**ELO 5**

**Randolph Bock 25905570**

**REII 327**

**NET 1 BIER**

Design 2017

Contents

[Scope of work: 1](#_Toc497037612)

[ASSUMPTIONS AND CONSTRAINS: 1](#_Toc497037613)

[Discipline-specific engineer methodology: 1](#_Toc497037614)

[Unit 1: 3](#_Toc497037615)

[1.1 Human machine interface module: 3](#_Toc497037616)

[1.1.1 Input to Machine: 3](#_Toc497037617)

[1.1.2 Visual output: 4](#_Toc497037618)

[1.1.3 Audio output: 5](#_Toc497037619)

[1.2 System on chip (SoC) control unit: 6](#_Toc497037620)

[1.2.1 State diagram: 7](#_Toc497037621)

[1.2.2 Flow diagram: 8](#_Toc497037622)

[1.2.3 Pin Functions & Alocations: 9](#_Toc497037623)

[1.3 Alarm sensor interface module: 11](#_Toc497037624)

[1.3.1 Movement detector: 11](#_Toc497037625)

[1.3.2 Door contact: 11](#_Toc497037626)

[1.4 Power supply. 12](#_Toc497037627)

[Unit 2: 14](#_Toc497037628)

[2.1 Network: 14](#_Toc497037629)

[2.2 Protocol: 15](#_Toc497037630)

[Alarm Code: 15](#_Toc497037631)

[Bill of materials: 16](#_Toc497037632)

[Alarm panel design: 17](#_Toc497037633)

[Alarm Panel construction: 18](#_Toc497037634)

[Physical Test/ measurements: 19](#_Toc497037635)

[Tools required for instalation: 21](#_Toc497037636)

[Health & Safety considirations: 21](#_Toc497037637)

[Tools used to enhance productivity: 22](#_Toc497037638)

**List of Figures:**

[Figure 1: Unit 1 System definition 3](#_Toc497037639)

[Figure 2: Visual output driver circuit 5](#_Toc497037640)

[Figure 3: Siren output driver circuit 6](#_Toc497037641)

[Figure 4: Finite State Machine 7](#_Toc497037642)

[Figure 5: Flow diagram 8](#_Toc497037643)

[Figure 6: GPIO Header allocation 10](file:///C:\Users\rambo_000\Dropbox\Universiteit\2de%20Semester\REII%20327\Design%202017%20-%20Randolph%20ELO5%20-%20Net1Bier.docx#_Toc497037644)

[Figure 7: Power supply interface 12](#_Toc497037645)

[Figure 8: Power Supply 13](#_Toc497037646)

[Figure 9: Unit 2 System definition 14](#_Toc497037647)

[Figure 10: Alarm panel Final Design 17](#_Toc497037648)

[Figure 11: Alarm panel Constructed 18](#_Toc497037649)

[Figure 12: Alarm panel (Inside View) 18](#_Toc497037650)

**List of Tables:**

[Table 1: LED functions 4](#_Toc497037651)

[Table 2: Pin functions: 9](#_Toc497037652)

[Table 3 : Power Budget 12](#_Toc497037653)

[Table 4: BOM 16](#_Toc497037654)

[Table 5: Development Testing 19](#_Toc497037655)

[Table 6: Implementation Testing 20](#_Toc497037656)

**List of abbreviations:**

* FSM – Finite State Machine
* UDP – User Datagram Protocol
* OS – Operating System
* SoC – System on Chip
* SBC – Single Board Computer.
* DCS – Door Contact Switch

# Scope of work:

1. Design circuit for security inputs (Including Door switches and PIR's)
2. Design circuit for output siren (to be run on 12v)
3. Design and implement a user interface
4. Design and code the SoC to handle all inputs and outputs
5. Design and build an enclosure for the Alarm panel
6. Order parts
7. Trade-off decisions
8. Design and implement a Direct ethernet / IP connection between SoC and Pc
9. Assemble the final unit as a whole

# ASSUMPTIONS AND CONSTRAINS:

* Constrain budget of R1500
* 12V- 6W power limitation
* Assume a ground wire will be accessible to ground the unit
* Assume a stable input power of 12V DC with little to no rippling.
* Assume input power does not exceed 24V
* Time constraint limited deadline and demonstration in November.
* Assume all components bought are correct and work as intended

# Discipline-specific engineer methodology:

All group members are acquainted with programming and electronics.

With Randolph having the most experience with electronic construction, the unit 1 where the electronic construction of the Alarm panel as well as the programming of the standalone system was allocated to him.

FJ having the most experience in computer programming was given unit 3, to design and develop the back end of the system, along with the database.

Tools used in the project to improve productivity and speed up the process of design as time management was a constraint of the project.

Methods used include Flowcharts, FSM diagrams, functional analysis, simulations, version control, and program specific developing environments.

Excel was used for project management to control the flow and work allocation of the project.

For the design process of the Alarm panel for unit 1 a technological survey had to be done to determine the best solution for the design. This includes datasheet, programming notes and tutorials, and trade-off studies.

The design of all circuitry was done in LT SPICE, simulations was run in the same program to validate the workings of the circuit design before building and implementing the design. All circuits were tested on breadboards, before being built and soldered onto prototype PCB board.

For easy and effective design and drawing of flow diagrams and Finite State Machines, an online tool “draw.io” was used. The flow diagrams and FSM are used as blueprints for the system code. Both indicate the process the code must follow to complete the task at hand.

The program for the system was written on a Raspberry Pi running the Raspbian operating system. Raspbian is a distro of the Linux OS Debian. The code was written in Python using the IDLE developing environment. Python version 2.7.14 was used.

All code written for the system was uploaded to GIT HUB for back-up and version control, along with any supporting documentation about changes in to code.

IDLE was also used to edit system files on the Raspberry Pi. The files had to be edited to run the system file for the Alarm at start-up of the Raspberry Pi.

# Unit 1:

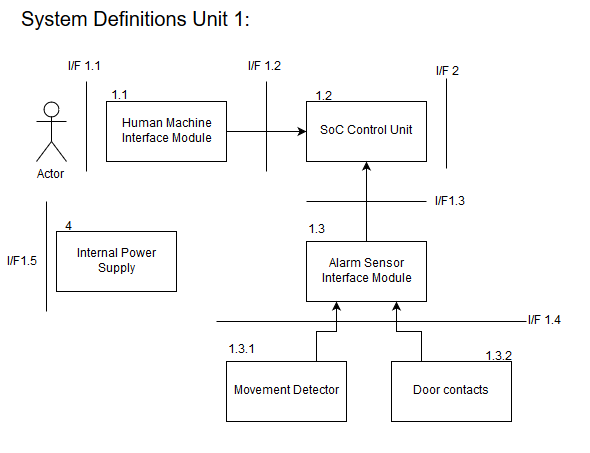


Figure 1: Unit 1 System definition

# Human machine interface module:

## 1.1.1 Input to Machine:

Input to the system will be done by a 3x4 matrix number pad directly interfacing with the SoC. The number pad will be used to enter a pin to disarm the alarm system as well as arming the system.

Specifications of the number pad:

* Pad Size: 68.9 x 76 x 0.8mm
* Cable length: 85mm (including connector)
* Connector: dupont 7 pins, 0.1 inch (2.54mm) Pitch
* Mount style: Self-Adherence
* Max. circuit rating: 35VDC, 100mA
* Insulation spec.: 100M Ohm, 100V
* Dielectric withstand: 250Vrms (60Hz, 1min)
* Contact bounce: ≤5ms
* Life expectancy: 1 million closures

## 1.1.2 Visual output:

The system will have a visual output in the form of LED’s. 3 LED’s will be used:

Table 1: LED functions

|  |  |
| --- | --- |
| Colour | Indication |
| Green | Power (Pulsing as a heartbeat) |
| Orange | Armed(on)/Disarmed(off) |
| Red | Fault (Fast flashing)/Triggered (Slow flashing) |

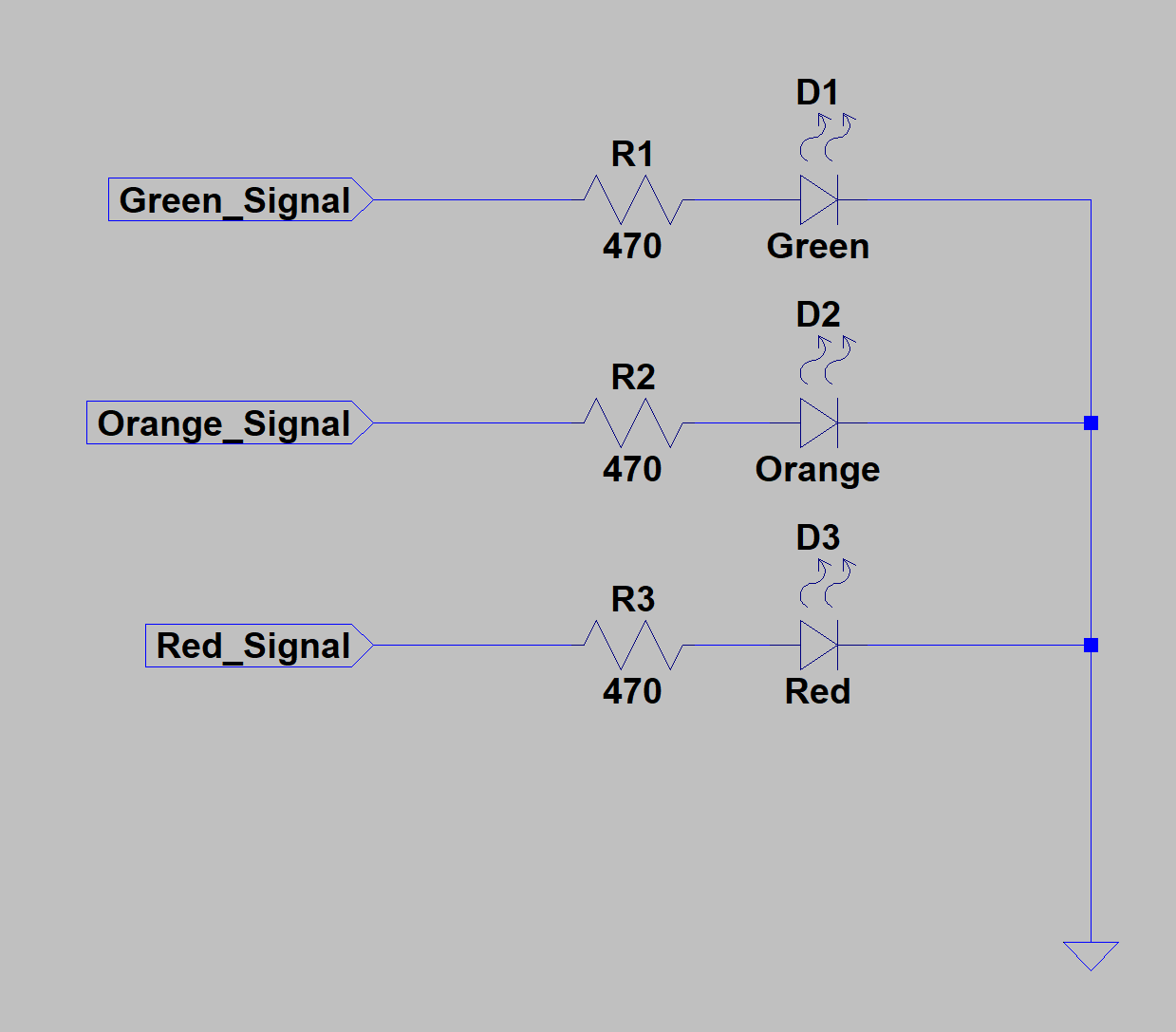


Figure 2: Visual output driver circuit

## 1.1.3 Audio output:

An audio output will be used in the system for the machine to interface with a human. The audio source will be a siren. The siren will sound when a positive input has been received by the system and the system is in the armed state.

The output driving circuit of the siren uses a NPN transistor (2N2222A) to switch on the siren from the SoC. The circuit is as follows:

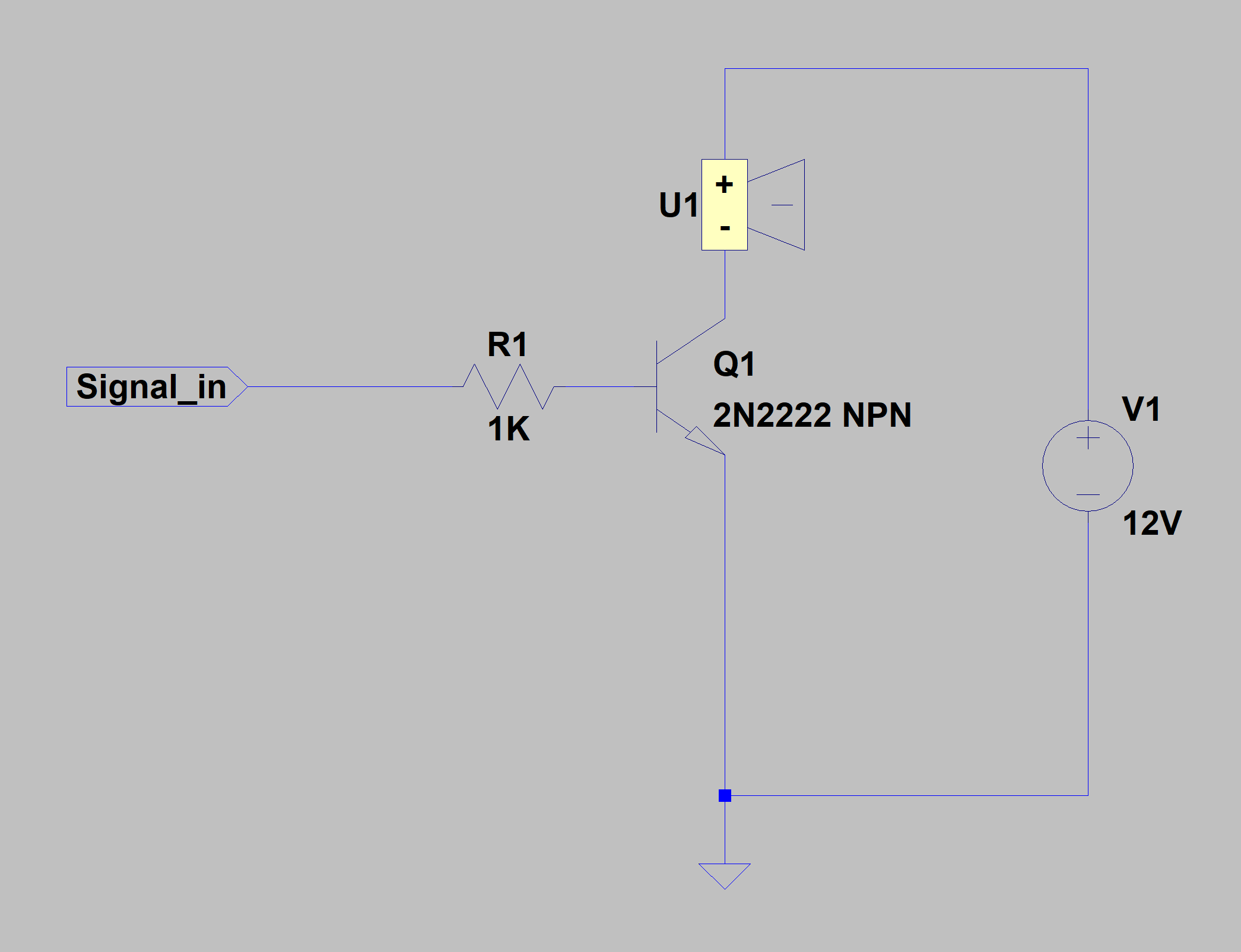


Figure 3: Siren output driver circuit

# 1.2 System on chip (SoC) control unit:

The SoC used will be a Raspberry Pi 3B, running the Raspbian operating system. The programming language used will be Python done in the Idle3 editor on the Raspberry Pi.

To improve power consumption of the system when implemented, certain peripherals not required to run will be switched of. This includes but is not limited to HDMI, and audio.

## 1.2.1 State diagram:

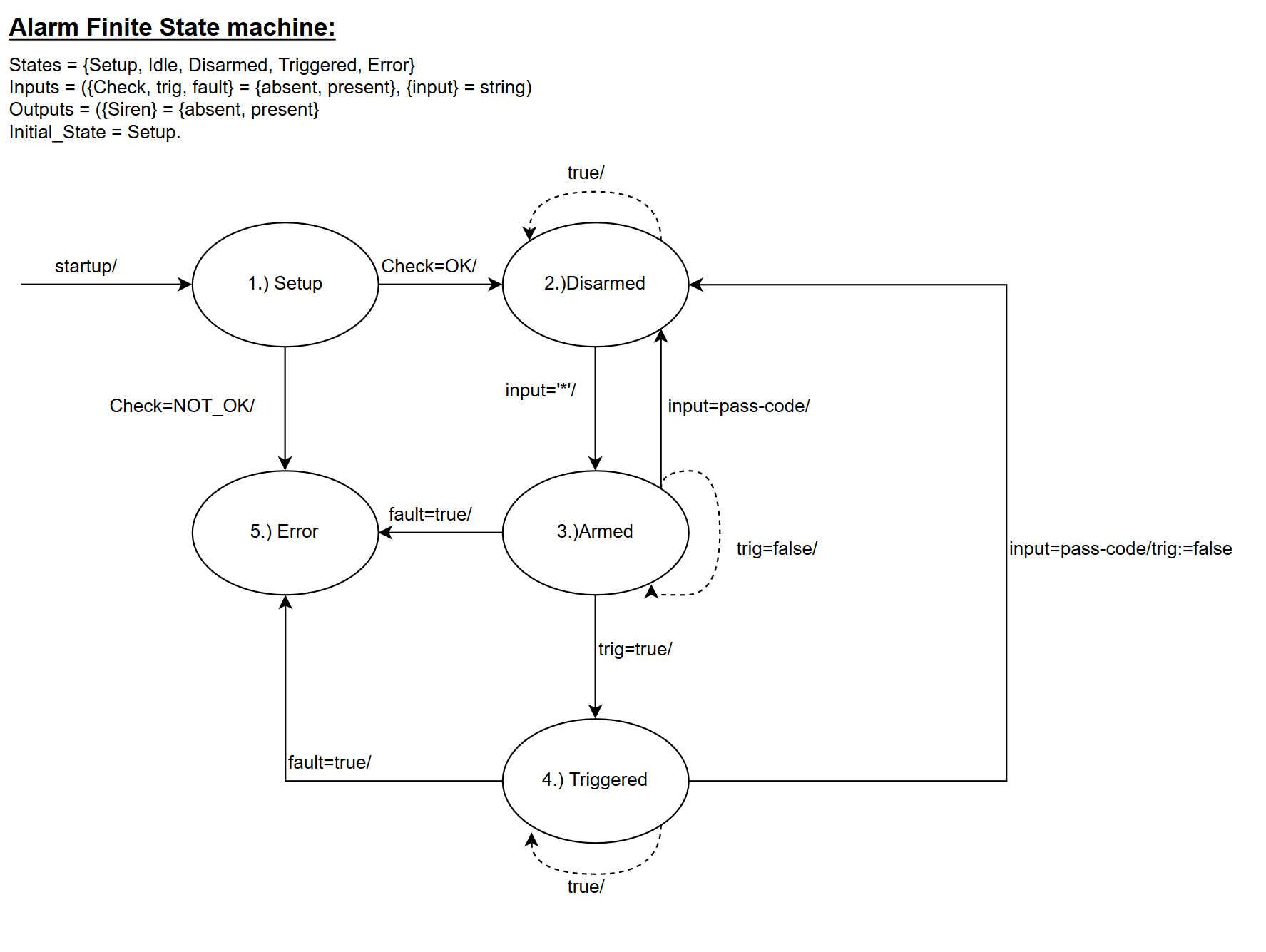


Figure 4: Finite State Machine

## 1.2.2 Flow diagram:

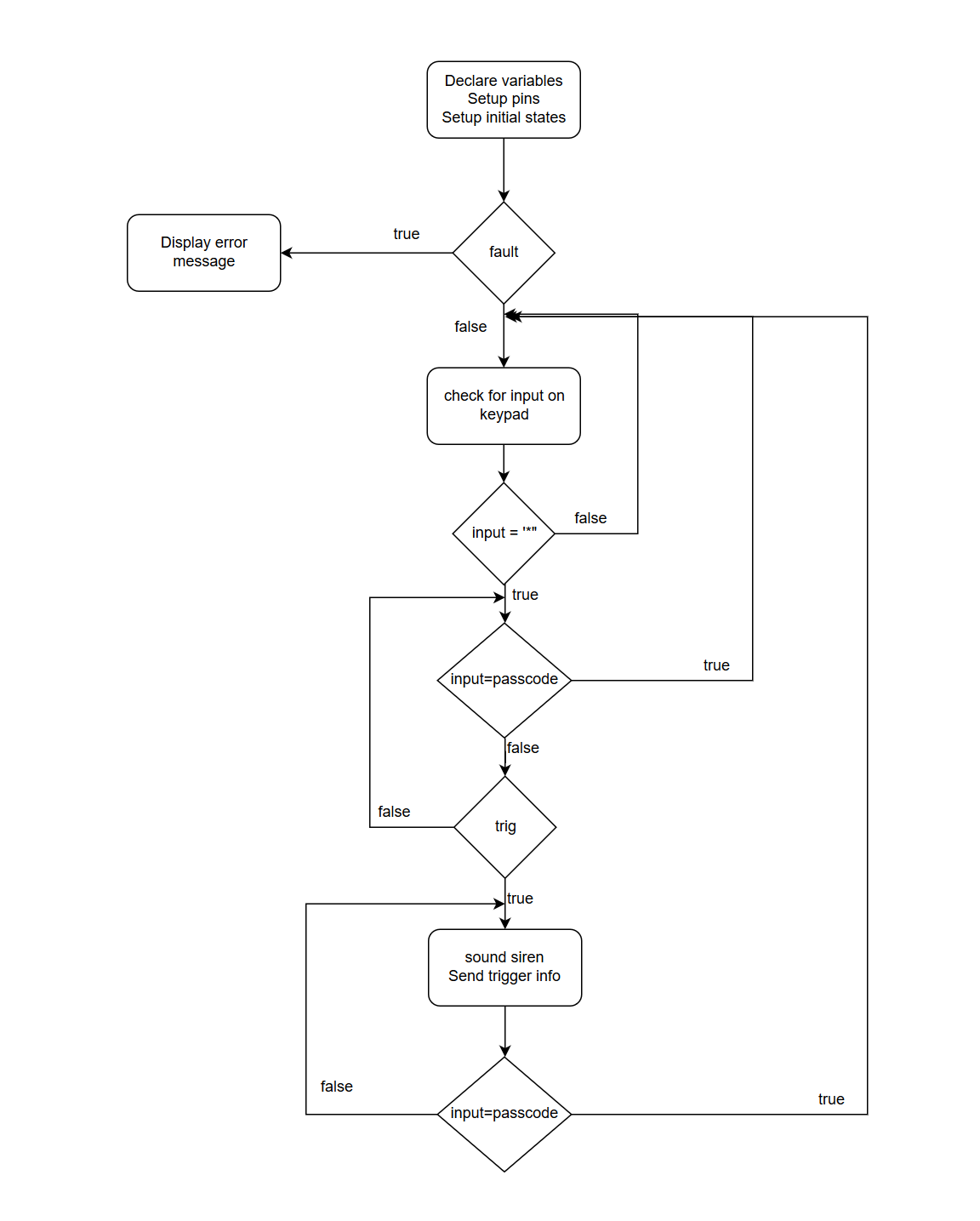


Figure 5: Flow diagram

## 1.2.3 Pin Functions & Alocations:

Table 2: Pin functions:

|  |  |  |
| --- | --- | --- |
| Function | Board Pin Number | Input/output/5V/3.3V/Ground |
| Buzzer | 7 | Output |
| PIR Sense | 13 | Input |
| Reed Switch Sense | 15 | Input |
| Orange LED | 16 | Output |
| Red LED | 18 | Output |
| Ground LED | 20 | Ground |
| Green LED | 22 | Output |
| Siren | 29 | Output |
| Keypad Row (1) | 31 | Output |
| Keypad Row (2) | 33 | Output |
| Keypad Row (3) | 35 | Output |
| Keypad Row (4) | 37 | Output |
| Keypad Colum (1) | 36 | Output |
| Keypad Colum (2) | 38 | Output |
| Keypad Colum (3) | 40 | Output |

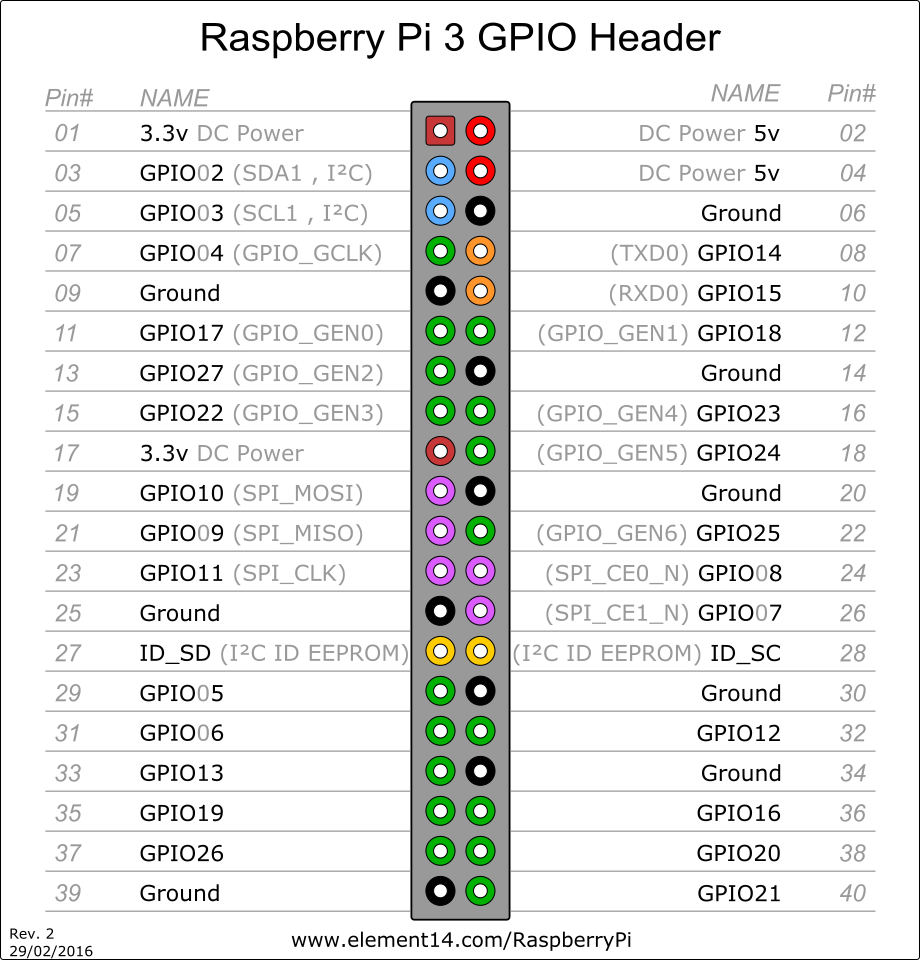


Figure 6: GPIO Header allocation

# 1.3 Alarm sensor interface module:

Input protection must be added to the system to protect the SoC’s pin from voltages that can cause harm. This is done by adding a Zener diode in reverse polarity in parallel to the sensors of the system. The Zener used is a PHBZX79C with a forward voltage of 3.3V.

## 1.3.1 Movement detector:

Movement detection will be done by a PIR sensor, referred to as “Passive Infrared”, or “IR Motion”.

The PIR sensor to be used is the HC-SR501 Infrared Motion Sensor

HC-SR501 Infrared Motion Sensor Specifications:

* Input Voltage: 4.5V - 20V
* Current Draw: <50µA
* Digital Output: 3.3V (High)
* Digital Output: 0V (Low)
* Working Temperature: -15°C to 70°C
* Delay Time: 0.5 - 200 Seconds
* Sensing Angle: 100° Cone
* Range 5m - 7m
* Dimensions
  + Sensor Lens Diameter: 23mm
  + Length: 24.03mm
  + Width: 32.34mm
  + Height (with lens): 24.66mm
  + Centre screw hole distance: 28mm
  + Screw hole diameter: 2mm (M2)

## 1.3.2 Door contact:

Door contact will be sensed with reed switches, connected to the 3.3v rail of the Raspberry Pi.

* Digital output: 3.3V (High)
* Digital output: 0V (Low)

A Low output will be given when the door is open, and a high output when the door is closed.

# 1.4 Power supply.

The power supply must supply a 12V rail as well as a 5V Rail to power the PIR sensors and the SoC. A 3.3V Rail will be obtained from the internal 3.3V rail on the SoC.

Input power will be 12V with a maximum of 6W power delivery. The power budget for the system is therefore 6W.

The input power will be supplied by two wires, a positive(12V) and Ground(0V) wire. The wires will be attached to the system securely using screw in wire terminals.

Table 3 : Power Budget

|  |  |  |  |
| --- | --- | --- | --- |
| **Power Budget** | | | |
| Component | Qty | Power per unit | Power total |
| SoC | 1 | 5W | 5W |
| PIR’s | 4 | 0,6mW | 2,4mW |
| Cooling fan | 1 | 500mW | 500mW |
| Inefficiencies | 1 | 200mW | 200mW |
| Total: |  |  | 5,7024W |

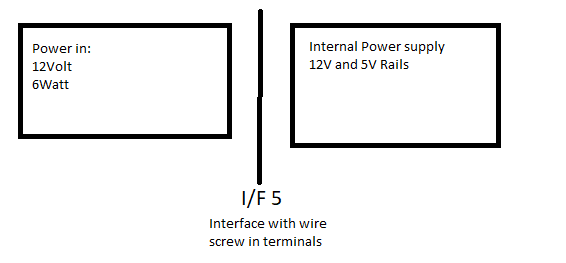


Figure 7: Power supply interface

Input protection for the system will consist of a P-Channel MOSFET, used for reverse input protection, as well as a 0.5A Fuse for short circuit protection of the power supply. A 1000µF capacitor will be added to smooth out the incoming power. A USB voltage regulator will be used to supply power to the SoC as well as creating the 5V rail.

The USB Voltage regulator used is the usb1002 with the following specifications:

* Buck Charge
* Product size: 33mm x 33mm x 8mm
* Input voltage range: 6-24V
* Output voltage: 5.2V (the output terminal is 5.3-5.4V, the load side is 5.1-5.2V)
* Output Current: 3A (without heat sink, the stability of the output current is 2A)

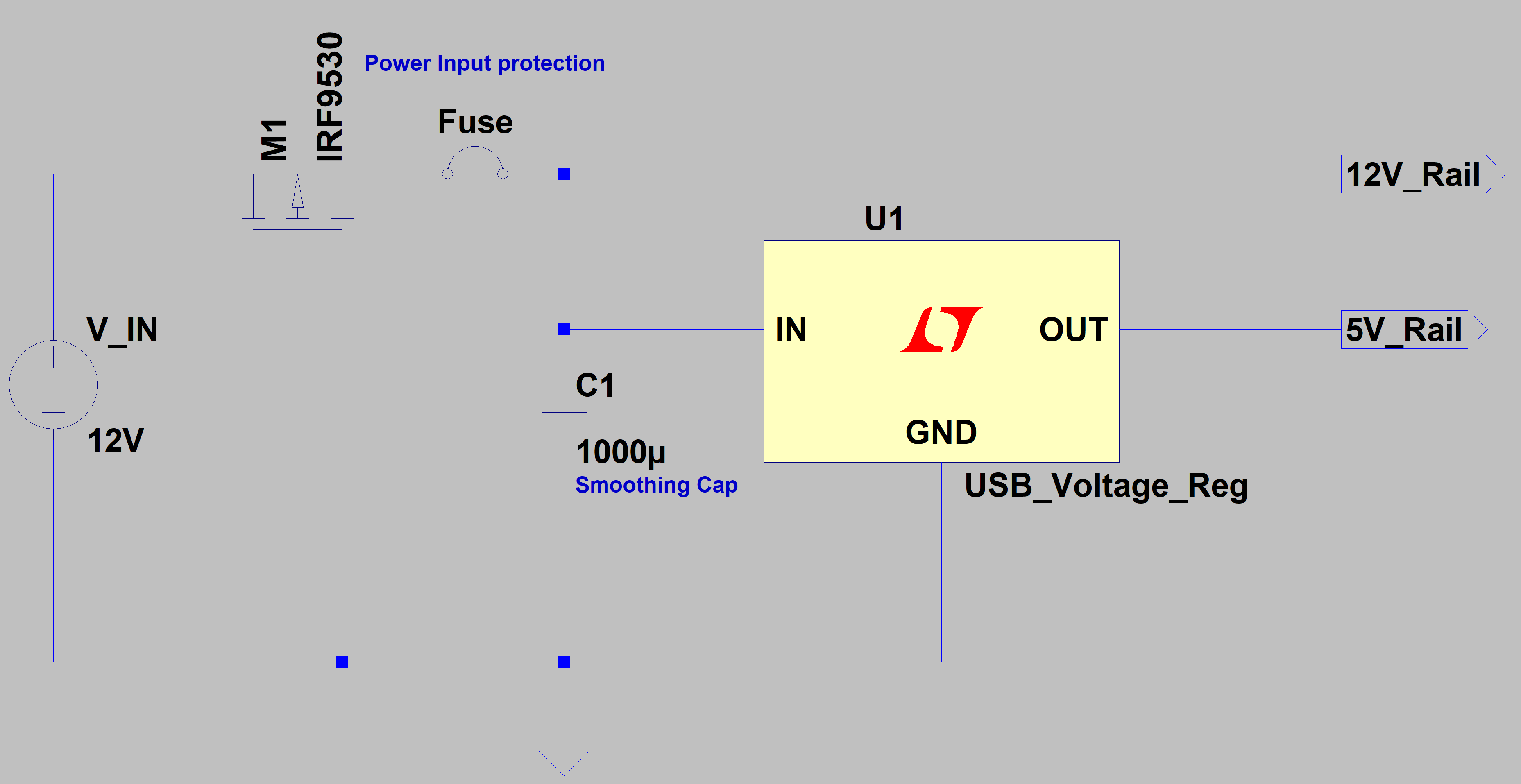


Figure 8: Power Supply

# Unit 2:

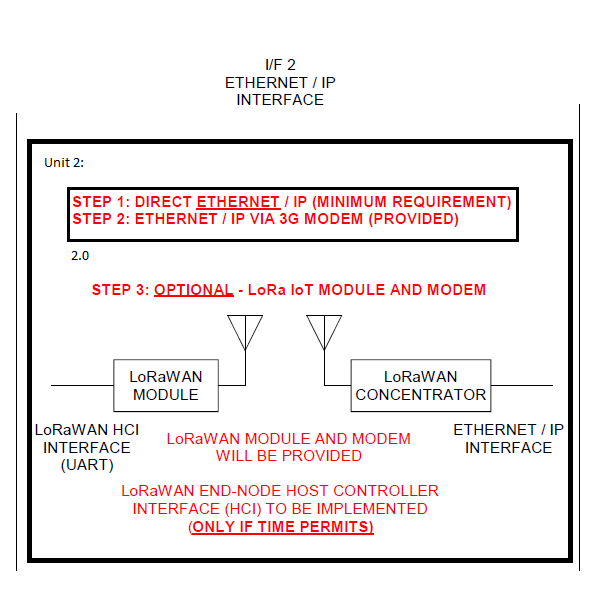


Figure 9: Unit 2 System definition

# 2.1 Network:

Transmission Control Protocol, TCP, to be used to send data from Panel to backend services.

For network connection:

1. The Alarm panel will send out a broadcast command over User Datagram Protocol, UDP. (Port 7000)
2. The Back-end server will then send it’s IP over UDP to the alarm panel
3. A TCP Connection will be set-up between the server and the alarm panel. (Port 7000)
4. Communications will then take place according to the protocol requirements below

# 2.2 Protocol:

Protocol payload fields:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Protocol ID (‘1B’) | Client ID | Date | Time | Section code | Parity |

The payload fields will be “#” separated.

# Alarm Code:

The code can be seen in Appendix A: Alarm Panel Code

The code for the Alarm panel is done in the python language, using the IDLE2 programming IDE. There are 3 modules in die code namely

1. Alarm
2. MatrixKeypad
3. Network

The Alarm module of code acts as the main code for the program, this module includes the other modules. The main module is responsible for most of the Alarm functions.

The MatrixKeypad Module is responsible for the setup and reading of inputs on the 4x3 matrix keypad.

The Network module handles the networking aspect of the system. Sending and receiving data over a specified network.

Auto start-up of the program when the system boots is done by editing the “.baschrc” file with the Linux command in terminal, “sudo nano .baschrc”. At the end of the file the command “sudo python Alarm.py” is added to the file and saved.

.

# Bill of materials:

Table 4: BOM

|  |  |  |  |
| --- | --- | --- | --- |
| Qty | Name | Component | Status: |
| 1 | 4 x 3 Matrix 12 Key Array Membrane Keypad | Keypad | Ordered & Received |
| 1 | 2N2222 | NPN transistor | Ordered & Received |
| 1 | Raspberry Pi 3B | SoC | Ordered & Received |
| 8 | PHBZX79C | Zener Diode | Ordered & Received |
| 4 | HC-SR501 Infrared Motion Sensor | PIR Sensor | Ordered & Received |
| 2 | Reed Switch | Door Contact Sensor | Ordered & Received |
| 1 | usb1002 | USB Voltage Regulator | Ordered & Received |
| 1 | IRF9530 | MOSFET PMOS | Ordered & Received |
| 10 | Screw in wire terminals | Wire terminals | Ordered & Received |
| 1 | Raspberry Pi 3 CPU HAT Heatsink | Heatsink | Ordered & Received |
| 1 | SanDisk 16Gb SD Card | SD Card | Ordered & Received |
| 1 | Brass Standoff spacers | Mounting hardware | Ordered & Received |
| 1 | Strip board | Prototyping Board | Ordered & Received |

# Alarm panel design:

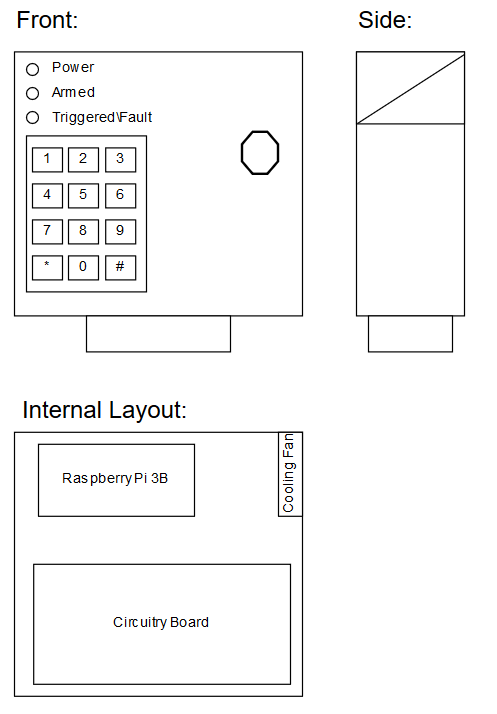


Figure 10: Alarm panel Final Design

# Alarm Panel construction:



Figure 11: Alarm panel Constructed

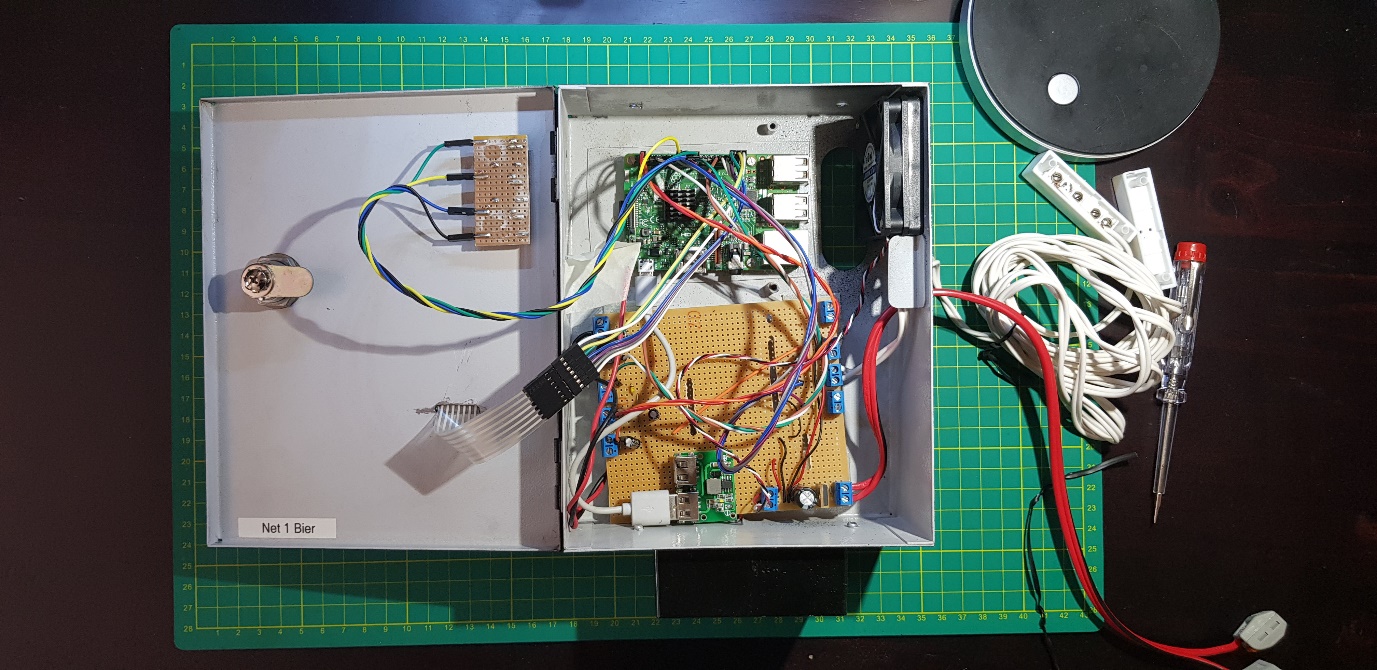


Figure 12: Alarm panel (Inside View)

# Physical Test/ measurements:

Physical simulations will be done on the system to ensure the requirements are met and that the system is in working condition.

Die system will also be reviewed by other members of the group before physical simulations to ensure a working system with the requirements fulfilled.

Test points will also be added to the system to measure voltages, to ensure all aspects of the system is working as intended.

Development Stage Testing:

Table 5: Development Testing

|  |  |  |
| --- | --- | --- |
| Action | Expected Results | ✓/🗶 |
| Program started | Terminal shows program has started and heartbeat signal is shown on the power status LED | ✓ |
| Buttons are Pressed on the Keypad | Terminal shows the button pressed | ✓ |
| The “\*” button is pressed in idle mode to enter armed Mode | The terminal displays that the system is arming, when the system is armed the terminal will display armed, and a heartbeat signal will show on the Armed status LED | ✓ |
| Moving the reed switch (DCS) while in armed mode | The siren starts ringing. The terminal displays a trigger on the Reed sensor. The Triggered status LED starts flashing | ✓ |
| Moving the PIR sensor while in armed mode | The siren starts ringing. The terminal displays a trigger on the PIR sensor. The Triggered status LED starts flashing | ✓ |
| Send a TCP message from SoC to PC | The message is received from the SoC on the PC through TCP | ✓ |

Implementation stage testing:

Table 6: Implementation Testing

|  |  |  |
| --- | --- | --- |
| Action | Expected Results | ✓/🗶 |
| Power up | The system run through the setup phase and the heartbeat signal is shown on the power status LED. The system is then in the idle state. |  |
| The “\*” button is pressed in idle mode to enter armed Mode | A buzzer will start sounding and after 5 second the system will enter the armed state. A heartbeat signal will show on the Armed status LED |  |
| Moving the reed switch (DCS) while in armed mode | The Triggered status LED starts flashing. The user is then given time to disarm the system. After the time has elapsed (20 seconds) and no passcode has been entered or the incorrect passcode has been entered. The siren starts ringing. A TCP message is sent showing a Door switch trigger has occurred |  |
| Disarming the system when the Reed Switch (DCS) has been triggered | The system turns of the armed status LED, and the Triggered status LED. Thereafter the system returns to the idle state |  |
| Triggering the PIR sensor while in armed mode | The siren starts ringing. The Triggered status LED starts flashing. A TCP message is sent showing a PIR sensor trigger has occurred |  |
| Enter the passcode + “#” while in armed state | The system turns of the armed status LED and returns to the idle state |  |
| Enter the passcode + “#” while in triggered state | The system turns of the triggered status LED and returns to the idle state |  |

# Tools required for instalation:

* Handheld drill + drill bits
* Mounting hardware
* Wire striper
* 4core& 2 core insulated wires
* Screwdrivers

# Health & Safety considirations:

* Construction hazards:
  + Electrocution due to incorrect insolation of components
  + Soldering burns
  + Cuts from sharp edges/ wires
* Final product safety concerns:
  + The system must be grounded to prevent electrical shocks
  + No loose wire should be accessible in the system
  + The system must be tamper proof
  + The system must not be accessible to children
  + Wire must be inspected for damage, if wires are damaged the wire must be replaced immediately.

# Tools used to

# enhance productivity:

* Software tools:
* Excel: For project management, Bill of materials, Trade of decisions
* LT spice: Drawing of circuits and simulations of circuits
* draw.io: Software used to draw the State diagram and the flowchart
* GitHub: used for version control of software
* IDLE2: python IDE used on the SoC to program the SoC
* Hardware tools: (used for construction of alarm panel)
  + Dremel Rotary tool (cutting and grinding for enclosure)
  + Soldering iron
  + Handheld drill
  + Wire striper
  + Screwdrivers