# UNIVERSITY OF ALASKA ANCHORAGE

College Engineering

Computer Science and Computer Engineering


# Home Alarm System With Raspberry Pi


by

**Gabriel A. Pérez Cortés**

Supervisor:

**Prof. Kirk Scott, PhD**


A CAPSTONE PROJECT SUBMITTED TO THE DEPARTMENT ENGINEERING
, AT UNIVERSITY OF ALASKA ANCHORAGE, FOR THE
DEGREE OF COMPUTER SCIENCE.


Anchorage AK, March 2015.

Gabriel A. Perez Cortes

gaperezcortes@alaska.edu

Version 0.0

# Abstract

The target of this capstone is to present a new new take on a home alarm system. The development focuses on making an open source implementation with the use of a Raspberry Pi. My work focuses on bringing a flexible and low-cost approach to a home alarm system which can also be controlled from a remote android device via Internet without the user needing to do much on his/her side. The requirements, hardware, and tools needed are explained. The presented system needs to endure indefinite use and be reliable.

The scope of the project relies on following the agile methodology and management to ensure iterations are made every so often. In a chronological manner, all the important pieces of the project are done until enough of the system is up and running to make iterations until a full system is complete. A lot of the system has been tested for performance, issues, so as the project and testing has progressed there have been several problems encountered, things learned the easy and the hard way. With all the testing made there have been some conclusions pointing towards the production of a new version that takes different angles with some aspects.

# Contents

# List of Figures

# ACKNOWLEDGEMENTS

Thanks for the help, Dr. Scott. You have taught me plenty of programming and pointed me in the right direction to continue honing my skills as a computer scientist.

# Chapter 1
# Introduction

---

## 1.1 Introduction

This project will be a home alarm system. It will be made using a Raspberry Pi, an RF receiver, some RF transmitters. The primary purpose will be to alert the homeowners with some kind of sound and maybe push notifications to registered Android devices. It will be entirely open-source and will be released under the GNU GPL.

Home break-ins are a common crime in today's world. When a person breaks into a house, having an alarm could easily deter the criminal from continuing their crime. Alerting a homeowner is crucial when they're in their own home allowing for a better handle on the current situation.

Some statistics:
In 2013, there were an estimated 8,632,512 property crime offenses in the nation.
In 2013, the rate of property crime was estimated at 2,730.7 per 100,000 inhabitants.

Of all property crimes in 2013, larceny-theft accounted for 69.6 percent. Burglary accounted for 22.3 percent and motor vehicle theft for 8.1 percent.
Property crimes in 2013 resulted in losses estimated at $16.6 billion.

This project is intended for those Raspberry Pi, tech savvy, and DYI enthusiasts. This alarm system is intended as an open source project, therefore, people will be able to use this and modify to their liking for a do their own project with its own features. Maybe we could see expansions of the project with additions such as security cameras, automated control of house lights, locks, etc.

## 1.2 Overview

The Raspberry Pi will the main component here for the user. It will be connected to a 433 MHz receiver, a WiFi module, and a touchscreen (eventually). The receiver will be able to get signals from any 433 MHz transmitter. These transmitters are each connected to an Attiny85. This microcontroller will send via the transmitter a specific code (like a unique ID) for differentiation purposes when

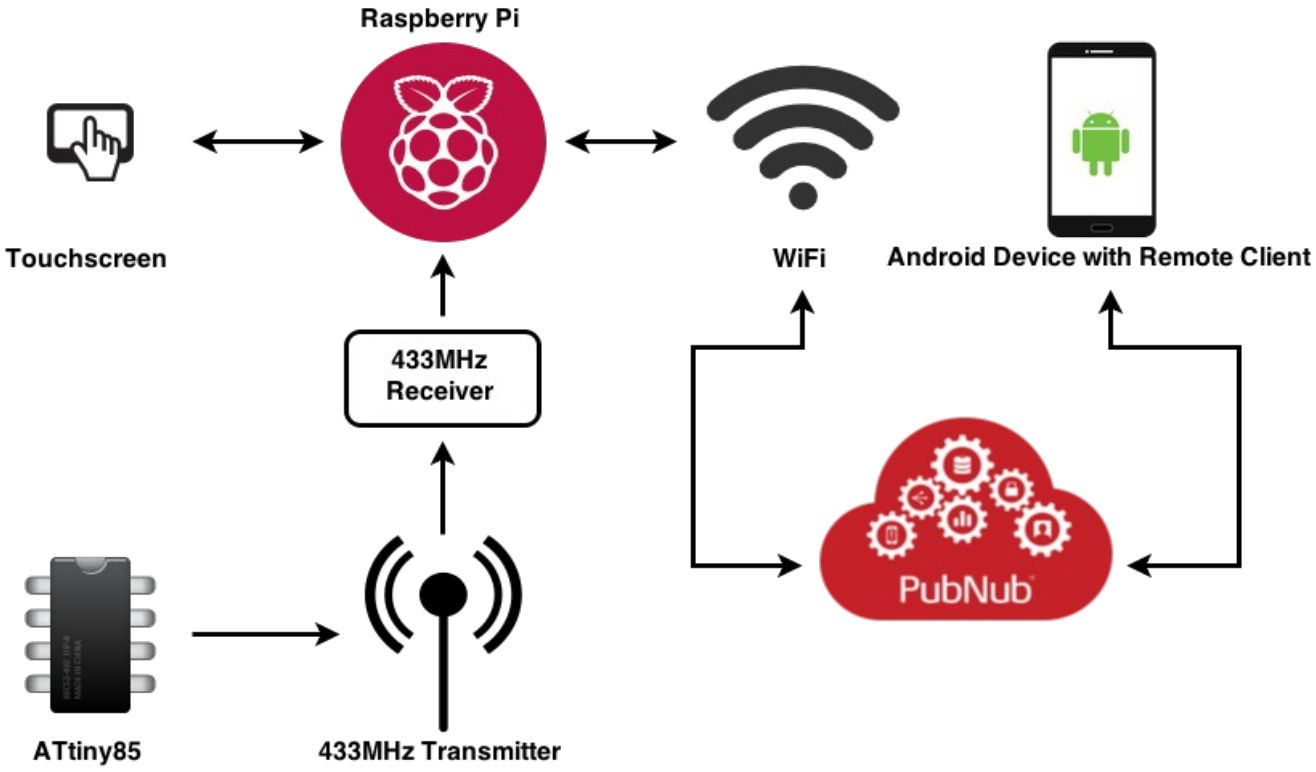



*Figure 1.1: Simple System Diagram*
received by the host. The wireless communication can be used enabled for internet communication

between Android devices and the Raspberry Pi. Both will handle messages via PubNub. For ease of use, the only thing needed by the devices is the PubNub application's publisher and subscriber keys; consequently, the user will not need to handle any network-side configurations such as setting a static IP address in the router's DHCP, and port forwarding. PubNub allows for encrypted messages, so security between the devices will not be a problem. The Android client will be able to control the alarm system (locking, unlocking, modifying certain settings) from anywhere with an active internet connection.

## 1.3 Hardware

Raspberry Pi (model B)

The Raspberry Pi is a computer the size of a credit card that was developed in the UK by the Raspberry Pi Foundation which costs 35 USD. It's intention was to promote computer science in schools. The RPi model B has a 700 MHz single-core processor with ARMv6 architecture with 512 MB RAM. It has a low power consumption of 3.5 watts, uses SD cards for non-volatile memory. Its video outputs can be done via RCA or HDMI (preferable for obvious reasons). Some of the operating systems this device can run are Linux distributions (Raspbian, Debian, Fedora, Arch), RISC, FreeBSD, and NetBSD. The RPi shines due to its cheap cost and big community, but the best feature it has to offer is its GPIO (General Purpose Input/Output). It opens a whole new world of opportunities to DIY hobbyists. Some of the many projects out there include home automation, home media centres, personal servers (HTTP/FTP/SMTP), cloud storage, phones, time-lapse cameras, jukeboxes, karaoke machines, 3D printers, drones, and robotics.

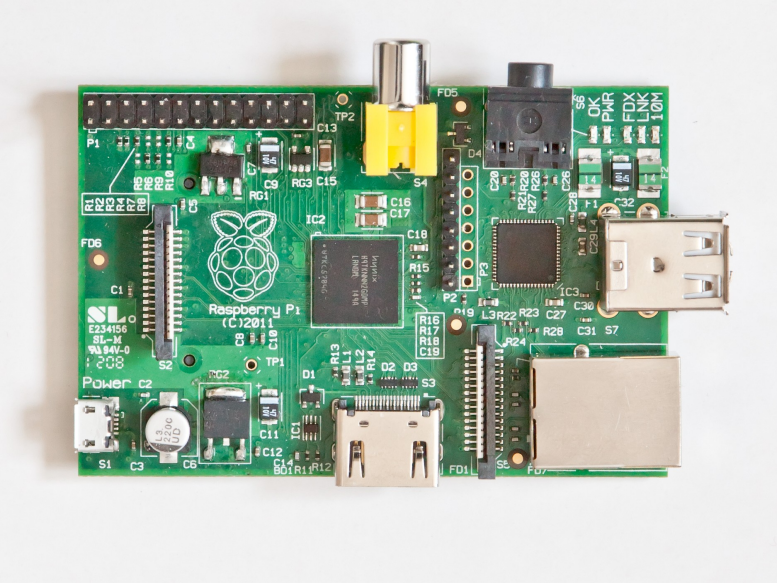



*Figure 1.2: Raspberry Pi Model B*

Arduino Mini

The Arduino Mini is a ATmega328 microcontroller based on the ATmega168. It comes with 14 digital I/O pins,8 analog Input Pins, 32 KB flash memory, 2 KB SRAM, and a clock speed of 16 MHz. Its primary purpose in this project will be to program the ATtiny85 microcontrollers. The Arduino Mini could be used for the same purpose as the ATtiny85's, but it turns out that these are much cheaper in comparison.

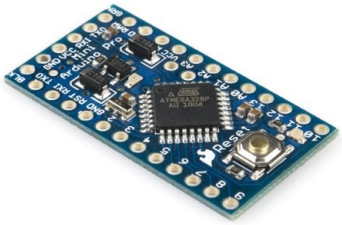


*Figure 1.3: Arduino Mini*

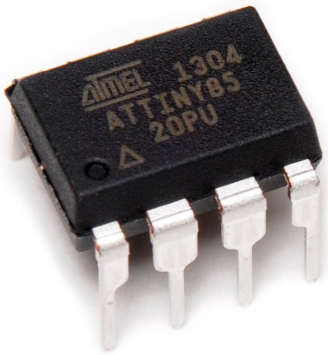



*Figure 1.4: ATtiny85*

ATtiny85

The ATTiny85 is a high performance / low power microcontroller. It comes with 6 GPIO (General Purpose Input/Output) pins, 8KB of memory, 512B of SRAM, and has a clock speed of 20 MHz. This chip is low cost and with low power consumption (between 2.7-5.5 volts) it can execute powerful commands, so it has a great balance of power consumption vs performance. Excellent to pair up with a transmitter and send bits of info.

433MHz RF

For development purposes a 433MHz RF receiver with 433MHz RF transmitters will be used. A pair of a receiver/transmitter can be found online relatively cheap ($2-3 on eBay).

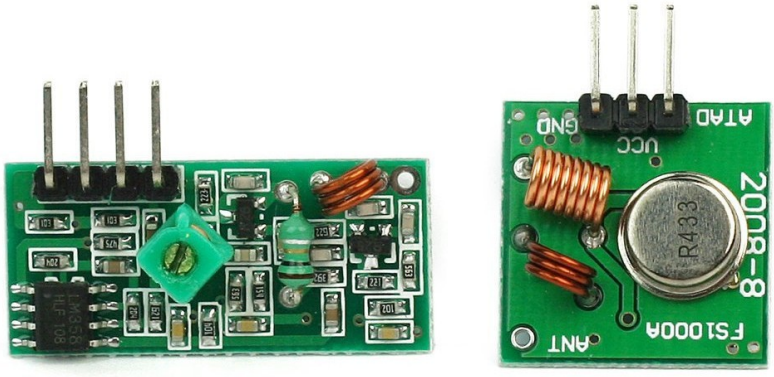



*Figure 1.5: 433 MHz receiver (left) and transmitter (right)*

## 1.4 Tools

Arch Linux

Arch Linux is a Linux distribution composed mainly of free open-source software. Its minimalistic approach makes it the perfect Linux to use with the Raspberry Pi due to the constraint in computing resources. Since Arch comes bare-bones, it is up to the user set up the system from the ground up or use a pre-built image; consequently, this makes Arch very flexible in terms of keeping the system light by installing only necessary software.

LXDE

LXDE or Lightweight X11 Desktop Environment is a fast and lightweight desktop environment for mainly Linux systems. Because LXDE uses less computer resources than most other desktop environments, it is the best suited for the amount of computational power offered by the Raspberry Pi. LXDE is so minimal on resources that it can be run on a Pentium II CPU and will take about 45 MB of RAM on i386 machines, so a system with only 128 MB of memory can support it.

Qt4

Qt4 is a cross-platform toolkit mainly used for developing applications with graphical user interfaces. The cross-platform aspect is made so little to no change needs to be done to the code for it to be compiled and ran natively. Qt4 contains a full set of GUI widgets as a developer would expect from any of the other well-known toolkits such as the ones provided in Visual Studio and Java. This is available for commercial and open source licenses.

Qt Creator

A great cross-platform IDE to develop C++ applications that use Qt as a front-end GUI. It supports the standard features of the better-known IDEs and comes packed with many tools that will help expedite the development process.  Qt Creator supports both the Qt4 and Qt5 toolkits.

433Utils

The 433Utils is an API designed to assist a developer in the usage of 433 MHz RF transmitters and

receivers with Arduino boards and the Raspberry Pi.

PubNub

PubNub is an easy-to-use realtime communications service. There is an API for all the major SDKs and popular systems being used in the present.

Arduino IDE

An open-source IDE that facilitates writing code and uploading it to Arduino boards. The IDE is written in Java and can run in Windows, Mac, and Linux systems.

# Chapter 2
# System Integration and Modeling

---

## 2.1 System Overview

The system setup is relatively easy considering there's only three main components: a Raspberry Pi, 433 MHz RF receiver, and 433 MHz RF transmitter connected to an ATtiny85. The Raspberry Pi is connected to the receiver on the left, which listens for signals sent by any 433 MHz RF transmitters. These transmitters will send a message periodically out (a heartbeat). All the hardware communication is handled by the software run by the Raspberry Pi and the GPIO.

The RPi plays the most important role as it's a small credit-card sized cheap computer that can perform very powerful tasks. Having a 700 MHz processor, and GPIO allows for parallel tasks to happen. The Raspberry Pi will be handle all the RF communications and facilitated with the use of the 433Utils by reading the input provided by the RF receiver as a series of bits. A front-end GUI that contains some input/output widgets responsible of carrying out commands by the user and visual responses. Also, local TCP server and a PubNub relay running in parallel receive remote commands from any of the android devices attached, and send messages to them when any event in the alarm happens such as tripping, locking, unlocking.

The sole purpose of an Arduino Mini in this project is to program our ATtiny85 microcontrollers, although, it could be used to handle the RF transmissions, if the RPi needed extra computing power.

## 2.2 Hardware Components

Chapter 1.3 provides an introduction to the relevant hardware along with their specifications, prices, common uses, and reasons to why they are used for this project.

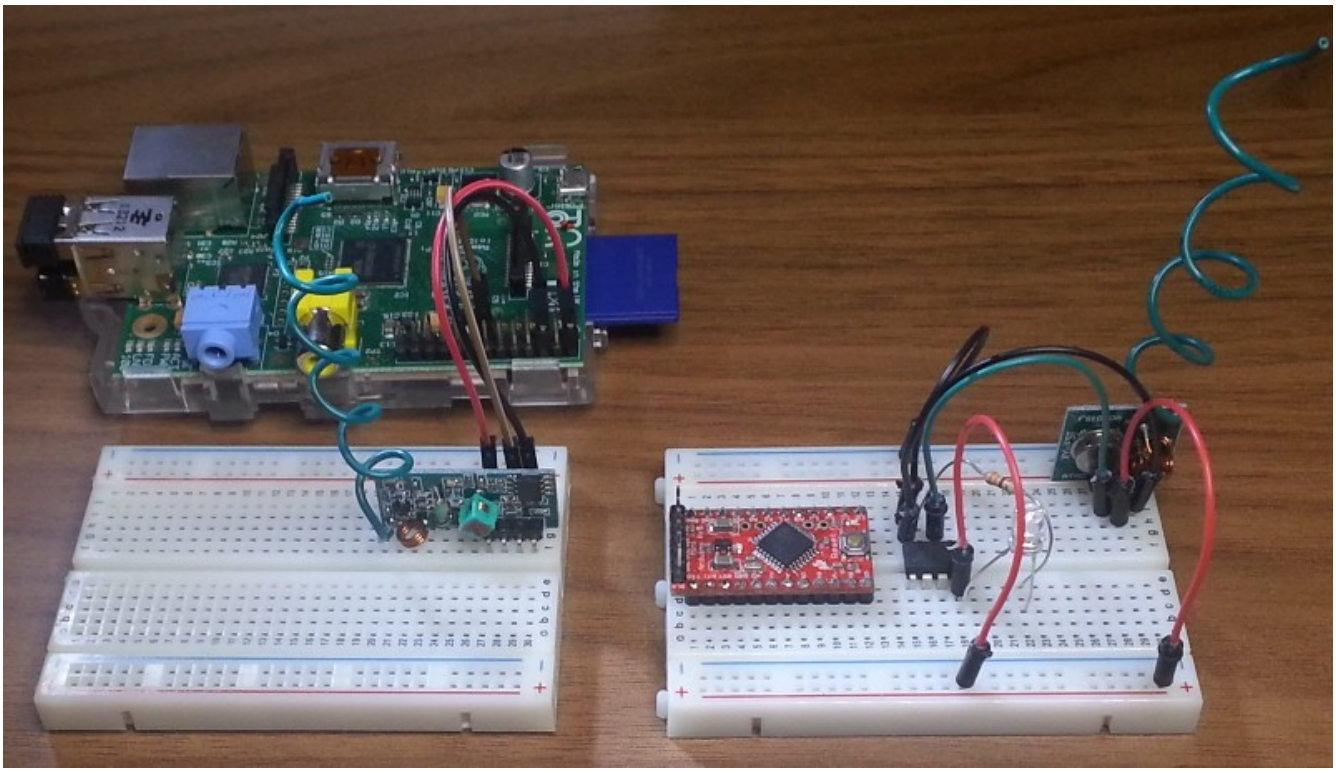


*Figure 2.1: Hardware Setup (RPi top, RF receiver left, RF transmitter with ATtiny85 right)*

The Raspberry Pi will supply power to the RF receiver via its GPIO 5v and ground pins. The communication between these two is done with a single GPIO pin. The RF transmitter and Attiny85 will both need some kind of external power source. A battery could be used, but for testing purposes an external 5v power supply will be attached to the breadboard. Data communications is also done via a single GPIO pin from the microcontroller.

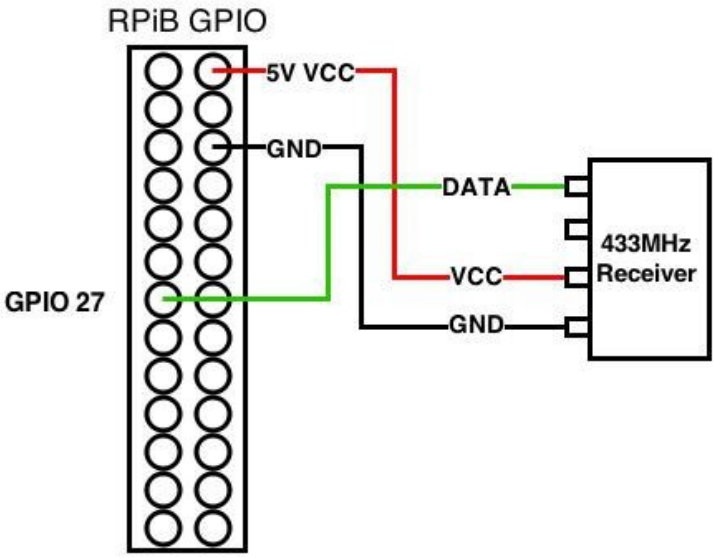



*Figure 2.2: RPi Hardware Connections*

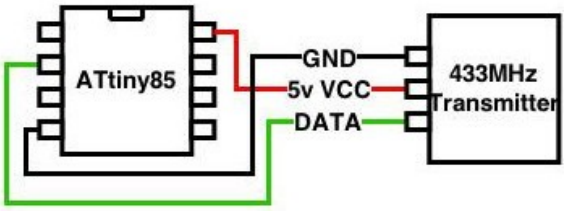



*Figure 2.3: RF Transmitter Connections*

## 2.3 Software Design Overview

**Timeline explained:**

The timeline is split into each major component. The system logic and front-end GUI are estimated to be the two biggest pieces of the puzzle. The extra portion is small useful additions that could be done as long as time allows. Any mistiming or delays in development time will result in getting only the basic product completed.

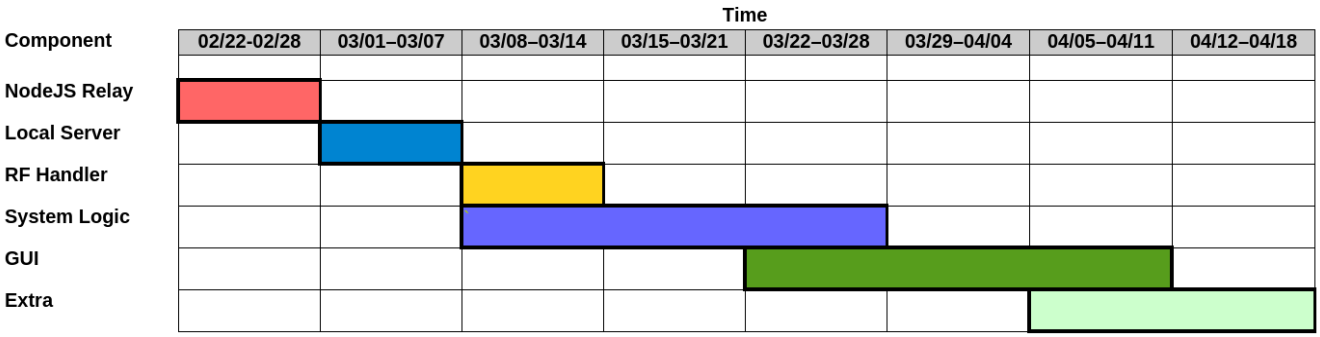



*Figure 2.4: Estimated Timeline of the Project*

**Components:**

The system is broken down into several software components which almost all play an equally important in role for the alarm:

A NodeJS relay server will be in charge of handling PubNub messages. It is relatively easy to set up due to the simplicity NodeJS brings when handling websockets, APIs that use websockets. The reason for setting up the PubNub server like this is because there's a library required by the C++ version of the API, which is buggy and notorious for compiling issues, setup problems, and lack of documentation.

A local NodeJS server that runs in parallel with the PubNub relay. This server will be in charge of allowing a local connection from the Alarm System via TCP and forwarding messages to and from the PubNub relay.

A portion of the system is in charge of handling the information received by the RF receiver attached to the RPi. The heavy work is done by the 433Utils library found online for both RPi and Arduino. The handler uses the 433Utils API to constantly read input provided by the RF modules.

The Graphical User Interface (GUI) is responsible for handling user input, and visual responses. The basic GUI will contain a pin pad just like any alarm system would, and a status bar showing the current state (locked/ unlocked/tripped). It will implement the MVC design pattern for code cleanliness and ease of use.

The system logic is the core of the project and probably the biggest part for several reasons. It ties together the graphical user interface, remote communications, RF handler. The logic is responsible for handling commands from the GUI, accepting credentials, parsing remote commands, and if time allows, extra functionalities such as system logging, and zones.

**Useful Tools:**

The Qt Creator is a full-scale Integrated Development Environment (IDE) built for designing software written in C++ and Qt toolkits. This IDE provides a set of tools to expedite the development process:
A built-in User Interface (UI) designer which is very straight-forward and provides all of the Qt toolkit widgets that are available for your particular version (Qt4 or Qt5). The UI file is incredibly easy to import and get running into C++ code, however, the IDE creates this for the user on UI creation.
Qt Creator supports for creation of compiler profiles. Since Qt supports cross-compiling, if need be, a the program could be compiled ahead of time for the Armv6 architecture used by the RPi or any other supported architecture.
A built-in debugger that allows the user to see memory, jump in and out of methods, and step through code line by line.

Valgrind integrated into the IDE. This is a powerful tool that checks aids the developer to find any misuse of memory (memory leaks) in C or C++ code. As long as the code is compiled with debug flags, Valgrind can do its work and pinpoint the source of memory issues.

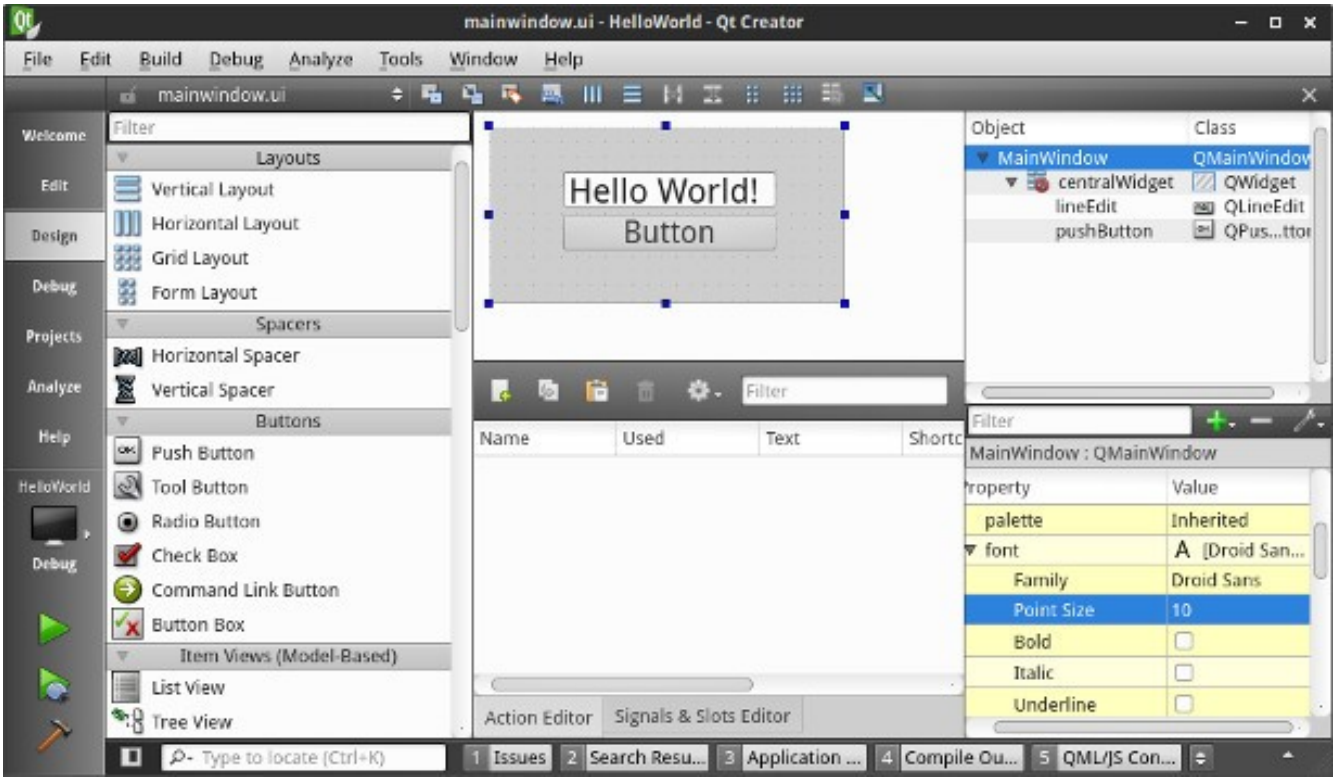



*Figure 2.5: Qt Creator's UI Designer*

## 2.4 Agile Methodology

The Agile methodolody is adapted for the project development. Unlike older more traditional development methods, Agile provides flexibility and direction throughout the products lifecycle. It provides the ability to re-plan and optimize a code (iterations), unlike other methods which a product is committed to before it's even coded. Due to all the flexibilities, a developer or development team wouldn't crash and burn because something was miscalculated in the planning of the entire project, unlike the waterfall method which relies on that every requirement will be identified and carefully planned before any coding occurs. Iterations help evolve a product rather than just developing the given project. Each iteration means a developer can look at the current stage of the product and plan any changes, fixes, anything necessary to better the next, since development is much easier as it is hard for any coder to envision the product's final stage from the very beginning.

# Chapter 3

# Design and Testing / User Interface

---

## 3.1 User Interface

**Explanation**
User Interface is a very important aspect of this project. An alarm system needs some kind of interface for the user/home owner to interact with. This system particularly won't use a membrane keypad with some kind of basic LCD display, so a decent GUI will be developed to create a good user experience that can be expanded with future updates. The goal is to eventually (most likely after the deadline due to time constraints) a more-than-just-a-keypad full-blown GUI with settings, options, which will be of an appropriate proportion for a medium-sized touchscreen.

**Why The UI Works**
The user interface follows an intuitive conceptual model built from the following:

Visual Affordance
For visual affordance the perceived and actual fundamental properties of the objects should determine how they can be used. So anything put in the UI should match some kind of object the user is acquainted with which in our case it resembles a typical alarm system.

Transfer Effects
The transfer effects are pretty obvious. An alarm system has a number pad along with some other buttons that provide certain functionalities. This UI replicates exactly that for an intuitive, easy to use experience.

Casuality
Casuality is some kind of representation of anything after an action. All the buttons in the UI provide a visual feedback when pushed and will emit some kind of beeping sound, so the user definitely knows something is going on. The panel in the top is there to show the status of the alarm system, therefore, after any action done by the user, there will be an update message according to the alarm's current status.

Visible Constraints
Because of the UI's basic structure, the user will be limited to only being able to push some buttons, but won't really be able to do anything other than that.

Mapping
The natural mapping of the UI takes advantage of physical representations. The UI has a number pad on the left side exactly how many other physical forms of keypads. The right side contains some self-explanatory buttons. These button groups combined create a natural intuitive layout.


*Figure 3.1: A Traditional Home Alarm System*

The user interface this project is using is done with the Qt 4 toolkit. The Raspberry Pi provides support for this toolkit. Qt 4 was picked mainly due to preference, the software support for GUI creation and use, and due to the widgets support Cascading Style Sheets (CSS) which provides the power to create very nice look and feel for GUIs. The CSS support adds a lot of flexibility and removes the need to programatically do changes to the visual aspects (i.e. widget aesthetic changes on events, formatting the entire program to look uniform, program-wide fonts).
The intended use for this UI is for the user to be able to push buttons according to whatever action they want to take. They shouldn't need to look at many instructions to get the system up and going. After all, keeping it simple is usually the better choice. As a user, I wouldn't want to see cryptic messages, or buttons that mean absolutely nothing.

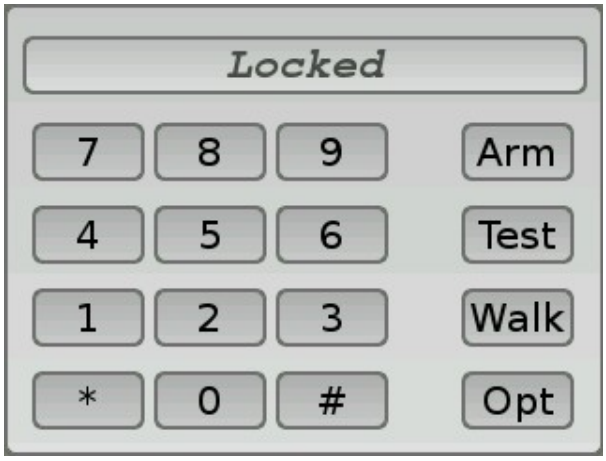



*Figure 3.2: The First Iteration of the GUI*

## 3.2 Design

**Logic of the System**

The design of the alarm system is meant to be as straight-forward as possible. The figure 3.3 shows the logical path the system goes through in order to function. The NodeJS server contains the Pubnub logic and a TCP server that waits for the alarm system that connects to it locally. The next run independently from each other, so these are multi-threaded. The alarm's logic is updated via RF switches tripped, via GUI updates (ie the user disables the alarm with the pin number) or via a remote message pushed. When the alarm is triggered, the alarm will sound, the GUI will be updated, and a message will be sent via PubNub to alarm the remote clients about the incident. Once the user disarms the alarm, the model will become updated along with stopping the alarm.

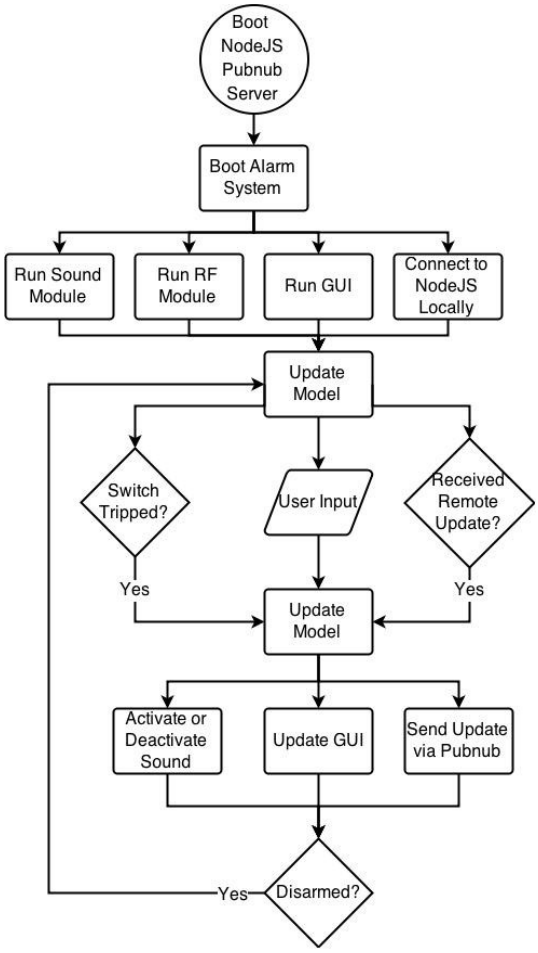



*Figure 3.3: Flowchart of System's Logic*

## 3.3 Testing both User Interface and System

**Testing Overview**
To verify that the system is working properly, a series of tests are going to be done. The GUI will be tested for various inputs button presses, to see if any kind error arises. The RF transmissions will be tested for correctness and fault. The alarm system itself is gonna be let running for an extended amount of time to monitor system usage. The Valgrind tool will be used to check for memory leaks. All of these test combined should be enough to prove that the alarm system is working properly.

There isn't much to test the GUI for because of its basic status. The GUI tests were done by testing all the buttons. Constantly pushing input to see if anything can be found that is not desired. The buttons will limit what a user can do. Preferable there should be no more option than the usual to avoid any kind of problems that could render the running system inoperable.

**The Tests**
Turning off and on a transmitter
Testing the transmitters and receivers functionality and reliability by watching the input over-time and testing the transmitters on and off periodically.

Uptime test
Tested the program by leaving it running for a many hours several time to test false alarm triggers, memory leaks by running the top command and monitoring the Rpi's CPU load and memory usage, and no random crashes later or runtime issues.

Testing CPU priorities
Tested the program for under different task priority levels along with the default. Even though the Rpi won't be running anything other than the alarm system on top of some other OS/user processes, it is still preferred for it to have priority over the others to mitigate loss of missed RF messages. The command used for this is *nice --10 {command} where --10 gives priority over 0 the default (max priority = -20, lowest priority = 19).*

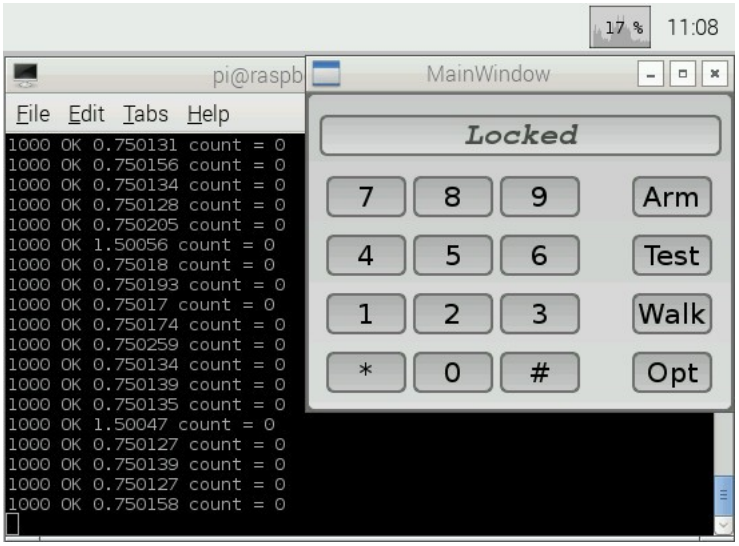


*Figure 3.4: Uptime Testing for CPU load and errors*

Verify Integrity of RF Transmissions
The RF transmitters and receivers are tested before being implemented into the project. After programming the Attiny85's with the transmission code, an RF sniffer is run by itself to make sure there are no hiccups in the messages being sent out. Each Attiny85 sends a different ID (ie 1000, 0111, etc...) for identification, so ensuring unique ID's are sent is a must.

Valgrind
Valgrind is a very useful tool that helps the developer identify memory management and threading problems. The tool is integrated with the Qt Creator to provide fast alongside Qt Creator, however, it is used from the command line in the **Rpi** because the program is compiled with itself.

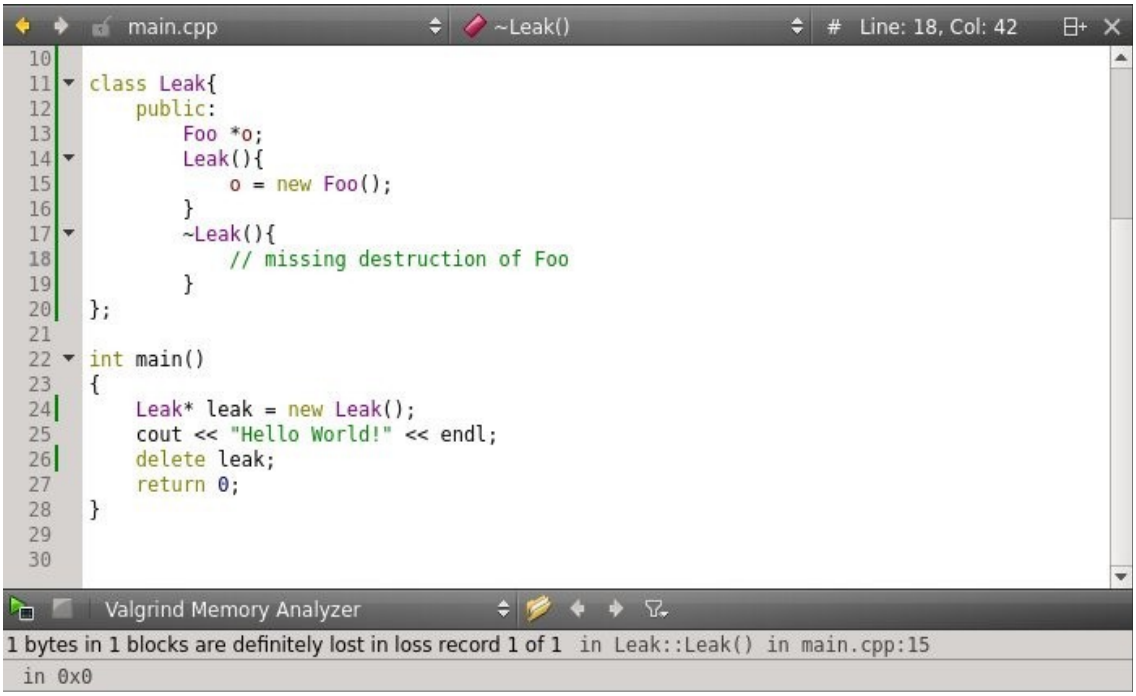



*Figure 3.5: A Valgrind Positive Memory Leak Example*

# 3.4 Agile Management

The agile coding methodology is an approach to developing software through iterations. Basically writing code from a skeleton to a full-blown program through updates and changes. In comparison, the agile management approach deals with an oversight of the project and the agile development team. So the project manager will facilitate the agile process in the developing of a project by: capturing and keeping track of data, analyzing it, and adjusting timelines accordingly; acting as a liaison between the team and the customer; promotes good engineering practices. The manager has a leadership, not a supervisor role.

# Chapter 4
# Setup And User Manual

---

## 4.1 Introduction

This is a manual with the instruction on how to set up the alarm system with a Raspberry Pi.
You will need:

- Technical skills
- Patience
- A Raspberry Pi 2 B (preferably), or Raspberry Pi B/B+
- An SD card for your OS
- Some kind of way of connecting your SD card to a computer
- An Arduino Uno/Mini
- PiTFT 3.5” Touchscreen for Model B or B+ (Model B works with B+ and RPi2)
- An Internet connection

The guide will run you through installing the operating system, changing some necessary settings, connecting to a wifi network, grabbing the necessary software, compiling, and running the project.

# 4.2 Installing The OS

### Get Raspbian

Raspbian is a Debian-based Linux distribution specifically made for the Raspberry Pi.

Download at: http://www.raspbian.org/RaspbianImages

### Flashing image file to SD card

For Windows Systems:

Download Win32 Disk Imager at http://sourceforge.net/projects/win32diskimager/ and flash the image file to the SD card with the provided GUI.

For Linux Systems:

Use the dd tool from the command line to flash the image like in this example: `dd if=/path/to/image.img of=/dev/sdx bs=1M` where if is the image file, of is the path to the SD card (usually sdb, depending on how many other devices have been mounted prior)

# 4.3 Post OS Install

### Raspbian Settings To Change

The first thing to do right after install is boot the system then log in with the default Raspbian credentials.

Run the command `sudo raspi-config` then this menu will show:

1. Select Expand Filesystem.
2. Select Change User Password and change the default password to something else
3. Select Enable Boot to Desktop/Scratch and then select Desktop Log in as

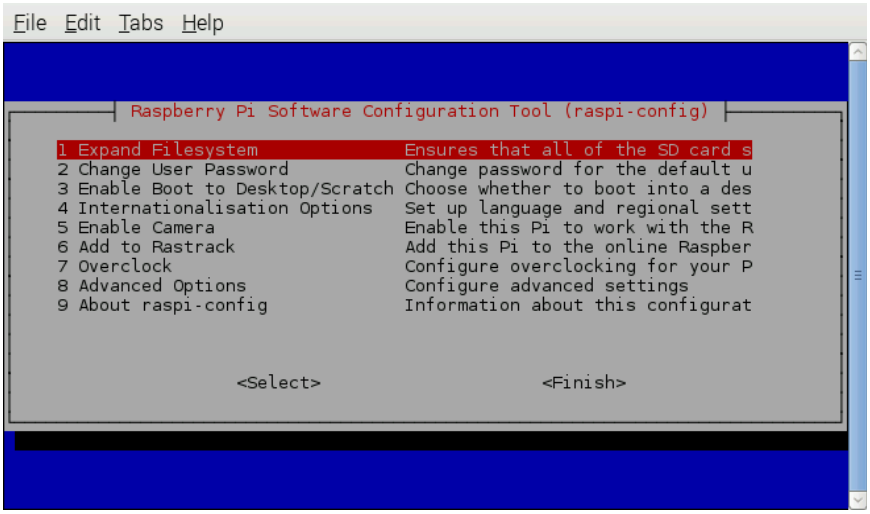



*Figure 4.1: raspi-config*

user 'pi' at the graphical desktop. This will enable the Raspbian OS to automatically log in as the user selected.

4. The Raspbian image by default will have a UK keyboard layout. Select Internationalisation Options, then Change Keyboard Layout, and follow the options to choose the desired layout.
5. Once done making all the required/desired changes. Select <Finish>.
6. Enter sudo reboot to restart the Raspberry Pi (required for some changes).

**Connecting To A WiFi Network**

If the Raspberry Pi will be connected via ethernet, then skip these steps.

We will need an internet connection in order to download or update the packages available for Raspbian, so follow these steps to connect to your wireless network:

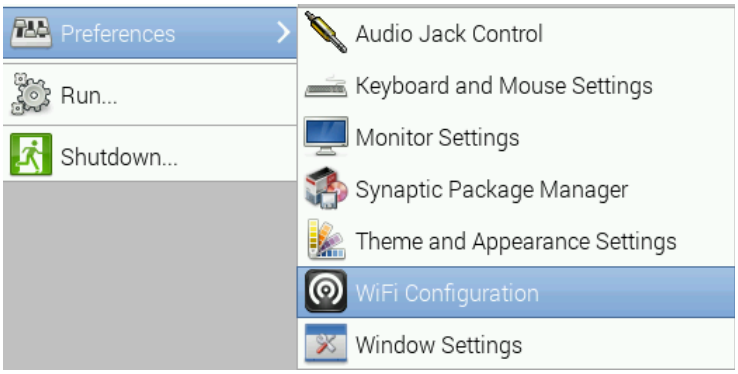



*Figure 4.2: Getting to WiFi Configuration*

1. Access the WiFi Configuration tool from the main menu.
2. Hit the Scan button to bring up a list of available networks.
3. Double click the desired network.
4. Click Connect.

**Grabbing The Needed Software**

There are several programs and libraries we will need to download and install to our OS before we can do anything:

1. Open a terminal.
2. Update the repository versions with sudo apt-get update.
3. Upgrade the system with sudo apt-get upgrade.
4. Install the Qt4 compiler with sudo apt-get -y install qt4-qmake.
5. Install WiringPi to the system:
    - Clone the WiringPi repository git clone git://git.drogon.net/wiringPi.
    - Enter cd wiringPi to go into the directory of the cloned repo.

- Enter `sudo ./build` to build and install the library to your system.
- Enter `cd` to go back into your home directory.

**Prep The PiTFT 3.5in**

If you don't want have SSH enabled then leave this step for very last. Setting up a PiTFT Touchscreen is fairly easy with the use of some scripts provided to facilitate job, removing the need for manual task. Go to https://learn.adafruit.com/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi for the easy/advanced guides on how to get the screen up and running.

## 4.4 Setting Up The Alarm System

**Setup The Code**

1. Download the repository with `git clone https://github.com/gaperezcortes/CSCE470.git`.
2. Put your PIN in the code (no other way at the moment)
3. To setup remote access, set up the PubNub keys:
   Open the Server.js file in the nodejs folder with the default text editor, or any other, and replace the following your PubNub keys:

```
var pubnub = PUBNUB.init({
  publish_key   : "pub-x-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  subscribe_key : "sub-x-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  ssl: true
});
```

4. Compile the code:
   - Open a terminal
   - Enter `cd CSCE470` to go into the directory of the cloned project repository
   - Enter `make` to compile all the code
   - Go grab a cup of tea (or whatever beverage)
   - Program is ready to be launched
   - If desired run now the program (will use *sudo* since RF needs SU privileges to work) by entering `./launcher` into the terminal

**Autostart Program**

The alarm system should be able to boot automatically as soon as the X server and desktop environment are up and running. It would be unnecessary to start the alarm system manually and basically renders it useless if for any reason the Raspberry Pi would reboot (power outage, manual reboot, some kind of crash).

To enable the alarm system to autostart:

1. Add the launcher as a desktop shortcut to the autostart folder (~/.config/autostart)
2. Ensure that the RF receiver data pin is hooked up to the GPIO 27 also known as PIN 13 (RPi A, B, B+, RPi2)
3. Reboot the system to verify alarm system boots on startup

# 4.5 Using The Alarm System

The alarm system is currently at its most basic state. It only allows for enabling or disabling the alarm (more features to be implemented in the future).


*Figure 4.3: Armed*

**Arming The System**

Press the Arm button.

Note: that once the system is armed, nothing can be done with it until it is disarmed.

**Disabling Armed/Tripped Alarm**

Press on the buttons corresponding to the pin then press the pound sign. If the wrong pin was input, press the pound sign to clear the buffer as nothing will happen if on a wrong input.


*Figure 4.4: Disarmed*

# Chapter 5

# Concluding The Project

---

## 5.1 Summary And Reasons

**Quick Summary**

This alarm system uses a Raspberry Pi 2 which brings plenty flexibility and low power consumption allowing this home security system to acquire plenty of potential and ability for later upgrading. Changes can be made to the UI

**Why C++ and not Python?**


*Figure 5.1: C++ or Python*

At first the project was going to be developed with the use of Python 3. The reason for this was preference. I already knew the Python 2.7 and 3 programming language and given all the libraries that are provided with and contributed online by other developers, constructing functions (such as sound, and socket programming) would be much easier to implement in contrast to C++. However, C++ was a must because some one of the libraries for dealing with RF communications was written in C; although, there was a wrapper class for Python, but turned out to be buggy with some broken functions as well. Even though C++ wasn't my strongest language, this project forced me learn more about it and polish my skill, so it definitely was something profitable.

## 5.2 Application To Field

The project is not meant to make a difference in the field of software as there is no research done. However, with the completion of all the basic functionalities of this alarm system, users can take advantage of the code provided to learn or use as an example on the Qt4 toolkit to learn on their own or become familiarized with a toolkit that is not provided with the language itself such as Microsoft Visual C#/C++, Java Swing/FX. This can help introduce Raspberry Pi enthusiasts to the capabilities of the little credit card sized computer and how to use the GPIOs along with external hardware. Since it's all open source and the repository will be available to anyone, a user can take upon him/herself to make their own project with the use of the provided.

## 5.3 Alternate Paths, Future Development, And Final Thoughts

**Things I would have done differently**

The project currently implements the RF communications with the Raspberry Pi. Since version 2 has much more computing power available, this isn't a problem. However, version 1 has very limited computing power in comparison, so running the RF sniffer is very tasking to the point of introducing errors to the system. To address this problem, and arduino handling the RF communications would offload a lot of computational power allowing the Raspberry Pi 1 to run the alarm system flawlessly.

**Future Additions**

These are things that I would implement later on after the semester ends:

- Add system logging to keep track of any events that happened throught time
- Parameterizable features like declaring zones for transmitters
- Update the User Interface to allow for more implemented features
- Use an SMTP server to send text messages to a phone number in case of alarm trip

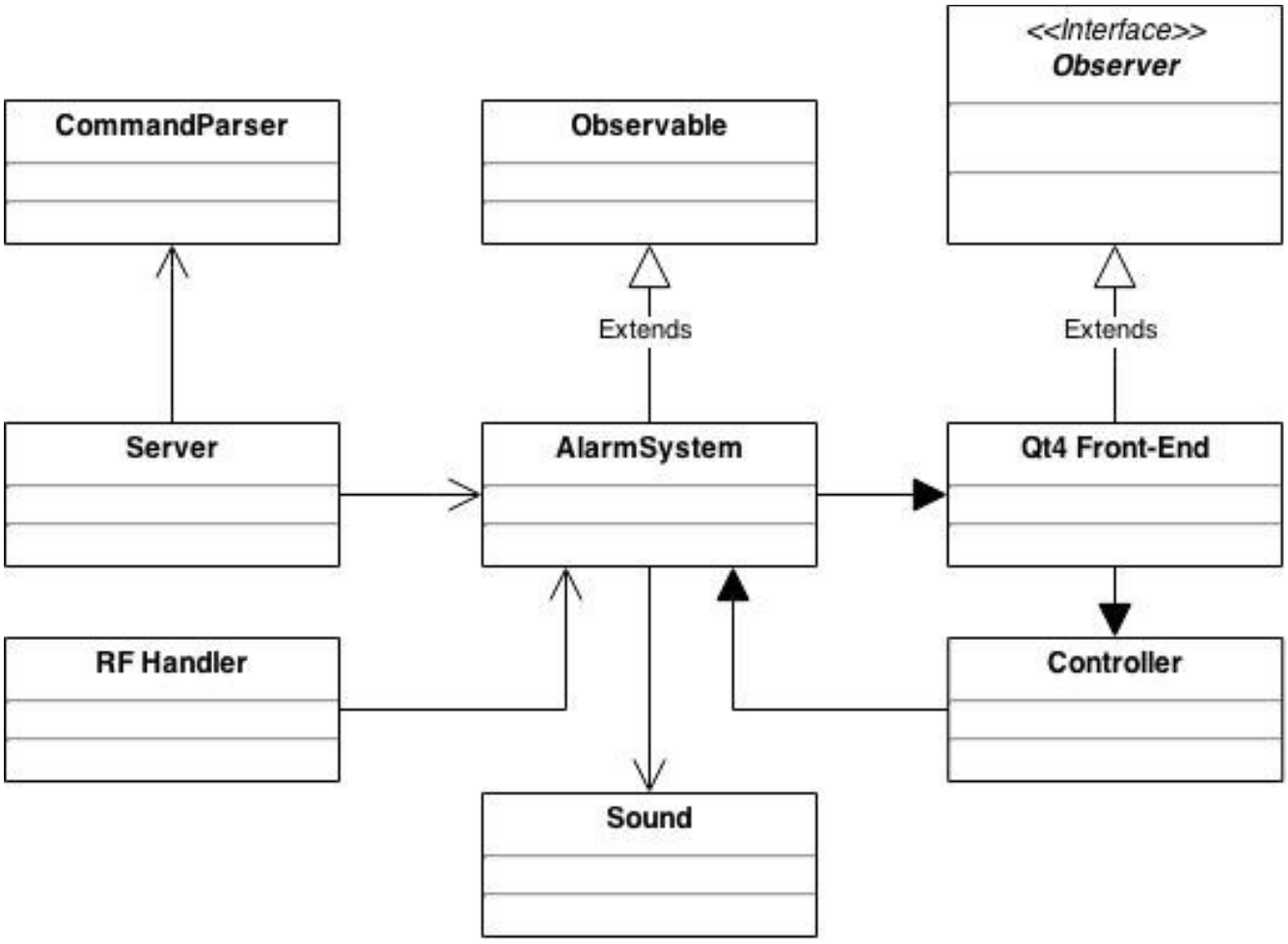
**Final Thoughts**

Constructing the Raspberry Pi Home Alarm System project was indeed fun. It allowed me to apply many of the skills I've been taught and have learned on my own throughout my school years as a computer science major, and to also prove myself as a software developer.

# Appendix A

# UML

---



# Appendix B
# Source Code

---

For the source code refer to:

- https://github.com/gaperezcortes/CSCE470/tree/RPi

# Bibliography

[1] Property Crimes in the US 2013, http://www.fbi.gov/about-us/cjis/ucr/crime-in-the-u.s/2013/crime-in-the-u.s.-2013/property-crime/property-crime-topic-page/propertycrimemain_final

[2] Arduino Mini information, http://arduino.cc/en/Main/ArduinoBoardMini

[3] Raspberry Pi information, http://downloads.element14.com/raspberryPi1.html

[4] Attiny85 information, http://www.atmel.com/devices/attiny85.aspx

[5] Raspberry Pi projects, https://hackaday.io/projects/tag/raspberry%20pi

[6] Qt Creator Info, http://qt-project.org/wiki/Category:Tools::QtCreator

[7] Valgrind, http://valgrind.org/

[8] Node.js, http://nodejs.org/about/

[9] 433Utils, https://github.com/ninjablocks/433Utils

[10] The Agile Samurai (October 5, 2010)

[11] How to write effective Test cases, procedures and definitions, http://www.softwaretestinghelp.com/how-to-write-effective-test-cases-test-cases-procedures-and-definitions/

[12] Agile Management, http://agilemethodology.org/

[13] Valgrind Quickstart, http://valgrind.org/docs/manual/quick-start.html#quick-start.mcrun