UNIVERSITY OF ALASKA ANCHORAGE

College of Science and Engineering

Home Security Alarm

by

Dari Donkkuro

Supervisor:

Prof. Dr. Kirk Scott, PhD

APRIL 13, 2015

A CAPSTONE PROJECT SUBMITTED TO THE DEPARTMENT OF ENGINEERING, FOR THE DEGREE OF COMPUTER SCIENCE.

Abstract

The target of this capstone is to present a new solution for home security system, thus the problem under study concentrates its main focus on the creation of a Do-It-Yourself security alarm system using raspberry pi technology and android powered device for remote control.

Raspberry pi is a general purpose micro-computer that was originally developed for educational purpose in 2006 at the University of Cambridge. It has all the components of a computer, including, SD Card, USB 2.0 port, RCA port, HDMI port, CPU,GPU, Ethernet port, just to mention a few. These features made Raspberry Pi a perfect fit for home automation. Since its emergence. Further discussion about raspberry pi and the project is covered in Chapter one in great detail.

Chapter two, system integration and modeling. In this chapter, my focus was on system integration and project implementation. How the GUI would be developed and the technologies that would be used was discussed. For testing purposes, android tab 4 would be used. Another technology that would be used is pubnub real time system. This used to ensure speedy project implementation and to improve efficiency. At this stage of the project, you would see figures of the GUI under development and estimated date of completion

Finally, in chapter three, you would see the final product tested. The figures shown in this chapter may change a little bit to improve look-and-feel for the final presentation.

ACKNOWLEDGEMENTS

The completion of this capstone project was the result of the help, cooperation, faith and support of two people, thus I would like to sincerely thank both of them.

Foremost, my special thanks for Prof. Kirk Scott as my supervisor during this project implementation. He provided me with the required support for the development of a better project. His supervision of my project has ensured the success of this work.

My special thanks also to my Teammate, Mr. P. C. Gabriel, for his cooperation, hard work and significant contributions on server and hardware programming.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Introduction

Home security alarm system, as the name suggests, is an alarm system using raspberry Pi and Arduino technologies. Although home alarm systems already exit, they are not affordable to home owners, not to mention home owners from developing worlds like African countries. The Raspberry Pi (RPi) is a credit card -size Linux computer, cheap general purpose computer that was produced at the Cambridge University laboratory in 2006 as an educational computer. The original aim was to educate or improve the programming skills of future computer Science undergraduate applicants [1].

Furthermore, RPi has all the components of a computer, including, SD Card, USB 2.0 port, RCA port, HDMI port, CPU,GPU, Ethernet port, just to mention a few. These features made Raspberry Pi a perfect fit for home automation. Since its emergence, RPi has been used in many different ways, ranging from home entertainment to weather forecasting, home automation including security, remote sensing, and so on.

Arduino on the other hand, is an open source hardware product with a programming language support that allows programmers to create custom applications using any of the supported languages on RPi. Using Arduino jointly with RPi any home electronic device can be controlled programmatically. This ability to control home appliance gives rise to the so called smart home [3].

Using RPi and Arduino, it is possible to fight crimes like burglary, especially in developing countries where home security is very poor. Very few people can afford to have a custom home system and pay the exorbitant cost involved. It is quite easy to create a home alarm system with a Raspberry Pi that is relatively cheap with promising features.

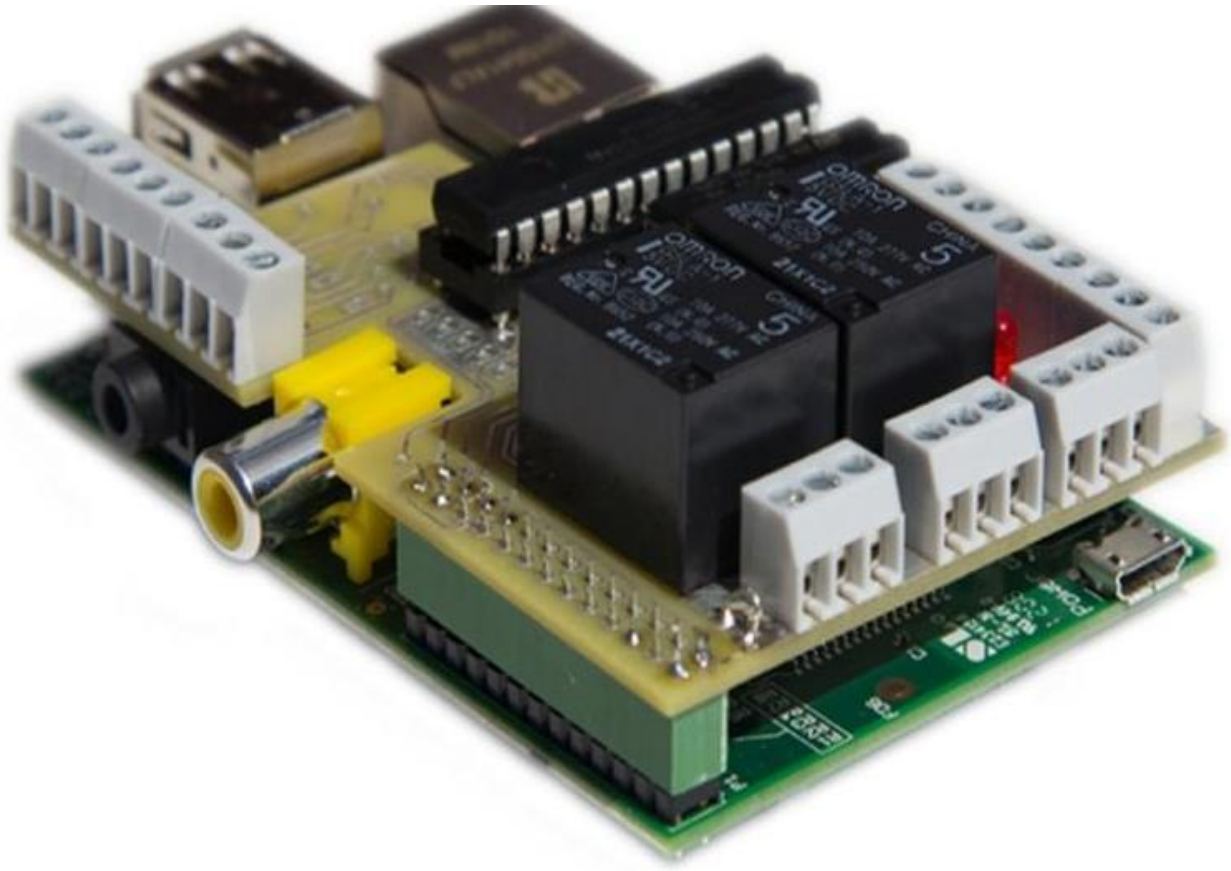


Figure 1.1: The raspberry pi board.

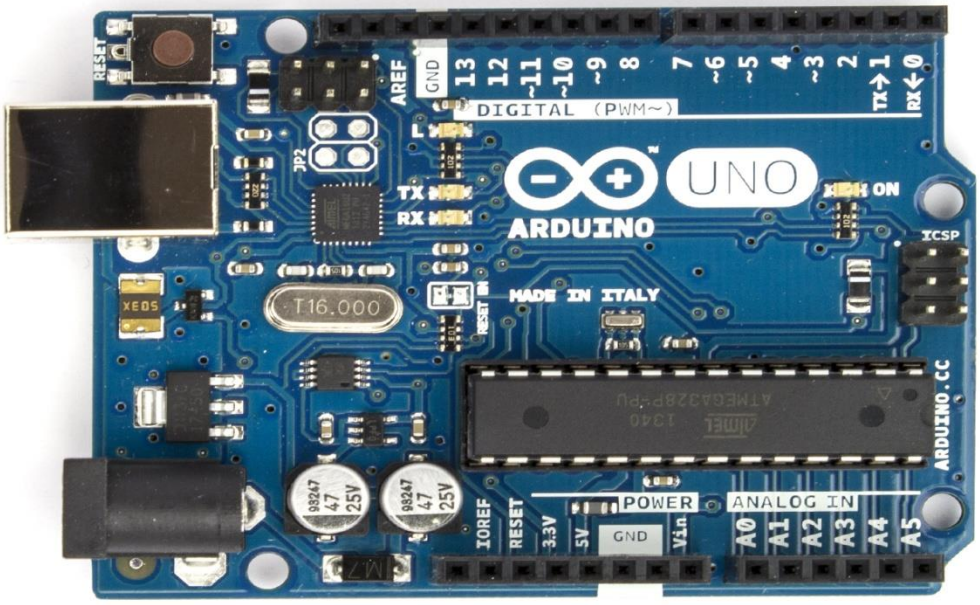


Figure 1.2: Arduino uno board.

1.2 Application

The implementation is based on the idea of home automation and remote control of one's house while away from home. This project is in joint collaboration with another student who will be responsible for the hardware set up and logic programming. Similar projects have been completed by other enthusiasts and have proved reliable in the past.

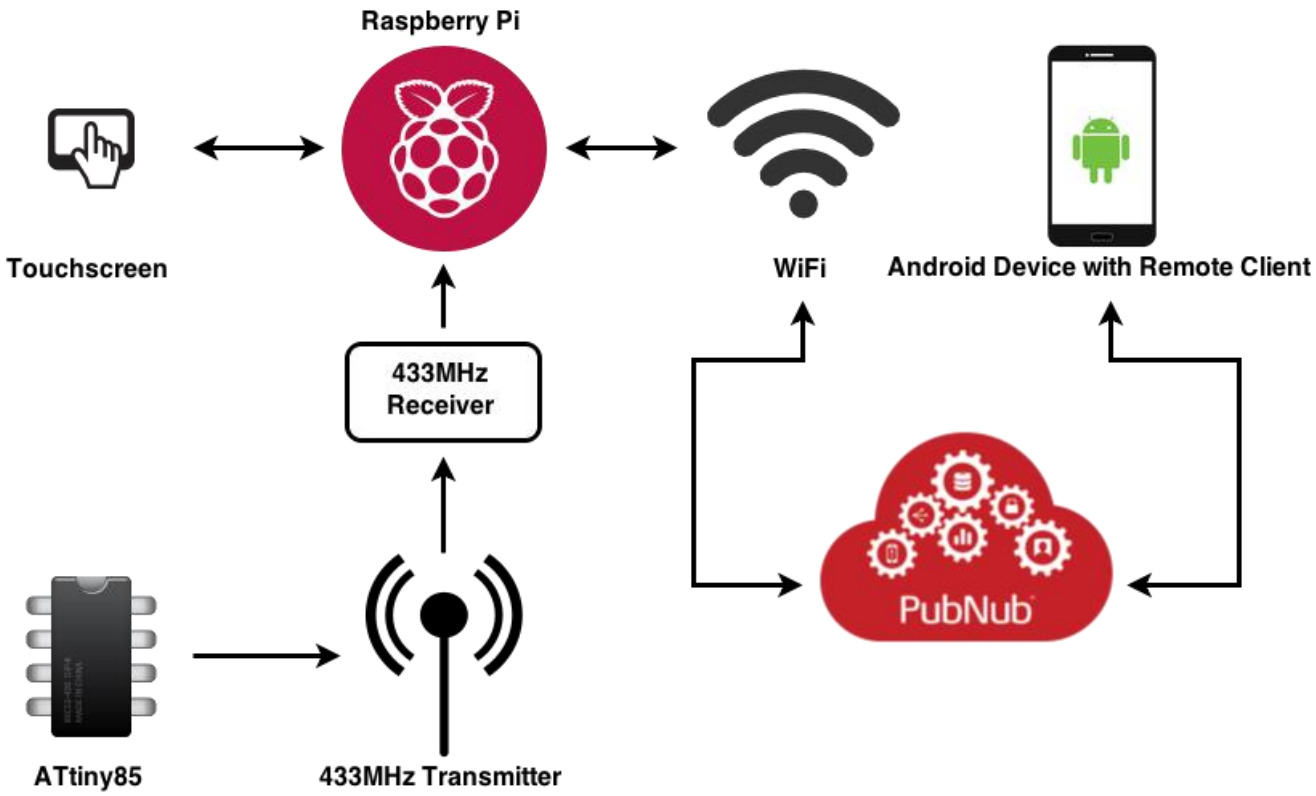


Figure 1.2: The project model and component dependency diagram.

The above figure (1.2) depicts the interactions between individual components of the project being implemented. This was our initial approach. However, through research, we came out with a better approach. Basically, instead of using a local server, pubnub, an open source real-Time network would be used. Individual responsibilities for this project would be discussed in the next section.

As mentioned previously, this project would be implemented as a group of two, where I would be responsible for developing an application using the android platform to communicate with RPi over pubnub network system.

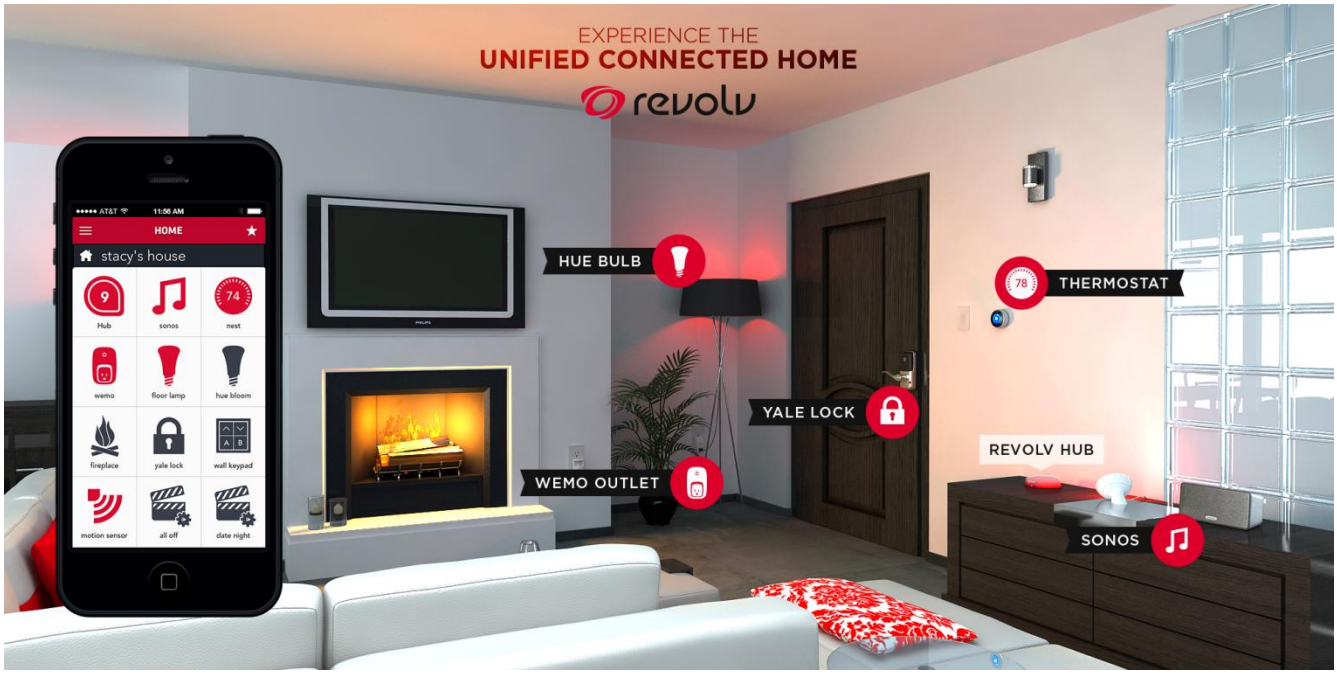


Figure 1.3: Connected Home example by Revolv.

To remotely control the alarm with a mobile device, the device will sustain a connection with pubnub network over which it would be able to receive and send messages over the channel. The messages to be passed would be simple commands such as: Lock to start receiving notifications when the state changes, Unlock to stop receiving notifications.

1.3 Motivation

The motivation for developing the alarm system is based on several factor such as growing interest in home automation and security. The home automation industry has been around since the 1970s but has not been used very much because of the high cost [3]. Usually, the price of a installing an alarm ranges between $250 - $750 or higher with more functionalities such as remote control or phone calls [4].
The fundamental aim of this alarm system is to improve home security while keeping the cost as low as possible [5]. Especially, home owners from poor countries like in Africa need better projection of their property by using cheap technological products like RPi and Arduino. Rather than a professional service provider, our approach would be a do-it-yourself (DIY) alarm using the above technologies.

Figure 1.3 shows a fully connected home that can be remotely controlled with a mobile device. For this project, we are only interested the state of an observed door, which can be either open or close as mentioned previously. When a door open event is triggered, the user would be notified or to make it even more realistic, an alarm would be triggered on the mobile device indicating an observed door open is being opened.

1.4 Recent Developments

The technology trends show that the home security market is developing rapidly in recent years at a rate of 9.1%[2]. According to the research report by marketsandmarkets.com, the world's largest users of home security systems are north America and Asia-Pacific with revenue growth of 55.6% and 28.4% from 2012-2017[2].

Home security systems can be categorized into two: Communication, and sensing technologies. A communication system can be wired or wireless. Each category is further segmented into alarm systems, intercom systems, video surveillance systems, access control and management systems, medical alert systems, energy management systems, and integrated security systems. In our project, we will only focus on the alarm system.

Chapter 2

System Integration and Modeling/Methodology

2.1 Introduction

In this chapter, I will discuss about how the android application would be implemented to communicate with raspberry pi, which was introduced in chapter one. My discussion would be focused on only the GUI development, the raspberry pi component of the system would not be discussed.

Obviously, the java programming language would be used for the GUI development, which is the language of the android operating system. I would be using android studio as the IDE for developing the application. Android studio, unlike eclipse, is specifically used for developing android applications. In addition, XML would be used for the layout and styling. Using XML for the user interface makes it easier and separates application logic from the view. It also keeps the codes short and manageable.

During the development phase, I will be using Samsung galaxy tab 4 for testing and debugging the application. I just bought it at the beginning of the semester specifically to be used for this project. I have already tested it with mock applications and it work great. I am planning to test the application on other device running the latest version of android including cell phones. The one I am currently using is running on android 4.4.2, which does not support upgrade quite yet. Debugging on cell phone would depend on availability because I am not planning on buying one at the moment.

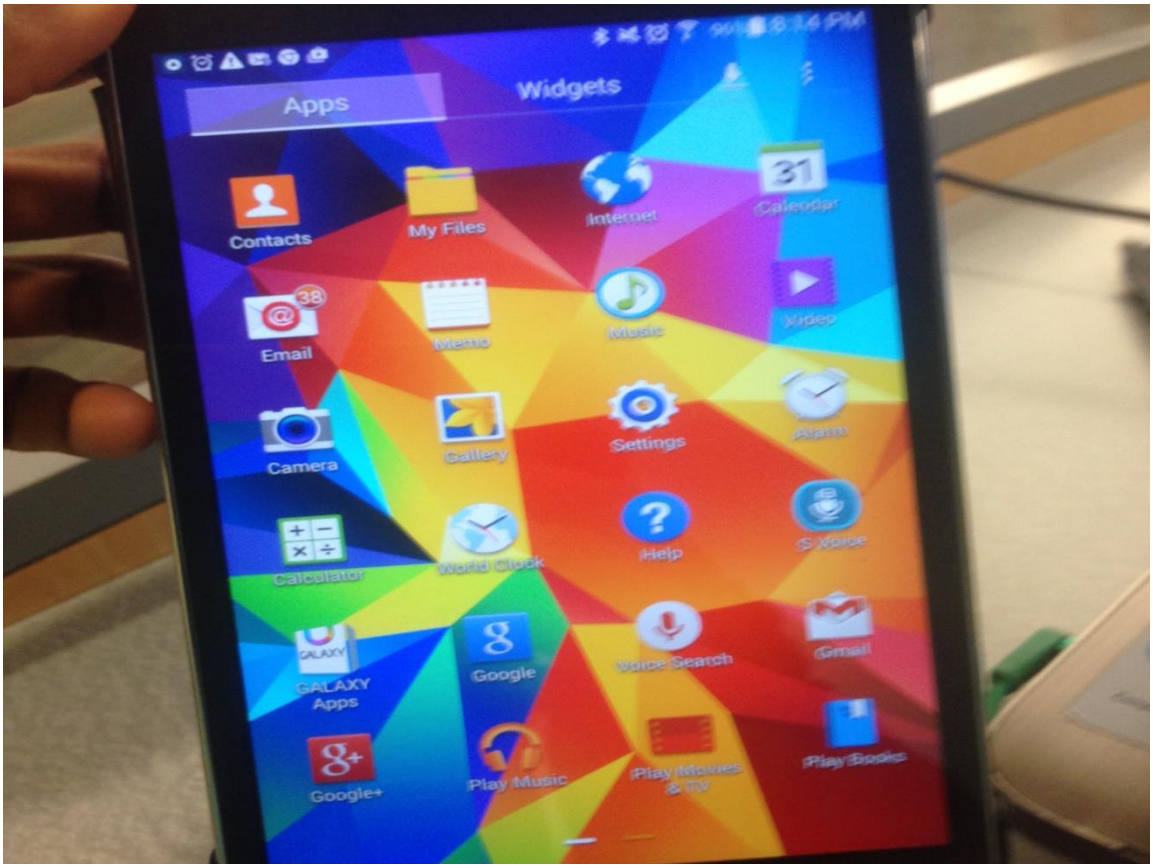


Figure 2.1 a: Front View of Samsung Galaxy Tab 4.

Figure 2.1 b: Front View of Samsung Galaxy Tab 4

2.2 Application

In this section, I will describe how the application would be implemented using the technologies described in the previous section. As mentioned in the previous section, I will be using java and xml for this application. There would be a minimum of three activities to keep the application simple to use.

```
public class Accessibility extends Activity {
    Button mButton;
    Intent mainPanelIntent;
   // TextView _view;
    ViewGroup root;
    private int xDelta;
    private int yDelta;
    EditText pin;
   @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
       setContentView(R.layout.accesspin);
     pin = (EditText)findViewById(R.id.pinText);
      mButton = (Button)findViewById(R.id.Sendbutton);
       mButton.setOnClickListener(
              (view) -> {
                    if(!(pin.getText().toString().matches("[0-9]{4}")))
                     Toast.makeText(getApplicationContext(), "Enter a valid pin to continue.", Toast.LENGTH_SHORT).show();
                   else{
                       mainPanelIntent = new Intent(getApplication(),AlarmPanel.class);
                        startActivity( mainPanelIntent);
                   }
              });
      }
```

Figure 2.2: This code snippet is an activity example.

The above figure (1.2) depicts how an activity is created. Notice the call to **setContentView()** method in the **onCreate()** method. The parameter of setContentView(), **R.id.accesspin**, in my case is a reference to a resource (res) file. The benefit of using a separate source file from code is to better separate presentation from application logic[6].

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#bbb">
    <RelativeLayout
        android:layout_width="350dp"
        android:layout_height="200dp"
        android:paddingLeft="26dp"
        android:paddingRight="26dp"
        android:background="#fff"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"
        android:paddingTop="40dp"
        android:id="@+id/innerlayout">
        <Button
            android:layout_width="96dp"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:text="Send"
            android:textSize="30dp"
            android:id="@+id/Sendbutton"
            android:layout_marginTop="73dp"
            android:background="#7FE817"
          />
        <EditText
            android:id="@+id/pinText"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
             android:ems="5"
            android:inputType="number"
            android:layout_marginTop="20dp">

            <requestFocus />

         <TextView
             android:id="@+id/textView5"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
              android:text="PIN CODE:"
             android:textSize="12dp"
             android:layout_marginTop="2dp"/>


    </RelativeLayout>

</LinearLayout>
```

Figure 2.3: Code snippet of a layout(Resource) file.

Using separate source files enables the developer to easily modify layout parameters without having to modify source code [7].
As you can see below, this snippets are relatively short and straightforward. Figure 2.3 shows the result of snippets. The same could be achieved programmatically [8], as shown in **figure 2.4** below.

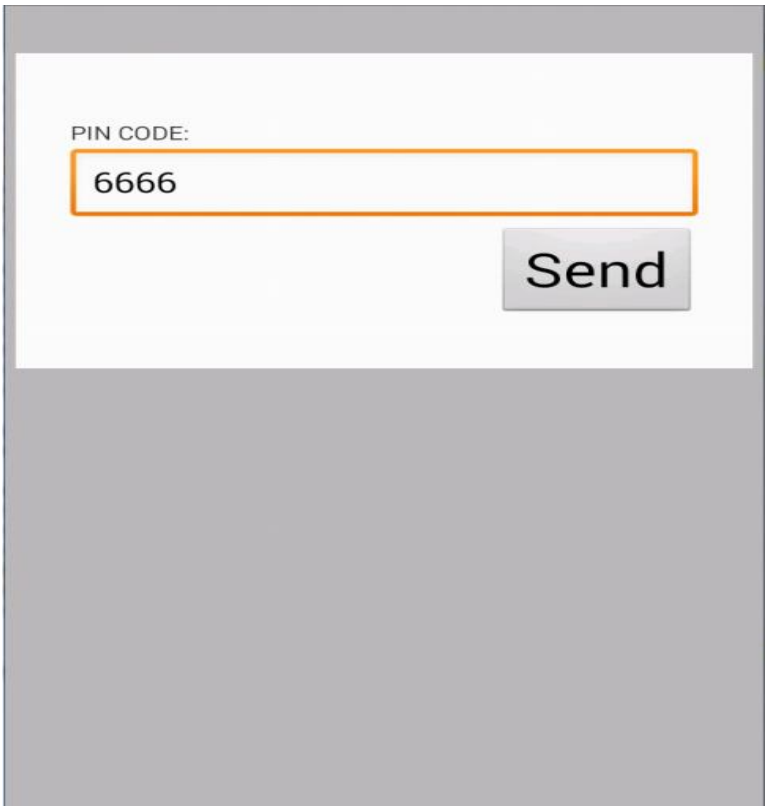


Figure 2.3: Result of running the activity above.

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // creating LinearLayout
        LinearLayout linLayout = new LinearLayout(this);
        // specifying vertical orientation
        linLayout.setOrientation(LinearLayout.VERTICAL);
        // creating LayoutParams
        LayoutParams linLayoutParam = new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT);
        // set LinearLayout as a root element of the screen
        setContentView(linLayout, linLayoutParam);

        LayoutParams lpView = new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);

        TextView tv = new TextView(this);
        tv.setText("TextView");
        tv.setLayoutParams(lpView);
        linLayout.addView(tv);

        Button btn = new Button(this);
        btn.setText("Button");
        linLayout.addView(btn, lpView);


        LinearLayout.LayoutParams leftMarginParams = new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        leftMarginParams.leftMargin = 50;

        Button btn1 = new Button(this);
        btn1.setText("Button1");
        linLayout.addView(btn1, leftMarginParams);


        LinearLayout.LayoutParams rightGravityParams = new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        rightGravityParams.gravity = Gravity.RIGHT;

        Button btn2 = new Button(this);
        btn2.setText("Button2");
        linLayout.addView(btn2, rightGravityParams);
    }
}
```

Figure 2.4: This snippet shows how to create a layout programmatically.

One advantage of creating a layout programmatically comes in handy when a dynamic GUI is desired [9]. The developer can manipulate layout parameters in code to control the GUI behavior. However, as

you can see, the snippet above generates a TextView and three buttons. One thing that stands out is the **layoutparams.** Every view must have its own, thereby causing redundancy in the code.

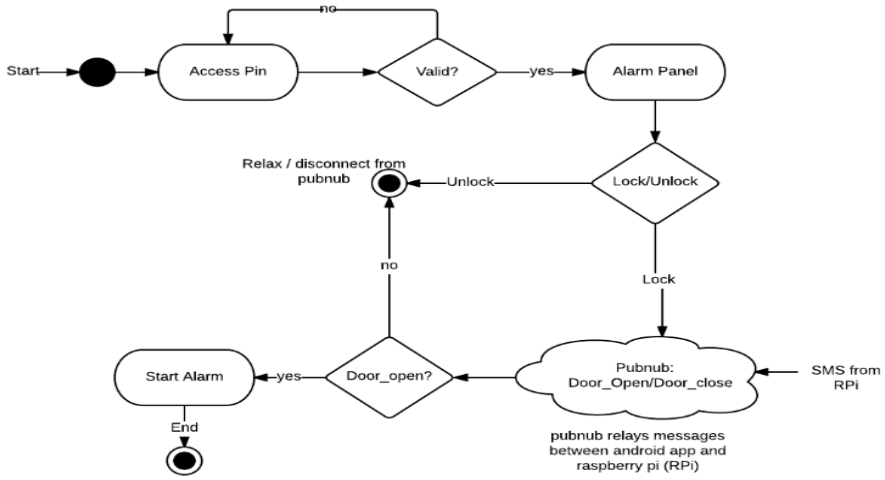


Figure 2.5: Flow Chart of android application.

Figure 2.5 is a sequence diagram showing the flow logic of the application. The heart of whole system is the pubnub component. It is a cloud messaging API that relays messages between two applications in real Time. For this project, pubnub mediates the communication between the android application and raspberry Pi.

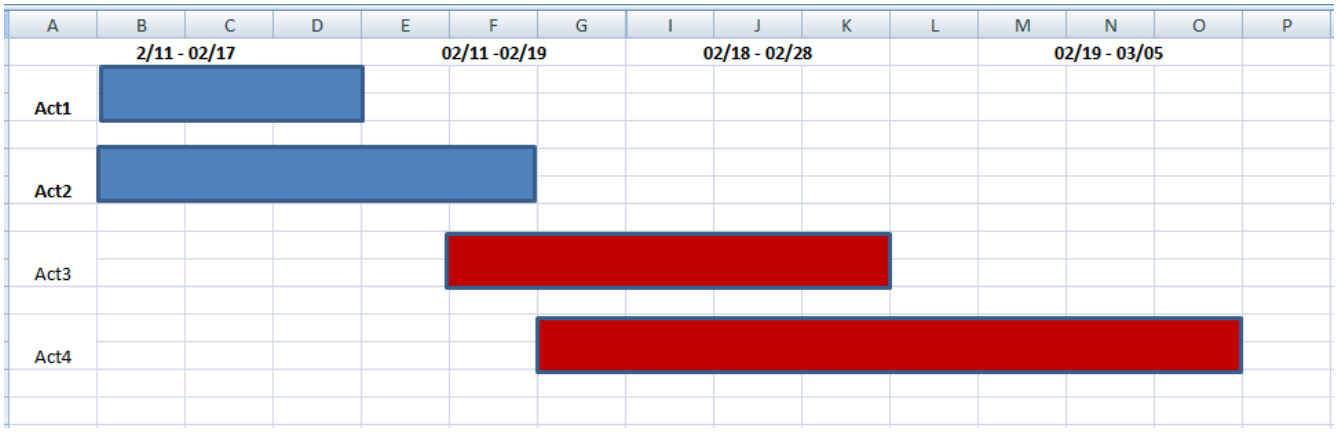


Figure 2.6: Application Gantt chart

In the Gantt chart above, activities three and four (Act3 & Act4) are the critical components of the system. Activity three handles network communications while activity four concerns itself with communications between the user and the application when the alarm goes off.

2.3 Motivation

In chapter 1, the motivation for developing the alarm system was given and would not be repeated in this section.

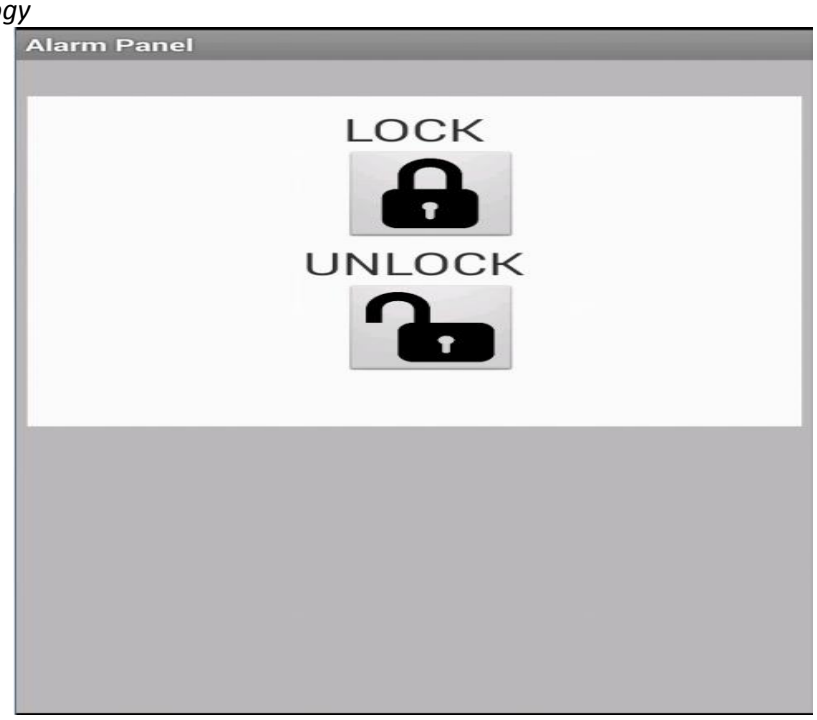


Figure 2.6: Alarm control panel

Figure 2.6 shows the graphical view of the alarm control panel under development.
It has two controls the user can use to lock and unlock the system remotely on his/her android device.

2.4 Agile Methodology

Agile methods or Agile processes is a software development methodology that generally promote a disciplined project management process. It encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, usually in two week durations or less. It is a business approach that aligns development with customer needs and company goals [10].

Chapter 3
Design and Testing

3.1 Introduction

Software testing is one of the most import phases in a system design. It is how the quality of a system is assessed. In this chapter, I will discuss about how the user interface for alarm system would be tested. Software testing usually take 40% ~ 50% of the entire system engineering [11]. According to Lu [11], Software testing is a very broad area, which involves many technical and non-technical areas, including specification, design and implementation, maintenance, process and management issues in software engineering. The goal of a Software test is to evaluate the performance of the system, trying to find errors that may exist and fix them.

As stated above, software testing is a broad field and can further be categorized into, unit testing, integration testing, acceptance testing, and system testing.

In this chapter, many testing techniques would be consider, since some techniques are inseparable in general. Although they are many testing techniques, they are only two classes of testing, including **black-box** and **white-box** testing [12]. The difference between the two classes is that, black-box testing focuses on the output generated by the system given an input, while ignoring the internal mechanism of the entire system. Meanwhile, a white testing takes into account the internal mechanism of the system.

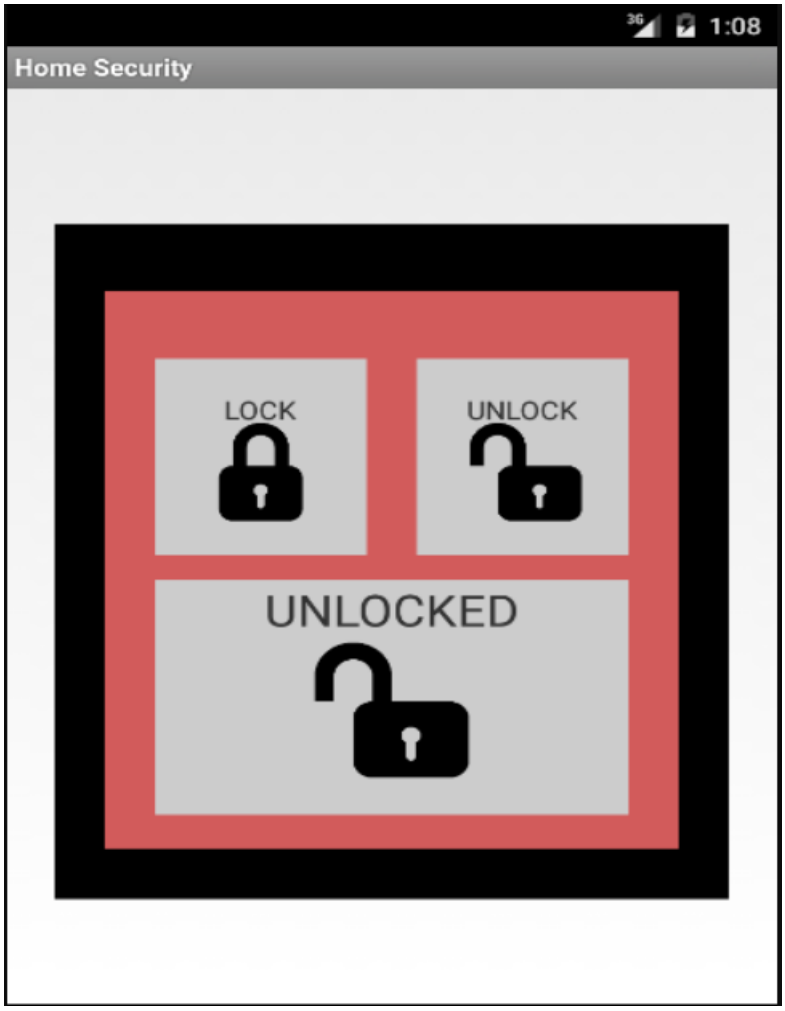


Figure 3.1: Initial state of the UI.

Figure 3.1 shows the control panel of the system. The user can interact with the system using the **Lock & Unlock** controls. By default, the system is in the unlock state, as shown in the lower panel. For security reasons, the user must provide pin code to be able to lock and unlock the system.

What is been tested for the control mechanics is based on result produced, whether the desired result was achieved. Figure 3.2, below shows what results when the user selects one of the two options (Lock or Unlock). In the lower panel, the user would see the current state of the system that reflects the user's action.

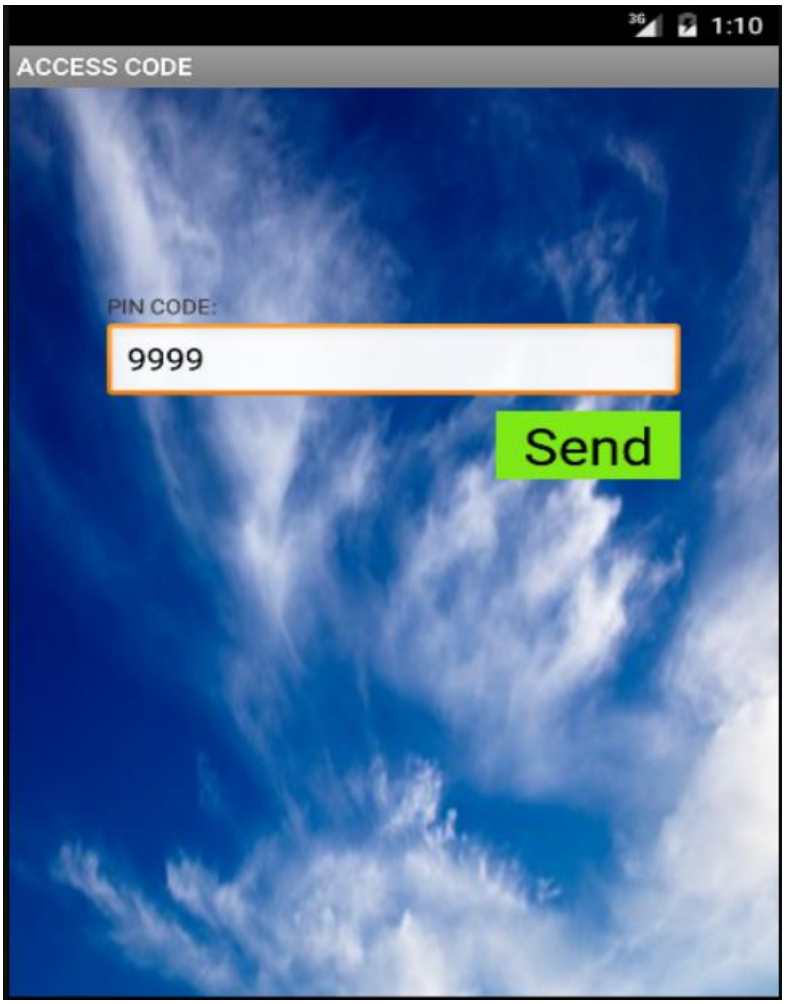


Figure 3.2: Lock and unlock require a code.

3.2 Testing Techniques

In this section, I will describe how the UI would be tested using the pubnub api technologies described in the previous chapters. As discussed in chapter two, the java language and xml were used for this UI development.
The access pin is used to determine the state of the system and would again be used later to decide the appropriate action upon receiving a message from raspberry pi. According to my standards, the lock and unlock functionality testing passes base on the following:

I. The user can successfully transition between activities by choosing an option.
II. The pin code entered is successfully save to file that can later be retrieved.
III. The state of the system is updated to reflect the user optio

IV. n.

Figure 3.3: This image shows the alarm going off.

In the figure above, when a message is received from raspberry pi, the alarm activity is fired. The user would be required to provide a valid code to stop the alarm. Figure 3.4 shows the flow diagram of the activities described.

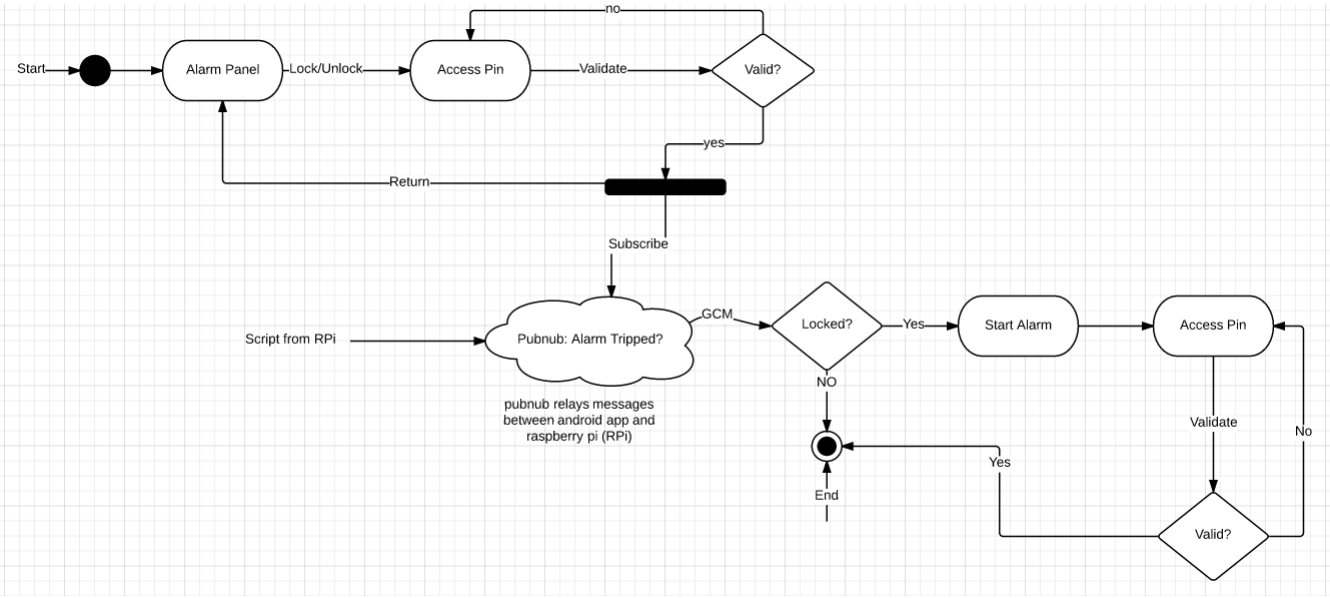


Figure 3.4: The GUI flow diagram.

So far, the testing strategy described till is mainly based on the output produced based on user action. If the desired result is achieved, then the unit/component been tested passes. This is a black-box testing strategy since code structure is not taken into account.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
    prefs = getSharedPreferences(
            "PHOME SECURITY SYSTEM", Context.MODE_PRIVATE);
    ComfirmationPrefs = getSharedPreferences("PIN_CODE", Context.MODE_PRIVATE);
      init();
     setContentView(R.layout.panellayout);
     txtview = (TextView) findViewById(R.id.stateLabel);

    subcribeButton = (ImageButton)findViewById(R.id.imageLockButton);
    UnsubscribeButton = (ImageButton)findViewById(R.id.imageUnlockButton);
    StateImgButton = (ImageButton)findViewById(R.id.StateButton);

    if(REG_ID !=null)
        gcmRegister();
        subscribe();
    this.registerReceiver((arg0, intent) -> {
            pubnub.disconnectAndResubscribe();

    }, new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION));

    pin = (EditText)findViewById(R.id.pinText);
    mButton = (Button)findViewById(R.id.Sendbutton);

     subcribeButton.setOnClickListener(
            (view) -> {
                 String lockState =  ComfirmationPrefs.getString("PIN_CODE","");
                   if(lockState !=null){
                       Toast.makeText(getApplicationContext(), "The system is locked",
                       Toast.LENGTH_LONG).show();
                       startFragmentActivity();
                   }

                   if(PIN_CODE !=null){
                   subscribe();
                   StateImgButton.setImageResource(R.drawable.lock);
                    txtview.setText("LOCKED");
                   }
            });

    UnsubscribeButton.setOnClickListener(
            (view) -> {
                    String lockState =  ComfirmationPrefs.getString("PIN_CODE","");
                     if(lockState == null){
                         Toast.makeText(getApplicationContext(), "The system is unlock",
                         Toast.LENGTH_LONG).show();
                        }
                    else{
                    startFragmentActivity();
                      if(PIN_CODE == lockState){
                         unsubscribe();
                          StateImgButton.setImageResource(R.drawable.unlock);
                          txtview.setText("UNLOCKED");
                          final SharedPreferences.Editor editor = prefs.edit();
                          editor.putString("PIN_CODE",null);
                          editor.commit();
                      }
                     }
            });
}
```

Figure 3.5: This is the structure of the main activity oncreate() method.

For the benefit of people who have not programmed in android, the code snippet above is onCreate() method, which is analogous to the main method in java/C#/C/C++. In this section, I would give a discussion about another software testing strategy used during this testing. As

stated before, they are two classes of software testing techniques, namely, black-box and white-box testing.

Unlike black-box testing, white-box software testing is concerned with **control-flow/coverage** testing[13]. Simply, a white-box testing considers the following coverage testing strategies:

1. Method coverage: A measure of the percentage of methods that have been executed by a test case. In the snippet above, 100% method coverage is possible when at runtime. Consider a call to **init()**, **gcmRegister(), subscribe(), setContentView()**, etc, the expected results are pubnub object is created, application is registered with GCM, subscribe to a channel, and sets the content view respectively.
2. Satement coverage. Similarly, a statement coverage measures the percentage of statements executed by a test case. If as system would be qualitative and reliable, then there should be a 100% statement coverage since it would be inefficient otherwise. In the abouve test case, we cannot obtain 100% statement coverage because of the conditional and user actions. For example when **REG_ID,** which is a one time registration ID is non-null**,** and when the user **subscribeButton** and **lockstate and PIN_CODE** are non-null, and finally, when **unsubscribeButton,** either the if-clause is executed or the else-clause is executed. Because of this, it is not possible to obtain a 100% statement coverage in the onCreate() method.
3. Branch coverage. Branch coverage is measure of the percentage of boolean expressions evaluated to either true-or-false. Since the user either locks or unlocks the system at a time, branch coverage is not possible at one time.
4. Condition coverage. Finally, condition coverage is the same as branch coverage execpt that it is concerned with sub-boolean expressions of compound expression that are evaluated to either true or false by a program.

Both of these software testing techniques are effective and widely used. Using a black-box testing, the tester needs not worry about the mechanisms about the code but focus mainly on the result produced given some input. On the other hand, a white-box testing requires the tester understands how the code is structured and how it works.

3.3 Agile Methodology

Agile methods or agile processes is a software development methodology that generally promote a disciplined project management process. It encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, usually in two week durations or less. It is a business approach that aligns development with customer needs and company goals.

Agile projects have stages, including seven events for product development [14]. The first step in agile project planning is collecting user stories – short notes about project requirements. The user stories are broken into tasks, estimate the time duration each task can be completed, client prioritization of tasks, product delivery after each iteration and so on.

Chapter 4

User Manual

4.1 Introduction

This is a home security system, which can monitor motions around a house and sends alerts to the home owner's mobile device. The user would then be able to unlock the system or lock through the application that this document is based on. The proposal for this project was made and covered in greate detail in chapter. Chapters two and three covered the development and testing process respectively. This document will be presented in a form of a user manual in th following sections. Basically, this document will explain the features implemented and how each operates.

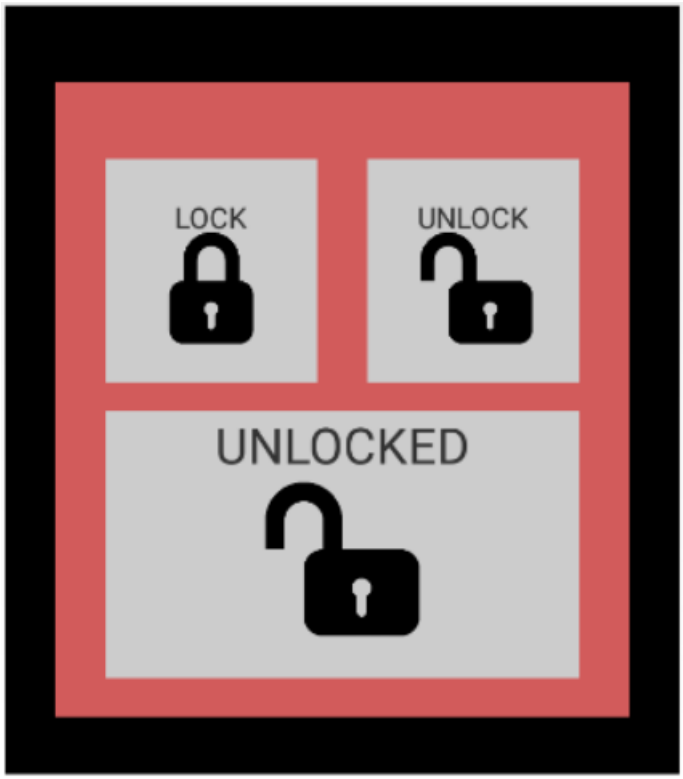


Figure 1: This is the final control panel

4.2.1 Home Screen

Finding the application in the home screen is easy. The icon itself resemble the application control panel in a bright yellow color as shown below.

Figure 2: Application icon

The home screen icon is shown in figure 1, which was designed using Microsoft word and paint programs. I chose the yellow color for this icon because it is easy to locate in the home screen. It is composed of action controls that user will generally interacts with when the application is launched.

It is strongly recommended that, the user uses proper backgrounds, that is, yellow backgrounds or backgrounds that are mostly yellow should be avoided. This is necessary because the icon is mostly yellow and it will be had to location in a similar background. It was tested on multiple backgrounds to determine the best one. It turned out that black or mostly black background would be the best option.

4.2.2 Control Panel

In **figure 1,** the system control panel was shown with two intractable lock and unlock buttons and a state. The use can only interacts with the lock and unlock functions. The state control tells the user the status of the system, whether it is locked or unlocked. One option can be taken at a time, that is, the user can lock the system if it was not already or unlock if it was already locked.

Depending on what option the user takes, a popup, as shown in figure 3, will be displayed, where the user can enter a pin code to lock the system.

4.2.3 Lock Process

Figure 3, shown below is a popup window that allows the user to enter a pin code to lock the system. This pin code will be saved and retrieved later for validation when the user wants to stop the alarm when it was tripped or when the user wants to unlock the system.

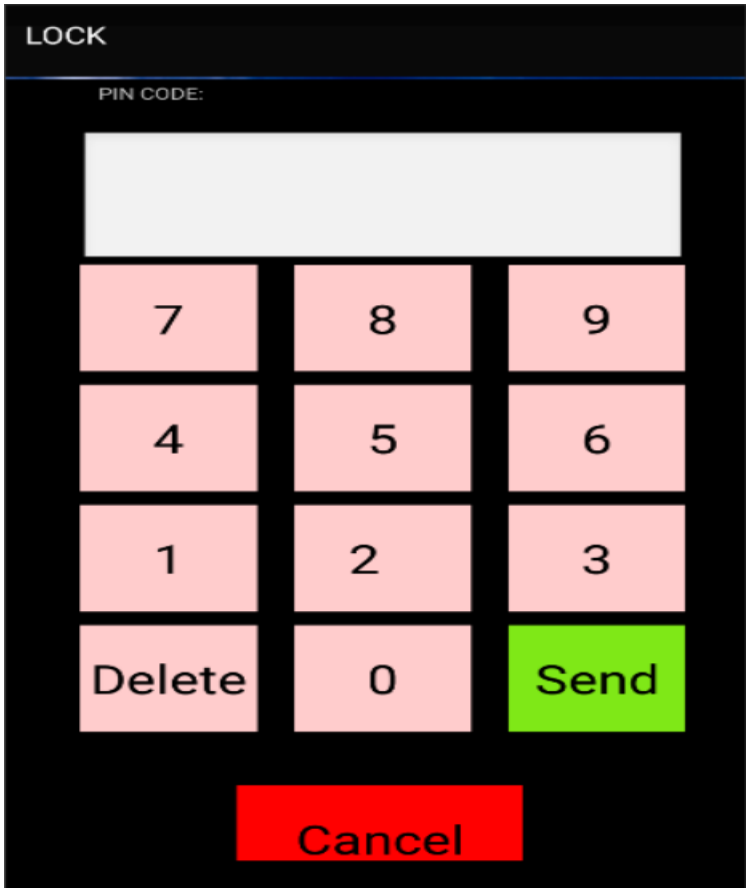


Figure 4: This is a popup of the lock panel

In this popup, the user can take two options: 1) can the action, 2) enter a pin code to lock. If the lock button, in figure 1, was hit on accident or the user decides not to lock the system after taking that

option, then the action can be cancelled by clicking the “Cancel” button. Otherwise, the user needs to provide a valid pin code to continue the locking process. It is recommended that the user uses a memorable 4-digit code for this because he/she will need it later. Hint: the last four digits of your cell phone number or a family member or a friend is good choice. You can also use the same screen unlock pin code that you have probably memorized.

It is strongly recommended the pin code should be kept secret and changed regularly. This is for security reasons. You do not want anybody stealing your device and gaining control over the system.

4.2.4 Unlock Process

Just like the locking procedure described in section 1.2.3, the unlocking process works in a similar manner. The dialog as shown in figure 5, can be brought to view by clicking the unlock button shown in figure 1. It works and looks very similar to figure 4, except it does the opposite task or undo what figure 4 did in the past

.

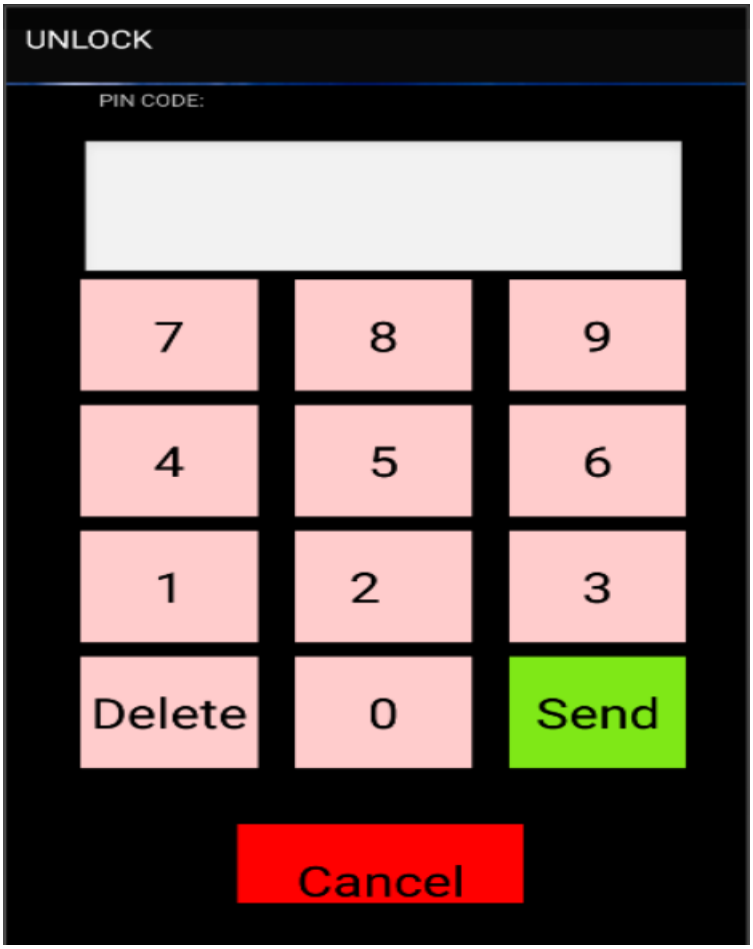


Figure 5: This is the unlock panel

As it is shown, you can either cancel or provide a valid pin code to unlock. The pin code is the same as what was used in the locking process. Bear in mind the pin code can be only four digits long in both the locking and unlocking processes. In both lock and unlock step, the popup panel will disappear when the screen orientation changes. In such a case, the panel will be presented where you can repeat the process.

4.2.5 Receiving Alerts

Once the system is locked, the user can then receive notifications through an android device.

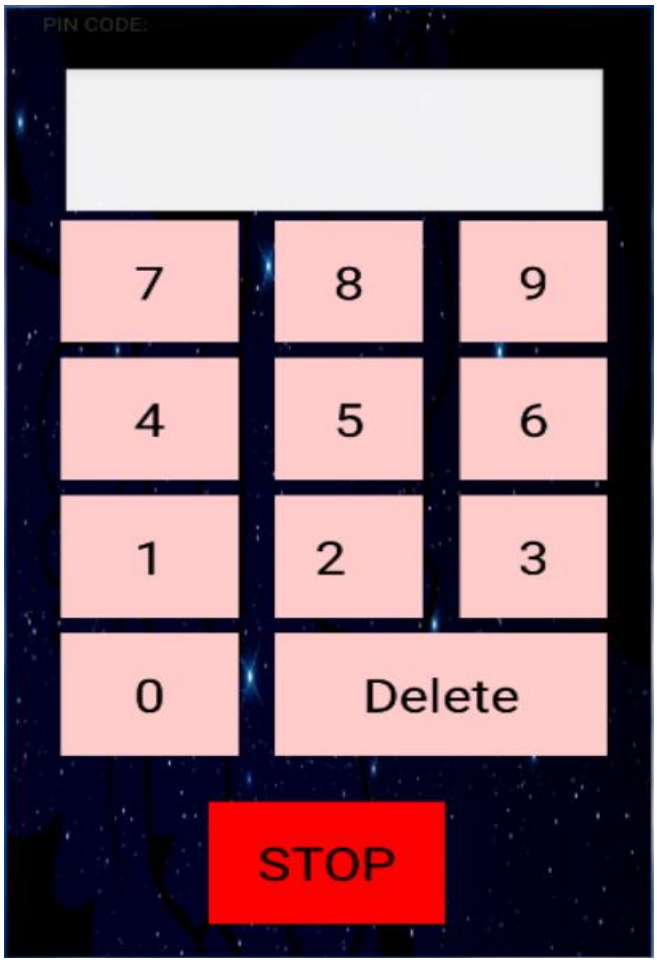


Figure 6: This is the alarm stop panel

In figure 6, the panel for stopping an alarm is shown. The user will be required to provide the same code that was used to lock the system to able to stop. For the purpose of this project, there is no limit on the number of trials. In practice, it will be a limit. This panel is forced to always be in portrait mode so that the activity is not killed the device is rotated.

Chapter 5

5.1 Project Summary

This is a DIY home security system which can detect door open & close activities in a house. The goal of the project is to utilize cost effective appliances, like raspberry pi motion sensor, and an android mobile device to build an integrated home security system. This security system will send an alert to the home owner when there is an attempt of break-in in his/her absence. The alert will be a sound played at loud volume by the user's android device. The user can then make decisions whether to call the cops or simply stop the alarm.

5.2 Motivation

Security systems are smarter means keeping homes safe in modern societies. However, as the market of security systems expands, the cost for security systems is also sky-rocketing. As a result, many security system customers are left with heavy monthly bills to pay. For example, as of 06/04/2015, ADT charges $29/month for a burglary monitoring service, equivalent to $348/year as compared with $35 lifetime equivalence. This price does not account for installation and other services. According to usnews.com, the f start-up cost for all equipment of a security system run between $600-$1,200, excluding installation cost. Unfortunately, home owners cannot afford this amount. The goal of this project is to provide an alternative way of acquiring a security system by building it yourself using raspberry pi and an android

5.3 Advantages

In the motivation section, I mentioned the cost of owning a home security system. I also mentioned how much it costs to build your own system using already existing equipment. It was obvious that building one yourself will safe you hundreds of dollars. Hence, not only does system provides protection to the user, it also saves him/her a lot of money.

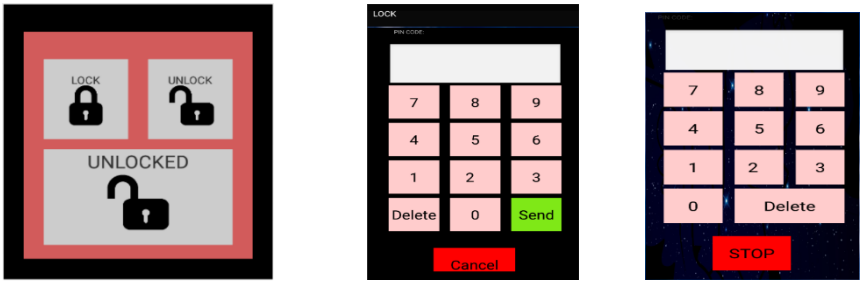


Figure 1 a: Control panel b: Lock panel c: Stop panel

Secondly, the user has more control over the system from anywhere. Just with a few button clicks, the user can lock and unlock the system from an mobile device. In figure 1a, there are lock and unlock functionalities. Each does exactly what you will expect; to lock and unlock the system. The user will need to provide a pin to lock, unlock and stop the system.

Thirdly, this project provides a much simpler user interface. Figure 1a-1c are the GUIs the user will mostly interacts with. Unlike the interface of a traditional service provider, this interface is easy to use and easy to install. The simplicity of the system as a whole played a big role in its development process.

Lastly, it provides protection. The goal of the project was to provide protection to the user, allowing him/her to control it from anywhere from a mobile device. The user will will be notified by playing a sound on his mobile device when there is any break-in atempt.

5.4 Recommendations for future implementation

My initial plan was to integrate a camera module into the system. There is nothing more refreshing to be able to view you house from anywhere on your device. Integrating a camera into the system can also allow a pet lover to watch their pets while at work. Even better yet, one can include a microphone functionality to allow the user to say things like: “Who’s here?” or commands a pet to stop an action. This can be achieve by implementing both surveilance and microphone functionality, so the user can view and talk. Hint: Use pubnub real-time android API or equivalence, GCM API to develop an application that implements the desired functionalities.

5.5 Conclusion

Imagine you could could receive an alert on your mobile device when roommate crosses some boundary or imagine you could fire up an application on your device right now and be able to see what is happening in your house, room, or yard and be able to yell at someone or something. Suppose your home already has security system, wouldn't it be nice to let a friend in by unlocking the system from your device without revealing your code? All of these are great features one can implement in future projects.

Appendix A

```
public class MainActivity extends Activity {

    Pubnub pubnub = new Pubnub("pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02","sub-c-930dd836-acd1-11e4-815e-0619f8945a4f");
    GoogleCloudMessaging gcm;
    SharedPreferences prefs;
    Context context;
    public static String SENDER_ID;
    public static String REG_ID;
    private static final String APP_VERSION = "3.6.1";
    Button pinButton;
    String PUBLISH_KEY = "pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02";//pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02";
    String SUBSCRIBE_KEY = "sub-c-930dd836-acd1-11e4-815e-0619f8945a4f";//sub-c-930dd836-acd1-11e4-815e-0619f8945a4f";
    String CIPHER_KEY = "";
    String SECRET_KEY = "";
    String ORIGIN = "Home Owner";
    String AUTH_KEY;
    String UUID;
    Boolean SSL = false;


    Button mButton;
    EditText pin;
    Intent mainPanelIntent;
//    pubnub = new Pubnub(
//            PUBLISH_KEY,
//            SUBSCRIBE_KEY);
//    pubnub.setCacheBusting(false);
//    pubnub.setOrigin(ORIGIN);
//    pubnub.setAuthKey(AUTH_KEY);
    static final String TAG = "Register Activity";

    private void notifyUser(Object message) {
        try {
            if (message instanceof JSONObject) {
                final JSONObject obj = (JSONObject) message;
                this.runOnUiThread(() -> {
                        Toast.makeText(getApplicationContext(), obj.toString(),
                                Toast.LENGTH_LONG).show();
                        Log.i("Received msg : ", String.valueOf(obj));
                    });

                } else if (message instanceof String) {
                    final String obj = (String) message;
                    this.runOnUiThread(() -> {
                            Toast.makeText(getApplicationContext(), obj,
                                    Toast.LENGTH_LONG).show();
                            Log.i("Received msg : ", obj.toString());
                    });

                } else if (message instanceof JSONArray) {
                    final JSONArray obj = (JSONArray) message;
                    this.runOnUiThread(() -> {
                            Toast.makeText(getApplicationContext(), obj.toString(),
                                    Toast.LENGTH_LONG).show();
                            Log.i("Received msg : ", obj.toString());
                    });
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        protected void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);
            prefs = getSharedPreferences(
                    "HOME ALARM", Context.MODE_PRIVATE);
            init();
            setContentView(R.layout.accesspin);
            pin = (EditText)findViewById(R.id.pinText);
            this.registerReceiver((arg0, intent) -> {
                    pubnub.disconnectAndResubscribe();
```

```
    private void saveCredentials() {
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("PUBLISH_KEY", PUBLISH_KEY);
        editor.putString("SUBSCRIBE_KEY", SUBSCRIBE_KEY);
        editor.putString("SECRET_KEY", SECRET_KEY);
        editor.putString("AUTH_KEY", AUTH_KEY);
        editor.putString("CIPHER_KEY", CIPHER_KEY);
        editor.putString("ORIGIN", ORIGIN);
        editor.putString("UUID", UUID);
        editor.putString("SSL", SSL.toString());
        editor.putString("SENDER_ID", SENDER_ID);
        editor.commit();
    }

    private Map<String, String> getCredentials() {
        Map<String, String> map = new LinkedHashMap<~>();
        map.put("PUBLISH_KEY", prefs.getString("PUBLISH_KEY", PUBLISH_KEY));
        map.put("SUBSCRIBE_KEY", prefs.getString("SUBSCRIBE_KEY", "SUBSCRIBE_KEY"));
        map.put("SECRET_KEY", prefs.getString("SECRET_KEY", "demo"));
        map.put("CIPHER_KEY", prefs.getString("CIPHER_KEY", ""));
        map.put("AUTH_KEY", prefs.getString("AUTH_KEY", null));
        map.put("ORIGIN", prefs.getString("ORIGIN", "Home Owner"));
        map.put("UUID", prefs.getString("UUID", null));
        map.put("SSL", prefs.getString("SSL", "false"));
        map.put("SENDER_ID", prefs.getString("SENDER_ID", null));
        return map;
    }
```

Appendix A: Appendix B for chapter thr

```
public class MainActivity extends Activity {

    Pubnub pubnub = new Pubnub("pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02","sub-c-930dd836-acd1-11e4-815e-0619f8945a4f");
    GoogleCloudMessaging gcm;
    SharedPreferences prefs;
    Context context;
    public static String SENDER_ID;
    public static String REG_ID;
    private static final String APP_VERSION = "3.6.1";
    Button pinButton;
    String PUBLISH_KEY = "pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02";//pub-c-ef34a7ed-52a5-4e3b-8d12-39a96d416b02";
    String SUBSCRIBE_KEY = "sub-c-930dd836-acd1-11e4-815e-0619f8945a4f";//sub-c-930dd836-acd1-11e4-815e-0619f8945a4f";
    String CIPHER_KEY = "";
    String SECRET_KEY = "";
    String ORIGIN = "Home Owner";
    String AUTH_KEY;
    String UUID;
    Boolean SSL = false;


    Button mButton;
    EditText pin;
    Intent mainPanelIntent;
//    pubnub = new Pubnub(
//            PUBLISH_KEY,
//            SUBSCRIBE_KEY);
//    pubnub.setCacheBusting(false);
//    pubnub.setOrigin(ORIGIN);
//    pubnub.setAuthKey(AUTH_KEY);
    static final String TAG = "Register Activity";

    private void notifyUser(Object message) {
        try {
            if (message instanceof JSONObject) {
                final JSONObject obj = (JSONObject) message;
                this.runOnUiThread(() -> {
                        Toast.makeText(getApplicationContext(), obj.toString(),
                                Toast.LENGTH_LONG).show();
                        Log.i("Received msg : ", String.valueOf(obj));
                    });

            } else if (message instanceof String) {
                final String obj = (String) message;
                this.runOnUiThread(() -> {
                        Toast.makeText(getApplicationContext(), obj,
                                Toast.LENGTH_LONG).show();
                        Log.i("Received msg : ", obj.toString());
                    });

            } else if (message instanceof JSONArray) {
                final JSONArray obj = (JSONArray) message;
                this.runOnUiThread(() -> {
                        Toast.makeText(getApplicationContext(), obj.toString(),
                                Toast.LENGTH_LONG).show();
                        Log.i("Received msg : ", obj.toString());
                    });
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        prefs = getSharedPreferences(
                "HOME ALARM", Context.MODE_PRIVATE);
        init();
        setContentView(R.layout.accesspin);
        pin = (EditText)findViewById(R.id.pinText);
        this.registerReceiver((arg0, intent) -> {
                pubnub.disconnectAndResubscribe();
```

```
    private void saveCredentials() {
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("PUBLISH_KEY", PUBLISH_KEY);
        editor.putString("SUBSCRIBE_KEY", SUBSCRIBE_KEY);
        editor.putString("SECRET_KEY", SECRET_KEY);
        editor.putString("AUTH_KEY", AUTH_KEY);
        editor.putString("CIPHER_KEY", CIPHER_KEY);
        editor.putString("ORIGIN", ORIGIN);
        editor.putString("UUID", UUID);
        editor.putString("SSL", SSL.toString());
        editor.putString("SENDER_ID", SENDER_ID);
        editor.commit();
    }

    private Map<String, String> getCredentials() {
        Map<String, String> map = new LinkedHashMap<~>();
        map.put("PUBLISH_KEY", prefs.getString("PUBLISH_KEY", PUBLISH_KEY));
        map.put("SUBSCRIBE_KEY", prefs.getString("SUBSCRIBE_KEY", "SUBSCRIBE_KEY"));
        map.put("SECRET_KEY", prefs.getString("SECRET_KEY", "demo"));
        map.put("CIPHER_KEY", prefs.getString("CIPHER_KEY", ""));
        map.put("AUTH_KEY", prefs.getString("AUTH_KEY", null));
        map.put("ORIGIN", prefs.getString("ORIGIN", "Home Owner"));
        map.put("UUID", prefs.getString("UUID", null));
        map.put("SSL", prefs.getString("SSL", "false"));
        map.put("SENDER_ID", prefs.getString("SENDER_ID", null));
        return map;
    }
```

Appendix B: Appendix B for chapter two.

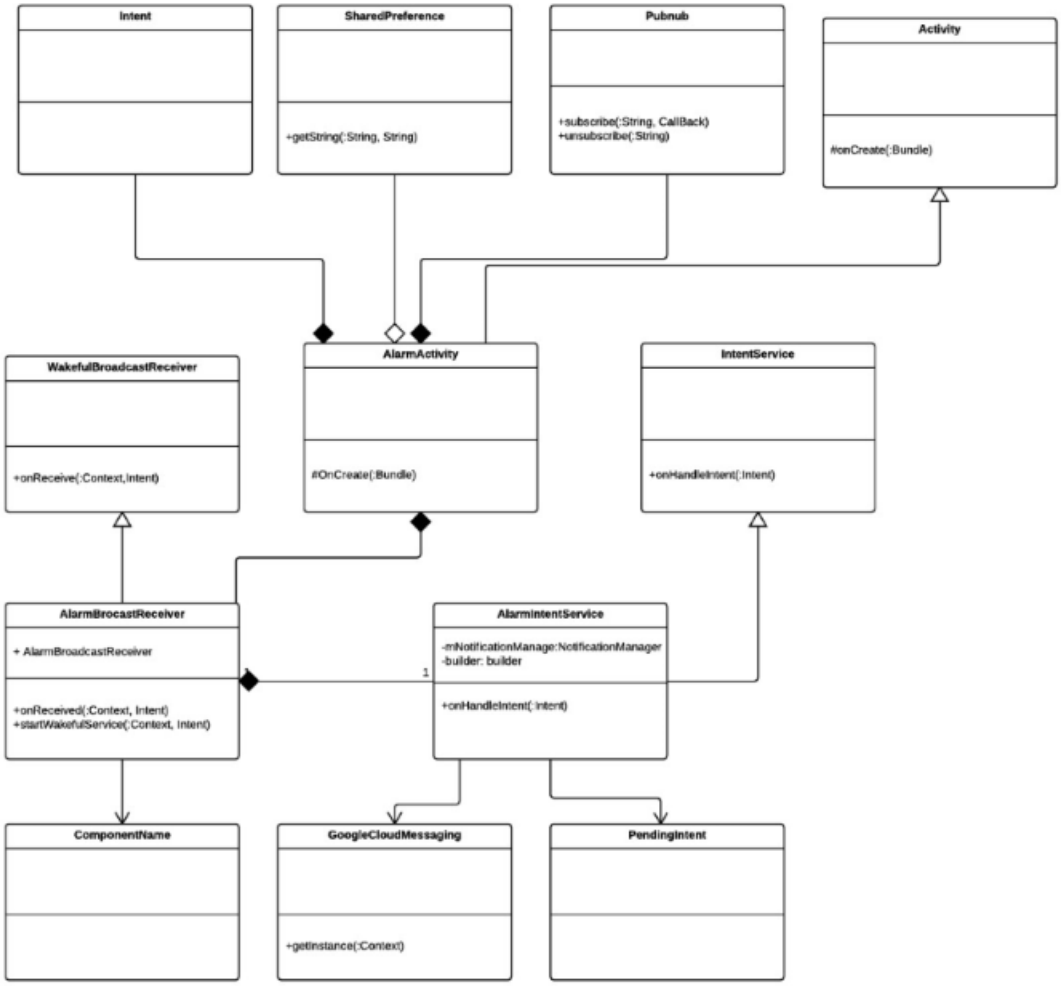
```java
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;

public class HomeAlarmActivity extends Activity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        final Button lockButton, UnLockButton; // alarm control buttons
        setContentView(R.layout.activity_home_alarm);
        lockButton =(Button)findViewById(R.id.lockbtn);
        UnLockButton = (Button)findViewById(R.id.Unlockbtn);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.home_alarm, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
    public void display(){}
    public void connect(){}
```

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".HomeAlarmActivity">

    <TableLayout
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:id="@+id/controls"
        android:gravity="center_vertical"
        android:layout_marginTop="50dp"
        android:layout_marginRight="30dp"
        android:layout_marginLeft="30dp"
        android:background="#773"
        >

        <TableRow

            android:id="@+id/controlRow"
            android:layout_marginTop="3dp"
             >
            <TextView
                android:text="Alarm controls"
                android:paddingTop="10dp"
                android:paddingLeft="50dip"
                android:paddingRight="20dip"
                android:layout_height="80dp"
                android:layout_width="300dp"
                android:textSize="30dp" />
        </TableRow>

        <TableRow
```

```
        <TableRow

            android:layout_width="60dp"
            android:layout_marginRight="10dp"
              >
            <Button
                android:layout_marginLeft="10dp"
                 android:layout_width="50dp"
                 android:text="Lock"
                 android:textSize="30dp"
                android:layout_marginRight="10dp"
                 android:id="@+id/lockbtn"
                 >

            </Button>

        </TableRow>
        <TableRow
            android:layout_width="60dp"
            android:layout_marginRight="10dp"
            >
            <Button
                android:id="@+id/Unlockbtn"
                android:layout_marginLeft="10dp"
                android:layout_width="50dp"
                android:text="UnLock"
                android:textSize="30dp"
                android:layout_marginRight="10dp"
                >

            </Button>

        </TableRow>
    </TableLayout>

</RelativeLayout>
```

Appendix B

UML Diagram



List of Figures

[1]

References

Andrew K. Dennis. Raspberry Pi Home Automation with Arduino, chapter 1 -7, 2013.

[2] marketsandmarkets.com. Home Security Solutions Market - Global Forecast & Analysis (2012 – 2017) By Products (Electronic Locks, Sensors, Cameras, Panic Button, Alarms), Security System Solutions (Medical Alert Systems, Access Control & Management Systems, Alarm Systems, Intercom Systems, Video Surveillance Systems, Energy Management Systems, Integrated Security Systems) & Homes (Independent Homes, Condominiums, Apartments). June, 2012.

[3] A.J. B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, and C. Dixon. Home Automation in the Wild: Challenges and Opportunities.

[4] CostHelper, inc. Home Security Cost. http://home.costhelper.com/home-security.html.

[5] Smart Alarm, LLC. http://smartalarmsecurity.com/index.html

[6] developers.android.com. http://developer.android.com/guide/topics/ui/declaring-layout.html.

[7] Taras Liskv.March 14, 2013. http://startandroid.ru/en/lessons/complete-list/220-lesson-16-creating-layout-programmatically-layoutparams.html

[8] techotopedia.com. http://www.techotopia.com/index.php/Creating_an_Android_User_Interface_in_Java_Code

[9] cPrime.com. Agile methodologies. https://www.cprime.com/resources/what-is-agile-what-is-scrum/

[10] Lu Luo. Software Testing Techniques.

http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf.

[11] Jovanović, Irena. Software Testing Methods and Techniques.

http://vipsi.org/ipsi/journals/journals/tir/2009/January/Paper%2006.pdf

[12] Laurie Williams. Testing Overview and Black-Box Testing Techniques

http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf
[13] Laurie Williams. White-Box Testing. http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf
[14] Mark C. Layton. Agile Project Management for Dummies. http://www.dummies.com/how-to/content/agile-project-management-for-dummies-cheat-sheet.html