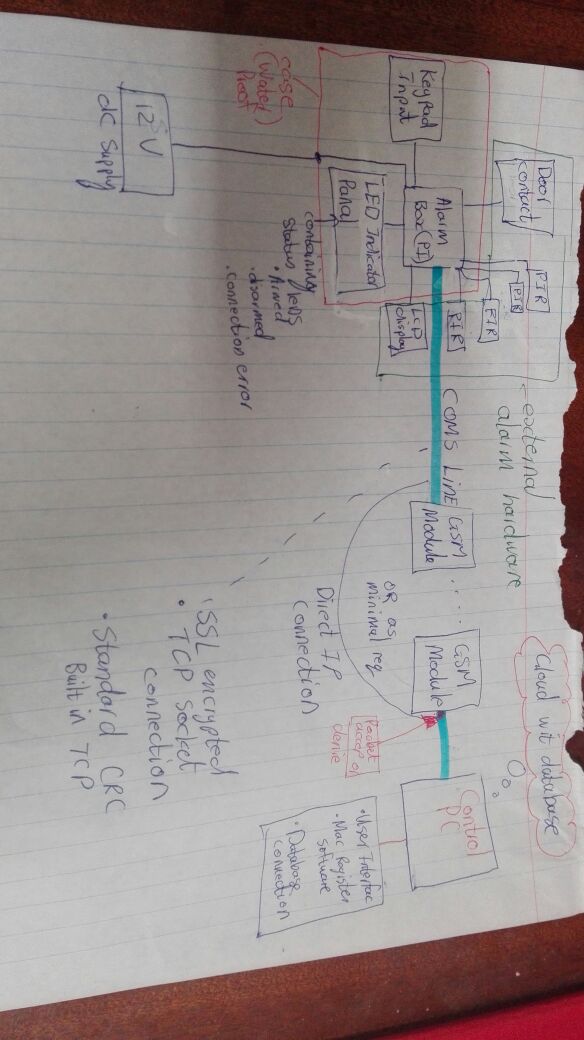


Figure :Design Drawing B

### 5.1.3 System Interface Definitions

Three interfaces

1. Alarm system (Raspberry PI)
2. Communication network
3. Control PC

* Alarm system **to** Communication network, on 7 second intervals
* ID
* Town
* Status
* Amount of times triggered
* Shield status

This information must be handed over to the network on a character string, which is smaller than 1452 bytes (maximum payload size of common TCP/IP packet). It is estimated that not more than 20 bytes would be necessary.

Furthermore, there is an agreement that the seven seconds interval time will be reduced to a shorter time period id possible. The seven second was a guess. When the network is up and running and the general amount of throughput is known this amount shall be

Communication network **to** Control PC

* ID
* Town
* Status
* Amount of times triggered
* Shield status

The control PC will receive a single TCP packet string containing the above information. A function shall extract the information. The control pc will make sure that this information arrives every time. If more than tree communication windows are missed, a connection error event will be triggered and updated as such. The alarm system will be setup as a client to the Control pc hosting the TCP server. Each time that the server receives a message from the client, it will reply with one of the following actions.

* No action everything is okay.
* reset and Arm.
* Received corrupt package, resent package.

### 5.1.4 System Integration testing

Integration testing will be performed on the following criteria.

* No interface has more importance than the others
* Three main parts exist, Alarm system, Network and control Station.

Integration testing procedure:

1. Each designer will test his/her own design, based on his/her design constraints.
2. Each member’s part of the design and final product, will be given passing conditions by another member of the team. Furthermore, be tested a member not responsible for the design. This is all to avoid the occurrence of conformation bias.
3. After individual units or stages is tested like explained above and passes, two of the three units will be combined and tested by all members. The Alarm unit and the network section will be combined and then tested by all members. Where the previous passing condition will be combined. The network part will echo all information back to the alarm section, for testing purposes. It is important that each stage of testing will only advance once the current stage has completely passed the testing conditions.
4. For the final stage of testing the control PC will be added and the complete system will be tested as a whole. Here all the passing conditions of each testing stage will all be added up.

Table : Passing conditions table

|  |  |  |
| --- | --- | --- |
| **Testing stage 1 passing**  **conditions (Alarm hardware)** | **Testing stage 2 passing conditions (Network section)** | **Testing stage 3 passing conditions (Alarm hardware)** |
| Each PIR sensor triggers an event when motion is detected. | The network is active and continuously feeds data. | Every time data is received the database on the cloud or pc should be updated. |
| The door contact activates an event when the door opens. | The complete data segment is transmitted correctly. | The database should not have any basic SQL injection vulnerabilities. |
| All relevant keypad buttons trigger registers on the raspberry pi. | Corrupted packages are identified and resent. | The user interface on the control pc should accept only meaningful user input and not crash when non-meaningful input is inserted by the controlling person. |
| Non meaningful input from the keypad is ignored, and does not cause the system to crash and malfunction. | Corrupt packages are not given over to the control pc. This is important since corrupt packages can be used to manipulate the actions of the control station, by an intruder. | No large excessive memory leaks should be present. |
| No internal wires are visible to the end user. | The data must be secure; this must be tested using wire shark. When the datagram is opened in wire shark, the TCP datagram, must be encrypted and not human readable. Thus when an intruder monitors the TCP transmission he should not be able to decipher the format of the data send over the network, otherwise he would be able to manipulate the control pc by sending corrupt manipulated data. | If external files are used and stored on the computer it should be encrypted. |
| The alarm updates all data field every 7 or less seconds. | Both sides of the network must only accept data from known mac addresses. This for example means that when the control pc receives a TCP packet from an unknown MAC it should be discarded. Furthermore, an attempted cyber-attack should be logged in the database by the control PC. This means that the MAC address of the raspberry pi should be registered. | The interface should be user friendly. |
| Watchdog service handles a brown- out situation by resting the MCU and continuing in the same state as state as before the brown out situation. This condition is critical. | The network must have enough effective throughput, to not miss more than three windows of communication. | The software should be easy to install, and not require advance installations procedures, like editing path variables or any files on the computer; moving files to different directories in order for the software to function correctly. |
| System does not crash when multiple sensors trigger a motion event at the same time. | The data payload should be smaller than 100 bytes. |  |
|  |  |  |
|  |  |  |

**Integration testing results**

Not completed at this stage, waiting the completion of implementation and prototype construction.

## 5.2 Ruben an Wyk

### 5.2.1

### 5.2.2

### 5.2.3

### 5.2.4

### 5.2.5

### 5.2.6 Design Implementation

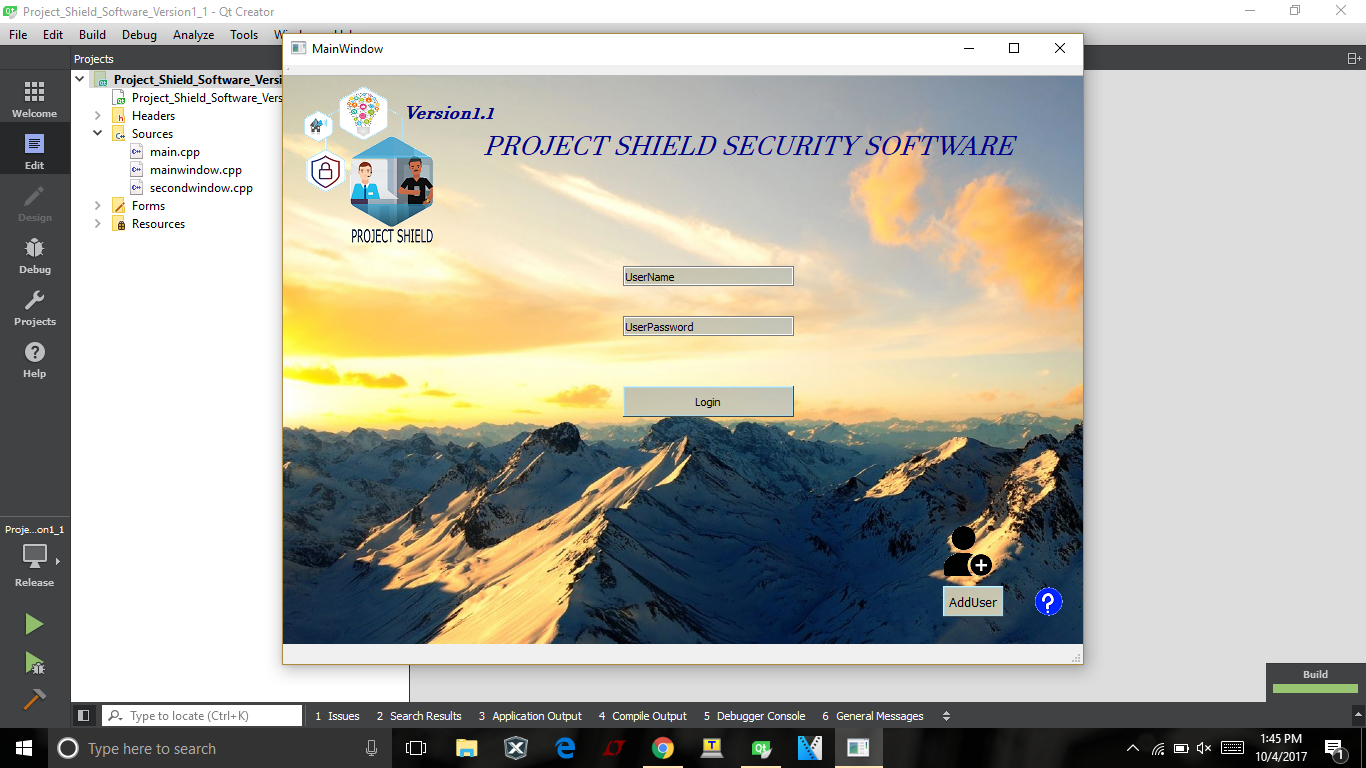
No part of the design has reached a point where full-scale implementation is possible, therefore this piece of the portfolio will be omitted.

## 5.2.7 System Test and Evaluation

As stated in the previous paragraph, no part of the project is operational thus far as to come to a place where it can be tested or evaluated for further betterment.

## 5.3 Johan Fick

The picture below is that of the program designed in Qt up to this point. The program has a fully functional login window with support of adding new users . The reason for not progressing with the program is simple and is due to the fact that there is uncertianty between the different software languages that we will be using . We will firstly have to consult Dr. Andreas Alberts to ensure that the different software languages that we will be using will , be able to interface with each other .



### 5.3.1

### 5.3.2

### 5.3.3

### 5.3.4

### 5.3.5

### 5.3.6 Design Implementation

No part of the design has reached a point where full-scale implementation is possible, therefore this piece of the portfolio will be omitted.

## 5.3.7 System Test and Evaluation

As stated in the previous paragraph, no part of the project is operational thus far as to come to a place where it can be tested or evaluated for further betterment.

## 5.4 Burger Odendaal

### 5.4.1

### 5.4.2

### 5.4.3

### 5.4.4

### 5.4.5

### 5.4.6 Design Implementation

No part of the design has reached a point where full-scale implementation is possible, therefore this piece of the portfolio will be omitted.

### 5.4.7 System Test and Evaluation

As stated in the previous paragraph, no part of the project is operational thus far as to come to a place where it can be tested or evaluated for further betterment.