Independent Security System With Remote Access Thesis Report

Supervisor: Zahidur Rahman (PHD)

Conducted by:

A.T.M Atef Tasnimul Haque 12101145

Syeda Prima Tasnim 10301014

School of Engineering and Computer Science BRAC University

Submitted on 28th December, 2014

Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole or in part, has been previously submitted for any degree.

Signature of Supervisor

_____

Prof. Zahidur Rahman PHD

Signature of Author

_____

A.T.M Atef Tasnimul Haque

Signature of Author

_____

Syeda Prima Tasnim

Acknowledgement

This is the work of Syeda Prima Tasnim and A.T.M. Atef Tasnimul Haque, students of the SECS department of BRAC University, studying CSE and CS respectively starting from the year 2010. The document has been prepared as an effort to compile the knowledge obtained by us during these four years of education and produce a final thesis which innovatively addresses one of the issues of the current practical world. Although there may be more serious and urgent issues that need resolution (hunger, poverty, healthcare, etc.), we felt the need to provide a more intelligent and 'human like' system to existing mobile and web platforms. We intended to develop a system, much like Apple's 'Siri' program, which could behave more like a human in its interactions with its users. However, through seemingly complicated and incomprehensible theories of our own, it became clear that we would fail this major task. Running out of time and patience, we decided to focus our efforts elsewhere. It was then that a professor from BRAC University had mentioned a problem that set us off into developing the system we now call RASS. The problem was to simply monitor the access granted to students to a certain room and log them for security reasons. Initially this seemed like something achievable and indeed it was. As we began developing the system, our supervisor started adding to it more and more features and functionalities to such a point that the system is capable now to provide an enterprise solution to corporations. Even so, this system has been piloted by one of the larger corporations in Bangladesh and is soon to be released after completing the beta testing. We had plenty of help from Mr. Moin Mustakim, SECS Dept., BRAC University, with the paperwork. The generation of this report would not be possible without his help.

Table of Contents

List of figures

Abstract

Independent Security System with Remote Access

This thesis essentially consists of three primary parts and has been described throughout according to their functionalities. At the core of the system lies the web server which is central to everything. Then there is the microcontroller unit itself which carries out actions in the physical world through electronic relays. The final component is an android and a web application that provides users with the interface to access the functionalities and features of the actual hardware. In a gist, a user can manipulate electronic switches located at any remote site, equipped with the microcontroller unit, through either the web or mobile application..

The hardware consists of a microcontroller unit, a Bluetooth shield, a GSM shield and six electronic relays. It uses polling to learn about commands given by users through the web or android application, from the central server. The system has an average poll time of fifteen seconds and an average fail rate of forty percent.

The central server runs on a RHEL 6 server hosted on the cloud and will need load balancing once the number of concurrent connections (Sites) exceeds one thousand. It uses a PgSql Database server with a minimum storage of ten GigaBytes.

The user interface consists of two parts, an android application and a web application. The android application provides a user with simpler options to work with while the web application provides complicated functionalities and features. The android application uses the API provided by the web application to fetch and push data to the central server. The web application is built on the Laravel framework for php (Version 4.2) and uses most of its features like its Templating Engine “Blade” and its Object Relational Model “Eloquent”.

Overall, the system is one of its kind as it is the only open source product available with its distributed set of features and functionalities throughout multiple platforms. However, there is still much room for further developments as the project is only in its childhood.

Chapter 1

With the goal of building a new generation of security system which can be controlled virtually, we've come up with our project called, RASS (Remote Access Security System).

1.1 Introduction

This is an embedded system that will integrate easily with any existing system. In other words, it is a module that can accept connections from more than ten peripherals in order to switch them (Hot or Cold). So one could hook up an electronic lock, TV, AC, generator or any other electronic device of choice to the module and control when they switch (on or off).

It generates and stores logs at the remote site; notifies the central server in case of unauthorized access. This whole system can be placed either in an intranet or the internet, depending on its application. The system provides a web based GUI to control and operate itself. We started the thesis by the end of 2013 and have completed it by 2014. Our main focus was to provide a solution that would be efficient and be very user friendly. Hence the choice of platform was the web, as it provides great tools to develop an interactive GUI that can help users navigate and operate the system.

1.2 Motivation

We found a gap between existing security systems of our households and that of corporate offices and apartments abroad. We noticed that almost all of the buildings in Dhaka, and most other cities in the developing countries do not have any control mechanism that grants access to users through any mobile or web platforms. After some intense research and study we were confident enough to resolve this issue in an efficient and feasible manner. Hence we started work

on this next generation security system that spans across both the mobile and web platforms. At that very beginning we started researching about micro controllers.

We started with the Arduino Uno and moved on to the Arduino Mega. We thought of developing a system which would allow an user to control and monitor a remote location. With the help of our supervisor we started our project research. At that moment, there arose a need to control access of students into BRAC University's Robotics Lab. It became a security issue as parts and tools were constantly missing. To protect the equipment of the laboratory the BRAC University faculties encouraged us to provide a solution to this problem. Hence, we decided to start work on our thesis and had in aim specifically the logging facility that we so very much needed. Finally, with the eight months of hard work and research we ended up developing our thesis project. This is the complete system; it is called **RASS** in short and is responsible for maintaining remote sites.

1.3 Thesis Outline

The thesis consists of nine chapters in all and are outlined below. Each chapter consists of at least one or more sections that describe a specific part of that individual chapter. A detailed description of each of these sections are also outline below.

Chapter One details the purpose, aim and motivation for the development of this thesis. It has three sections - introduction, motivation and thesis outline.
The *Introduction* section describes the purpose and aim for the development are mentioned.
The *Motivation* section describes the motivation behind the whole development.
The *Thesis Outline* section is self explanatory.

Chapter Two describes the problem that this thesis targets to eliminate, the current solutions that exist in the market and the solution that this thesis proposes. It has three sections - background study, problem description and proposed solution.

The *Problem Description* section describes the problem that this thesis is attempting to solve in detail.
The *Previous Works* section lists the most recent and promising work done by others to provide a similar solution.
The *Proposed Solution* section describes in detail the solution that this thesis provides in order to eliminate the problem described.

Chapter Three describes the entire system design and the main components. It has three sections - microcontroller unit, mobile application and web application.
The *Microcontroller Unit* section details the design and overview of the microcontroller unit that makes up the independent security system module.
The *Mobile Application* section details the android classes, layouts and the manifest file. This also outlines the communication between the microcontroller unit and the central web servers.
The *Web Application* section details the entire application structure and the database design along with the restful api routing.

Chapter Four describes the entire system architecture which details all the separate individual components making up each major unit and the purpose for their requirement. It has four sections - system design, microcontroller unit, mobile application and web application.
The *System Design* section describes the complete architecture of all the three major components. This includes the schematics for the microcontroller unit design, the database design for the web application and the restful api for the hosting server.
The *Microcontroller Unit* section contains ten sections, each for one of the components used inside the main controller unit itself. The ten sub sections are *arduino uno*, *arduino mega*, *LCD*, *GSM shield*, *bluetooth shield*, *micro SD shield*, *wiegand RFID reader*, *electromagnetic lock*, *software serial communication* and *I2C serial communication*. These sub sections describe the utility each of these modules provide to the entire functionality of the microcontroller unit.
The *Web Application* section contains four sections detailing the developments of all the server side and client side programming, including the usage of various web development frameworks.

This section has four sub sections - *database design*, *model view controller design pattern*, *server side programming* and *client side programming*. These sub sections describe in detail the implementation of all of the mentioned modules.
The *Mobile Application* section contains two sections detailing the development of the android application needed to interact with the independent controller unit and the AJAX calls made to the server. This section contains two sub sections - *native android application* and *asynchronous javascript calls*.

Chapter Five describes the integration of the primary components into the whole system. It details the combination of the microcontroller unit, the web application and the mobile application to provide the service referred throughout as 'remote access security system'. This chapter has two sections - *cloud deployment* and *system interfaces*. Each of these sections describes the whole development process needed to provide the RASS service.
The *Cloud Deployment* section details the implementation of the restful api needed to provide an interface to both the microcontroller unit and the mobile application.
The *System Interfaces* section lists and details all of the interfaces that the system specifically provides to the user for access control. This section has two sub sections - *web application functionalities* and the *mobile application functionalities* that the user can take advantage of.

Chapter Six describes the entire system's performance and reflex time for all the different actions. This chapter has three sections - *basic analysis* and *system redundancy*.
The *Basic Analysis* section details three different analytics that are relevant to the system and has two sub sections which are *GPS analytics* and *RFID analytics*.
The *System Redundancy* section provides a detailed evaluation of the failsafe mechanism included in the system and it's advantages. This section has two sub sections - *RFID unit* and *micro SD card unit*.

Chapter Seven describes the drawbacks that the system has incorporated within and has two sections - *system vulnerabilities* and *system improvements*.

The *System Vulnerabilities* section consists of two sub sections - *DDOS attacks* and *Power Drainage*. These sub sections describe in detail the security issues associated with the system.
The *System Improvements* section consists of one sub sections - *RPC communication*. This sub sections describes in detail the advantages that the system will provide with the implementation of each of the specified improvements.

Chapter Eight describes the concluding notes of the authors for this thesis and contains three sections - *deployment*, *limitations* and *risks*.
The *Deployment* section describes in detail the entire procedure needed to deploy the system at a remote location and also set up the central server.
The *Limitation* section describes in detail all the limitations of the system from a user's point of view.
The *Risks* section describes in detail all the relevant risks of using this system.

Chapter 2

Although a lot of research has already been done in this field and many techniques, far advanced than that used in this thesis, have been implemented by others, no other solution like this exists to provide the service that this thesis has derived to provide to an end user. To understand what we have accomplished in this thesis and how it can revolutionize the future of security systems, we must look at the problem that we have attempted to solve. After a clear understanding of the problem and its solutions, we can easily compare how the previous works by others were not service oriented and hence have failed to solve the problem, where we have succeeded. Lastly the solution proposed by this thesis will be explained in great detail.

2.1 Problem Description

After some intense brainstorming we have found that there is no current system that allows a user to virtually access and control multiple electronic devices at one or more remote locations. Moreover, no current system exists which allows a company or a large corporation with multiple remote sites, located both locally and internationally, to both grant access control and organize those remote locations into zones.

As an example, the telecom service providers in Bangladesh has more than five thousand BTS centers where they store very expensive equipments provided by their vendors. Currently, they do not have any automated centralized system that would allow them to provide access control to the various electronic components located at these sites. These companies have employees spanning from hundreds to thousands who might need access at any of the remote sites. The current system requires them to fill out paperwork to acquire a master key that grants them entry into a remote site. This system is very inefficient and slow and can be very difficult to maintain. Keeping this problem in mind we developed our solution which solves all of the major problems

that these companies are facing. Our solution allows any user to obtain access to any given remote site through an RFID card which has to be granted access via the web application interface. A user thus has to notify the central authority to be granted access to the desired remote site and once the access has been granted the user can easily use his RFID card to grant entry into the site. Besides access control, any activity at these sites can also be monitored from the central monitor station via the web application interface.

Another example would emphasize on an average family living in either the suburbs or the urban regions of a country. Considering Dhaka city itself, there are serious security issues with certain developing areas of the city. In such areas, users with access to smart phones can easily gain access to their desired electronic devices with their smart phones, including an electromagnetic lock, which would prevent any unauthorized access.

In most cities in and around the country, there is no control mechanism which grants the user to access to a system via mobile or website with all the features mentioned above. Secondly, there is no such mechanism which provides a detailed login summary about any authorized or unauthorized entries. Furthermore, the RASS system provides the central monitoring station to generate and analyze various graphs and reports generated via the web application interface.

2.2 Previous Works

According to the studies, there are various remote procedure call protocols have been developed. Sun Microsystems is the inventor of the RPC paradigm, which is now known as the Open Network Computing Remote Procedure Call.

Among all the RPC analogues and implementations found elsewhere, we found D-Bus, CORBA, JSON-RPC and Java Remote Method Invocation (Java RMI) helpful for our thesis. The functionalities of these protocols are different but all of them uses RPC.

D-Bus, is an open source inter-process communication (IPC) program, the functionalities are similar to CORBA. It is a system which communicates concurrently-running computer programs (processes) with each other.
D-Bus provides two types of functionality: First, communication between desktop applications in the same desktop session; to allow integration of the desktop session as a whole, and address issues of the process lifecycle. Second, communication between the desktop session and the operating system, where the operating system would typically include the kernel and any processes.

CORBA, **Common Object Request Broker Architecture** provides remote procedure invocation via an intermediate layer, which is called the *object request broker*. It uses an object-oriented model although the systems that utilize CORBA do not have to be object-oriented. CORBA enables three types of collaboration, they are; collaboration between systems on different operating systems, programming languages, and computing hardware.It is an example of the distributed object paradigm.

Java's **Java Remote Method Invocation** (Java RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC). The original implementation depends on Java Virtual Machine (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another. The protocol is known as Java Remote Method Protocol (JRMP).
A CORBA version was later developed in order to support code running in a non-JVM context,

JSON-RPC is an RPC protocol that uses JSON-encoded messages. It is a very simple remote procedure call protocol which is encoded in JSON, defining only a handful of data types and commands. JSON-RPC allows for notifications, where it sends data to the server which does not require a response and for multiple calls to be sent to the server which may be answered out of order.

2.3 Proposed Solution

After much needed researching and understanding of the problem itself, we have proposed a feasible solution that addresses all aspects of the problem and solves them quite efficiently. During the design phase of the system our efforts revolved around the core of the problem itself, ie: providing access control to users dynamically based on their access, granted from the central monitoring station.

Due to the unavailability of ethernet or other fast internet connectivity like wireless internet or 3G internet in remote areas, we found it feasible to implement GPS based internet using a sim as the primary source of connectivity. This also sets up the foundation for RPC based communication implementation as during the GPS connectivity initiation phase, an unique real IP is allocated to the modem. This IP can be used to activate full-duplex protocols of communication like TCP IP.

The primary security if provided by an electromagnetic lock connected with an RFID reader to grant or deny access to the site based on the access privilege allocated to an RFID card. This locking mechanism has been implemented in such a manner so as to provide access both via the web application interface and the mobile application interface, based on the mode of communication being used by the user. All access privilege data has been stored in a micro SD card to maintain redundancy in case of any issue with either modes of communication.

Therefore, in case of any internet failure or bluetooth failure, a user can use an RFID card to gain access to the system independently. This is the fundamental fail safe for the system.

In addition to five existing ports available for connecting any electronic device of choice, an additional five ports remain for configuring and adding further devices if desired. Each of these ports are digital pins on the microcontroller and can be provided with digital outputs, thus allowing for more than zero or one outputs if required. The ports, once activated or deactivated trigger internal relays which in turn switch (on or off) external electronic devices that have been connected to the respective ports.

The method used for synchronization with the central web server is known as *polling* and is guaranteed to succeed. **Polling** in this context refers to the process of continuous packet transmission from the microcontroller unit to the central web server to keep itself updated as to the status of each port (on or off). The microcontroller unit polls four times a minute to acquire the current state of the system and once a successful poll happens, the state is updated instantly and maintained in that state until the next successful poll occurs. In case of prolonged consecutive unsuccessful polls the systems reboots the GSM modem and initiates the synchronization process all over again, still maintaining the current state of the system.

The synchronization process is quite simple in itself as every poll is nothing but a HTTP GET request which is made to the central web server. The response made by the server is also a HTTP frame and the response data contains the state that the system should resolve to. Every successful poll is logged at the central server which is used to provide an approximate estimation as to the downtime or uptime of any remote site. If the server doesn't log a successful poll for a prolonged period of time, it warns the central monitoring station via the web application interface. Furthermore, if during synchronization any unauthorized access is found in a poll, the web server alerts the central monitoring station via the web application interface.

The electronic devices connected to the ports can also be controlled via the mobile application interface. The bluetooth communication mode of the system allows for any changes to system's

state via the mobile application interface. Once the system's state has been altered, the system automatically retains the latest change while the mobile application notifies the central server of the changes made. Hence, changing the system's state from either interface is instantly notified to the central web server which always stores the current state of the system.

Chapter 3

The RASS system as described above consists of three core parts - the *microcontroller unit*, the *web application* and the *mobile application*. The microcontroller unit is standalone and needs to be installed at a remote location which is to be monitored from the *central monitoring station*. The microcontroller unit itself consists of two separate units - the **master** and **slave** units. The Web Application provides an **administrator** with a *graphical user interface* (**GUI**) to monitor, control and grant access privilege to any number of **users**. The interface provided to the administrator is an intermediary to the web servers as each action performed on the web application itself is passed on to the web server in turn. The Mobile Application provides **users** with *privileged access rights*, an interface to control and update the state of the system, ie: switch on an electronic appliance, via a smartphone. All of these functionalities and features are described in brief below.

3.1 Microcontroller Unit

The sole purpose of the microcontroller unit is to remember the current state of the system, ie: the current switching condition of the appliances, and synchronize this state with the central server. The master and slave units have been separated out so that in case of any communication error, like the absence of internet or bluetooth, the system does not become a black box. Thus we have guaranteed a failsafe even in the worst of cases. This is achieved by the *Singleton Design Pattern* implementation in the programming of the units. The master unit is only responsible for synchronization of the system with the central web server. The slave unit is responsible for any changes to the system's state itself, ie: switching of the relays.

3.2 The Web Application

The web application is the primary interface that an **administrator** must use to operate the entire RASS system. Administrators are authenticated by a username and password before allowing them access to the *central monitoring station*. After logging in, the administrator can perform one of seven core functionalities:

The admin can trigger any switch at any remote site or zone from the *sites* or *zones* panel.
The admin can grant or deny *access privilege* to multiple **users** for multiple sites or zones at the *access rights* panel.
The admin can monitor *logs* generated at all sites and zones at the *logs* panel.
The admin can view *reports* generated by analyzing the logs for remote sites at the *reports* panel.
The admin can *create*, *read*, *update* and *delete* users, sites and zones on their respective panels.
The admin can view notifications from individual sites and zones at the *notifications* panel.
The admin can *deactivate* the microcontroller unit at any site or zone from the *activity* panel.

These seven panels provide the administrator with all the core functionalities of the system and can be easily navigated to from the top navigation bar. Aside from the administrator, general users can also log in to the web application using their username and RFID card number as password through the login interface. Once logged in, the user can view and update the current status of any or all of the sites that he has been granted access to. The user may also update his personal information and RFID card number.

3.3 The Mobile Application

The mobile application is another side of our security system to control the system virtually. First of all, the user will login using the username and password/RFID to get started. Once logged in,

the user can manage all the connected appliances through the mobile application interface. In case of any unauthorized access attempts, the remote server will be alerted.

Four options are available after logging in; *Sites, Log, Zones* and *Map.*

The Sites option displays a list of sites that the user has been granted access to. After selecting a site from the list, a user can control the switches.

The Log option displays a list of sites that the user has access to. Upon clicking on a site, all the logs generated by that site is listed for the user to view.

The Zones option displays a list of zones that the user has been granted access to. After selecting a zone from the list, a user can control the switches.

The Map option shows the geo- location of the sites that the user has been granted access to.

A user can log out anytime by simply pressing the logout option which takes the user back to the login page.

Chapter 4

Before beginning the design phase of the product, we collected numerous backlogs that helped describe the product in great detail. We followed standard software development techniques such as writing **use cases** and **user stories**. This helped us further clarify misconceptions and confusions regarding the design of the **RASS** system.

4.1 System Design

Gathering from the backlogs, use cases and user stories, we swiftly designed the first logical prototype of the system. The main components of the system are the **microcontroller unit**, the **central servers** and the **mobile** and **web applications**.

The *microcontroller unit* is the physical component that must be present at the remote site and has to be installed manually. Manual installation will involve connecting all the electronic appliances that are desired to be controlled remotely. Once installation is complete, it should be ready to process commands send to it via the web servers.

The *web* and *database servers* are located on the cloud and provide the gateway for sending commands to the microcontroller unit located at the remote site. The web server is maintained by the web application which is hosted on the server itself.

The *web application* is built on a php framework and it serves requests made by the microcontroller unit at any remote site. While the service is made, the application updates the state of the system based on any changes made by a user. After a request has been served and the microcontroller unit at the remote site has received the transmission, it resolves its state to match that mentioned by the web server. Thus the whole synchronization process is resolved.

4.2 Microcontroller Unit

The master unit has four main components that are crucial to its functioning. It is connected to a GPS modem, which provides internet connectivity for synchronization. It is also connected to a bluetooth adapter, which provides bluetooth connectivity with nearby devices. An LCD is connected to this unit as well in order to print various runtime messages. The most important component is the **software serial** *interfacing* with the slave unit. This interface allows the master unit to send commands to the slave unit.

The slave unit also has four core components connected to it, one of which is responsible for the actual triggering of connected electronic appliances. Six relays make up the *relay* component of the slave unit and are responsible for triggering switches. A micro SD card has been interfaced with the slave unit in order to store local logs of access and also the list of privileged users who have access. Whenever a synchronization is successful the user access list stored in the SD card is also updated accordingly. The failsafe of the system is assured by the RFID reader which is also interfaced with the slave unit. The slave unit will grant any user with a privileged RFID card access (by unlocking the electromagnetic lock). However, the most important component of the slave unit is the **software serial** *interfacing* with the master unit. This interface allows the slave unit to receive commands transmitted by the master unit and execute them accordingly, thus keeping the state synchronized with the central web server.

4.2.1 Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328 and it has 14 digital input/output pins and 6 analog inputs. This microcontroller is the brain of the slave unit of the entire microcontroller unit and is interfaced with the components mentioned above. It receives all control commands from the master unit via software serial communication.

4.2.2 Arduino Mega

The Arduino Mega is a microcontroller board based on the ATmega1280 and it has 54 digital input/output pins and 16 analog inputs. This microcontroller is the brain of the master unit and is interfaced with all the components mentioned above. It sends all control commands to the Arduino Uno of the slave unit via software serial communication.

4.2.3 LCD

This is a basic display which uses a library named LCD library for arduino. This library allows an Arduino board to control Liquid Crystal Displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The LCD is used to display certain outputs when a successful poll occurs and also for prolonged unsuccessful polls. It also shows the status of the system which is either *online* or *offline*.

4.2.4 GSM Shield

This module uses the Serial ports 50 and 51 for Serial Communication between the SIM900 GSM shield and the Arduino Mega. Simple “AT” Commands have been used to initiate and perform HTTP requests like GET and POST. A detailed list of these commands can be obtained online from the SIM900 documentation. Simple AT commands have been used by the system to make HTTP GET requests to the central web server.

4.2.5 Bluetooth Shield

This module uses the Serial ports 18 and 19 for Serial Communication between the Bluetooth Shield and the Arduino Mega. The shield is almost plug and play and no configuration is required to establish communication. The Serial1 ports of the Arduino Mega have been used for serial communication between the bluetooth shield and the microcontroller.

4.2.6 Micro SD Card Shield

The SD Card shield uses the Arduino Mega's MISO and MOSI pins to establish a master slave communication protocol using the ICSP headers. The Chip Select (CS) pin is used to specify which pin the Mega uses to read or write data to the SD card. Once the connections are made, the SD card can be accessed via the Arduino Mega. The SD Card module is needed for the system to be actively secured in case of any network failure causing a connection sever between itself and the central servers. In such a case, the RFID numbers of authenticated users will be retained by the system through the SD Card and access can still be viable. We were unable to encrypt the data stored in the SD Card and therefore no level of security can be provided.

4.2.7 Wiegand RFID Reader

The Wiegand interface is a de facto standard commonly used to connect a card reader or keypad to an electronic entry system. Wiegand interface has the ability to transmit signal over long distance with a simple 3 wires connection. The RFID reader has been used through the open source Wiegand32 library to establish communication with the Arduino Mega.

4.2.8 Electromagnetic Lock

An electromagnetic lock or magnetic lock, is a locking device. Generally it consists of an electromagnet and an armature plate. These two parts: electro magnet and armature plate make the electromagnetic lock, where the electromagnet portion of the lock is attached to the door frame and a mating armature plate is attached to the door. When the door is closed the two components are in contact. When the electromagnet is energized, a current passing through the electromagnet creates a magnetic flux that causes the armature plate to attract to the electromagnet, creating a locking action, therefore the door is locked. Because the mating area of the electro magnet and armature is relatively large, the force created by the magnetic flux is strong enough to keep the door locked. Even under any large amount of stress the door remain locked because of the magnetic force. One of the relays of the slave unit of the microcontroller unit is connected to an electromagnetic lock. Based on the application logic, triggering the relay results in opening or closing the lock.

4.2.9 Software Serial Communication

Serial Communication is used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. The Arduino Mega has 5 separate software serial ports. Out of these five, three have been used, by the GSM shield, the bluetooth shield and the TX RX for usb communication. The arduino uno has only one software serial port and that has been interfaced with the master unit to receive control commands.

4.2.10 Serial I2C Communication

I2C stands for **inter-integrated circuit** and it is basically a serial computer bus used for communication between multiple master and slave units that are usually low speed peripherals. Almost all arduino boards have this bus available to allow for integration with any legacy systems and is called the **ICSP** legacy jumper connections. We have used the ICSP jumpers on both the arduinos on the master and slave units to establish our own *I2C bus*. After the creation of the bus itself we have used the I2C communication protocol to programmatically declare the master and slave units. This setup was initialized using the **Wire.h** library which implements the I2C protocol for communication. During the setup it was crucial to synchronize the clocks of both the master and slave units for the communication protocol to be instantiated.

4.3 Web Application

The web application is the core piece of software that the RASS system provides as a service. In other words, the web application is a **SAAS** that completes and constructs the whole system itself. The application provides a GUI to the *administrators* and *users* to access, control and monitor all the remote sites connected to the central web server, from where the web application is served via the internet. The application is built on a core PHP backend framework (Laravel) that uses the MVC design pattern. As for the frontend, Angular JS, a javascript framework is used which also uses the MVC design pattern.

The application is designed using the **Laravel** framework and there are individual **controllers** that serve a **view** based on the **model** that needs to be accessed through database queries. All queries made to the database are made through an object relation mapping called **Eloquent**. These views are served using an apache server and are generated using the **Blade** templating engine. All of the views are stored on the server and are cached in the server.

4.3.1 Database Design

The database used for the web application is a relational database management system and is called **MySql**. The tables and their relations are described in the figure below:

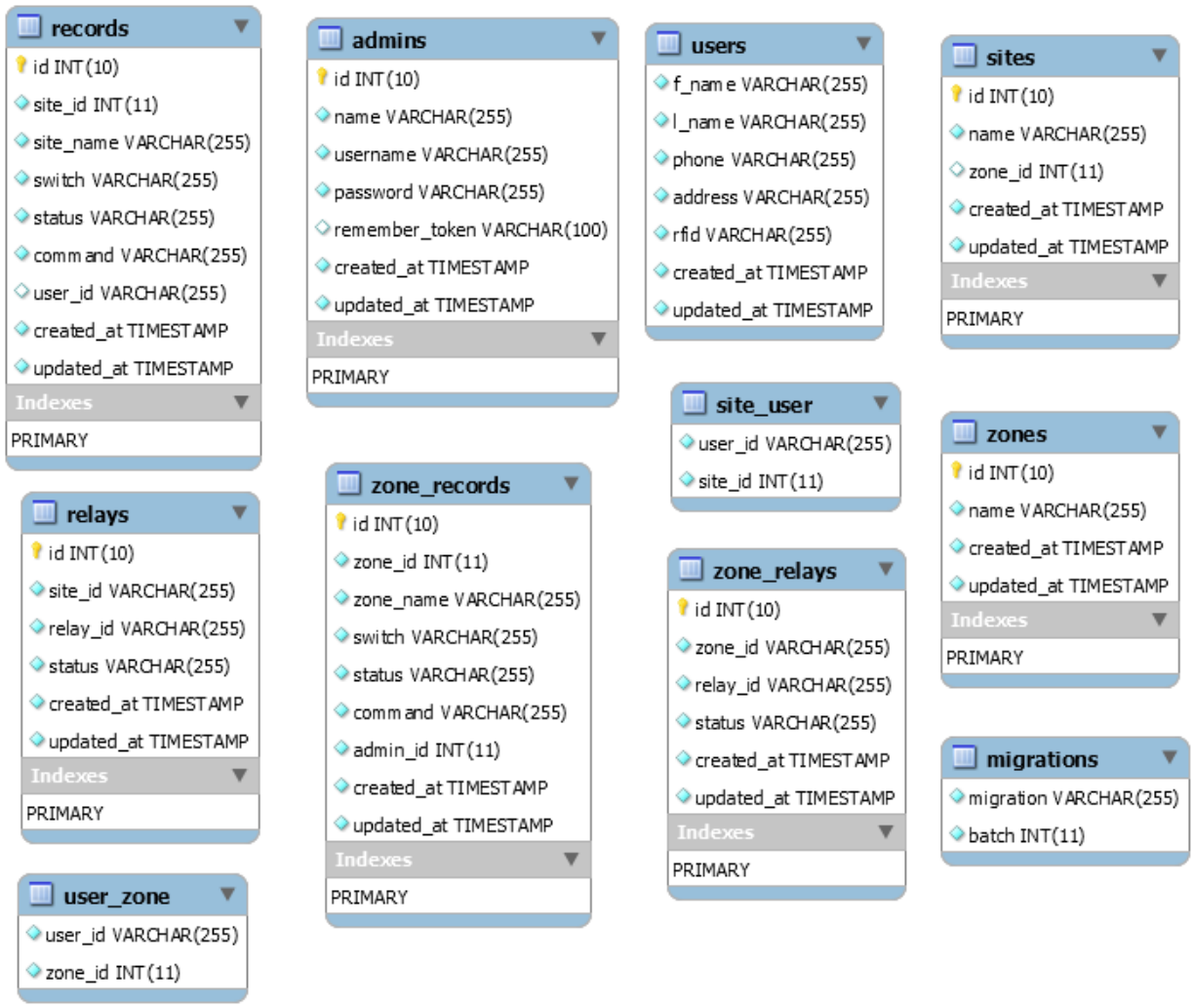

Figure 4.3.1.1 Database Management System

4.3.2 Model View Controller Design Pattern

To maintain and scale the system we have implemented a MVC PHP framework called Laravel. The model view controller design pattern consists of **controllers** (logic), **views** (markup) and

models (objects). A controller for a class contains all the application logic regarding any end to end rendering of web pages relating to that class. It renders any views that are required and populates the views with data accumulated by querying the model for a specific class.

4.3.3 Server Side Programming

The RASS System has fourteen controllers - *BaseController*, *HomeController*, *ImageController*, *LoginController*, *RecordController*, *ReportController*, *SiteController*, *SiteUserController*, *SiteZoneController*, *UserController*, *ZoneController*, *ZoneSiteController* and *ZoneUserController*. The **BaseController** is the parent controller from where all the other controller inherit housekeeping properties.

The RASS System has eight models - *Admin*, *Record*, *Relay*, *Site*, *User*, *Zone*, *ZoneRecord* and *ZoneRelay*. These models provide the interface needed to define the object relational mapping between the database and the logical models.

The RASS System has eight major view directories - *reports*, *site_user*, *site_zone*, *sites*, *users*, *zone_site*, *zone_user*, *zones* and the *base* views directory. These directories contain all the individual views that have been included or generated by the controllers.

4.3.3.1 Laravel Framework

The laravel framework is based on the Symfony framework for php and uses the MVC design pattern to organize code and provide readability and maintainability. It ships with its own ORM called Eloquent. We have used this framework along with all of its core functionalities (ORM, Controllers, Migrations and Seeders) to develop our web application. The github repository for the application lists in great detail all the commits and merges made during its development and

can easily be forked to make any upgrades. The application was built using the Laravel 4.2 framework and uses all of it's dependencies

4.3.3.2 Object Relational Model

In order to integrate databases with their data types and methods, an object-relational model has to be designed. It is essentially a relational model that allows users to integrate object-oriented features into it. The benefits that are offered by the Object-Relational Model include: **Extensibility, Complex types, Inheritance.**

Extensibility - the users are able to extend the capability of the database server.

Complex types - Complex types offer better flexibility in organizing the data on a structure made up of columns and tables. It allows users to define new data types that combine one or more of the currently existing data types.

Inheritance - users are able to define objects and tables that obtain the properties of other objects, also to able to add new properties that are specific. ORDBMS, The object-relational database management systems provide an addition of new and extensive object storage capabilities to the relational models at the center of the more modern information systems of today.

We have used the Eloquent ORM to define our models and have thus gained access to all of it's features helping us develop the web application faster with minimal or almost without having to make any sql queries directly.

4.3.3.3 Restful API

REST is the underlying architectural principle of the web. The clients and servers can interact without the client knowing anything beforehand about the server and the resources it hosts. In that case, the key constraint is that the server and client must both agree on the media that is used, which in the case of the web is *HTML*.

The API for our web application follows a restful pattern and has made it simpler for us to integrate the system with its counterparts - the mobile application and the microcontroller unit. The API for the current release of the version is given below:

sites_url : http://rass-enterprise.herokuapp.com/sites/

zones_url : http://rass-enterprise.herokuapp.com/zones/

update-switch : http://rass-enterprise.herokuapp.com/updateSiteRelay/

relaystatus_url : http://rass-enterprise.herokuapp.com/relays/

4.3.4 Client Side Programming

The frontend of the web application has been designed with the Angular JS MVC framework created by Google. The **app.js** file contains the configuration files for the application. The **controllers.js** file contains all the controllers that the application contains. There are eight controllers - *SitesCtrl*, *SwipersCtrl*, *AdminsCtrl*, *ZonesCtrl*, *RelaysCtrl*, *LogsCtrl*, *ReportsCtrl* and *LoginCtrl*. These controllers use the services defined in the **services.js** file. There are two services for each **resource** and they use the restful API mentioned above to fetch and push data to the server asynchronously.

4.4 Mobile Application

The Mobile Application uses Ajax calls through an API to fetch and push data to the web application, which in turn forwards the requests and responses to and from the central servers. The Web Application is very heavy and uses an MVC framework and makes complicated queries to the database servers through an API. The frontend of the application is built with Angular JS, and uses numerous dependency injections. It also uses the Promise, Singleton and Factory Design Patterns.

4.4.1 Android Native Application

RASS, mobile application is an android native application.There are seven Activity classes in this application. They are: MainActivity.java, Home.java, Sites.java, Zones.java, Map.java, Log.java and finaly ControlDevice.java. We have implemented BaseAdapter extension in order to show the list of sites or zones in a ListView. There is a JSON.java and JSONPerser.java class in order to read the json API.

The wireframe (Figure 4.4.1.1) [1] shows the flow of the mobile application. The application starts from MainActivity.java, where the user sees a login option by username and password/RFID. There is a “Remember me” option, if the user checks the checkbox the application saves the username and password so that next time when the application runs it can restore the saved username and password in order to login. We have implemented “SharedPreferences” here to save the username and password.

After logging in, a new layout shows up, named Home.java. Home.java contains four buttons: They are; *Sites, Log. Zones* and *Map.* There is another button named “Logout” which simply logout the user from the application.

When we click the button “*Sites”*, it opens Sites.java activity. This activity contains an AsyncTask class,where the JSON API works. It executes a json call to the server and gets the

information from the remote server. There is a ListView which uses BaseAdapter to show the site names in a list from the remote server. We can click on the item from ListView. When we click on a name of a site, it opens the ControlDevice.java activity. ControlDevice.java contains BluetoothAdapter which is necessary to use the bluetooth module. If the bluetooth device of the mobile is **on**, then we can use our bluetooth to control the electronic devices, if the bluetooth is **off** then we use the internet to control our electronic devices. In the case of "Bluetooth Off" situation there we have used AJAX calls to the remote server. In the case of "Bluetooth On" we do not make any AJAX call, we simply connects to the arduino board via bluetooth.

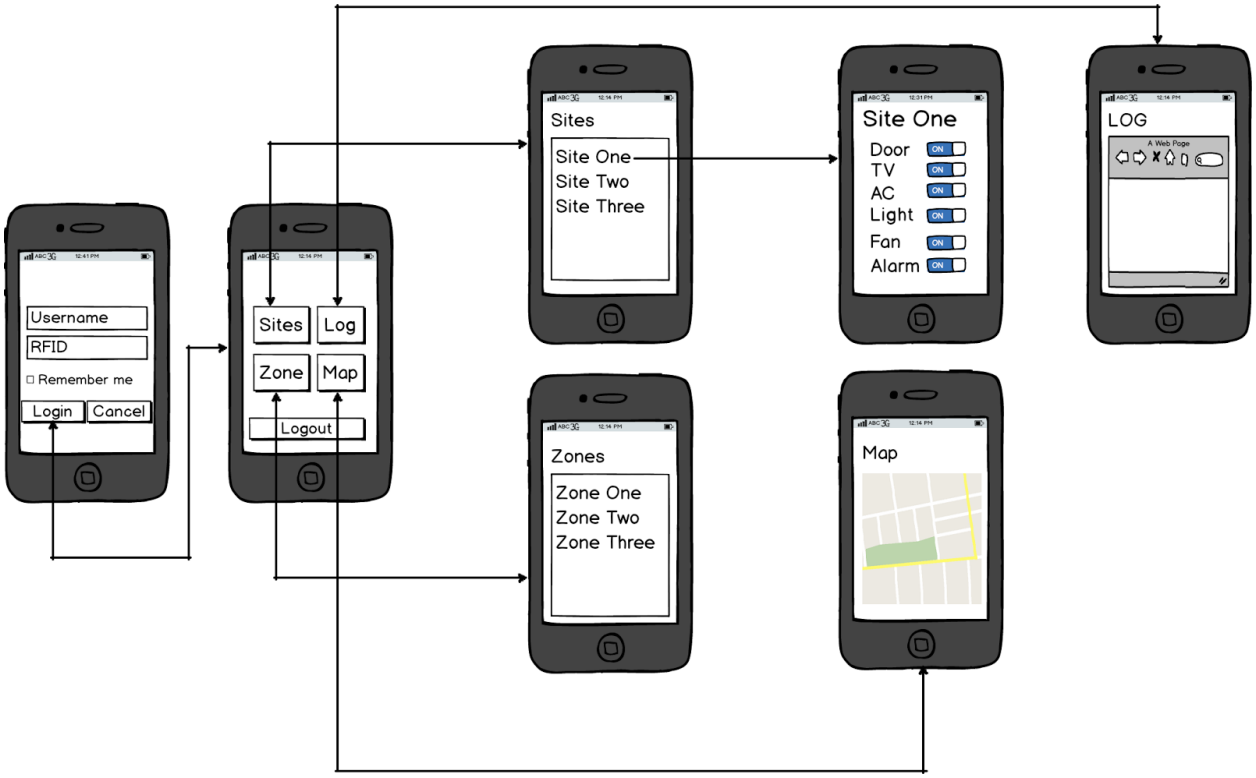

Figure 4.4.1.1 WireFrame of the RASS application

When we click the button *"Zones",* it opens Zones.java activity, and just like the Sites.java class, it also executes a json call to the server and shows the zone lists in a ListView via BaseAdapter.

4.4.2 Asynchronous Javascript API Calls

JSON stands for JavaScript Object Notation. It is a simple format through which we can interchange data that can be easily read by humans and machines. JSON is a text format that is language independent.

JSON represents data in a text format so that can be easily parsed. It uses two different of structures: Collection of name/value pair and Array.

So using these two structure we can transfer data between two machines in a simple and efficient way. The first structure, Collection can be used to model object because an object is a collection of attributes that hold some values. The second structure, Array can be used to model list, array of objects and so on.

In our application we have four json url's. sites_url, zones_url, update_switch, relaystatus_url.

The API's are mentioned above in section 4.3.3.4 Restful API.

In Android, and in general in all environments, there are two type of operation: Convert java class to JSON data which is Serialization and Parse JSON data and create java classes which is Deserialization. We have followed deserialization in our application. We parsed the JSON data and created java class for each JSON url. The first step is instantiate a parser that helps to get the values inside JSON.

JSONObject jObj = new JSONObject(data); where data holds the JSON string. Now, if we want to get any data from the JSON, e.g if we have a JSON which has email address in it and we want to get that email address then we will put that in a string, like this:

String surname = jObj.getString("email"); here "email" is the data we want to get. Using these pieces of code we can handle JSON in Android.

Chapter 5

To implement the whole system, three major steps need to be completed. Firstly, the microcontroller unit(s) needs to be installed at the remote site(s) and all required connections set up. Once this is completed, the web server needs to be online to communicate with the remote sites. After the web server is online, the mobile application needs to be downloaded into a smart phone to complete the system.

Setting up the microcontroller is very basic as the schematic described below demonstrates :

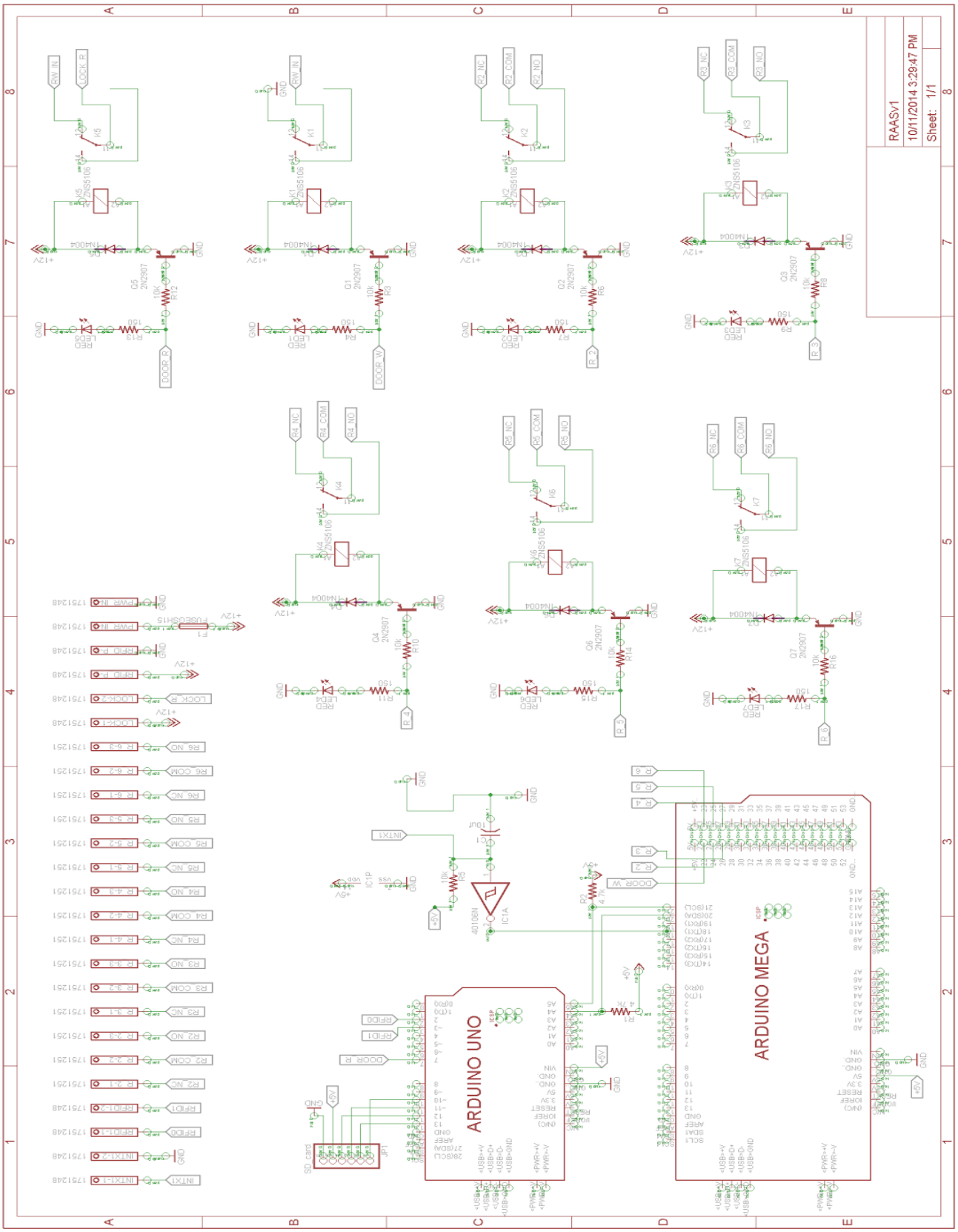

Figure 5.1 Schematics Diagram

5.1 Cloud Deployment

To deploy the web application to the cloud, we have used the free heroku cloud hosting with the average initial dyno for load balancing. Reading heroku's documentation provides application specific console based deployment commands that have been used to successfully deploy the application.

Heroku's deployment tool can be downloaded from their website and must be installed in the computer from where the deployment is originating. The application is then simply fetched from its repository in Github and deployed in any one of heroku's servers online.

5.2 System Interfaces

An interface is a shared boundary by which computer system exchange informations between software, hardware, peripheral devices, humans and a combination of all. The section System Interface consists of web application functionalities and mobile application functionalities. The user interface of the web application is described in the part of web application functionalities and the user interface of the mobile application is described in the part of mobile application functionalities.

5.2.1 Web Application Functionalities

The web application provides an **administrator** an interface to *monitor*, *control* and grant *user access rights* to multiple **users** at any remote **site** or **zone**. The home page of the web application is shown on Figure 5.2.1.1, where an administrator must log in first. An administrator can *create*, *update*, *read* or *delete* **(CRUD)** a site, zone and user, after logging in to the system.

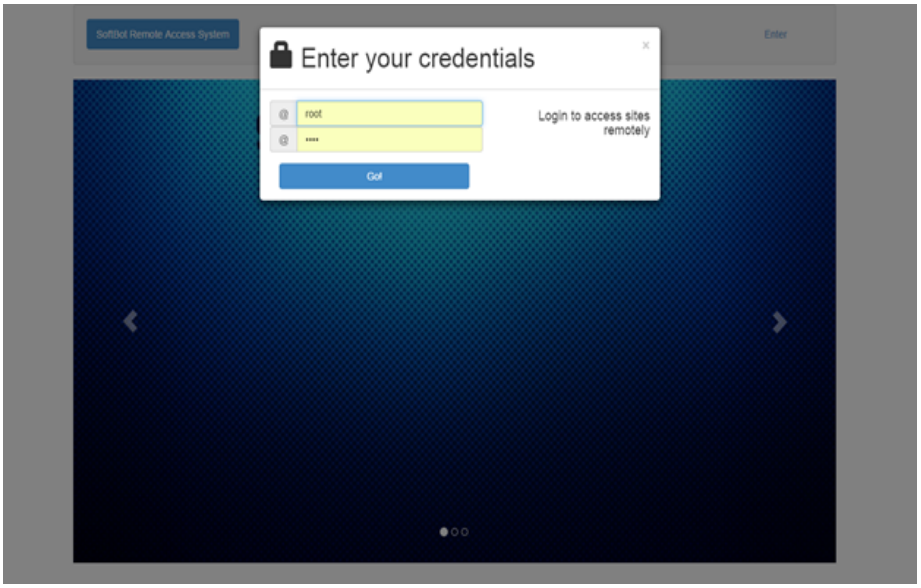

Figure 5.2.1.1 Login Panel

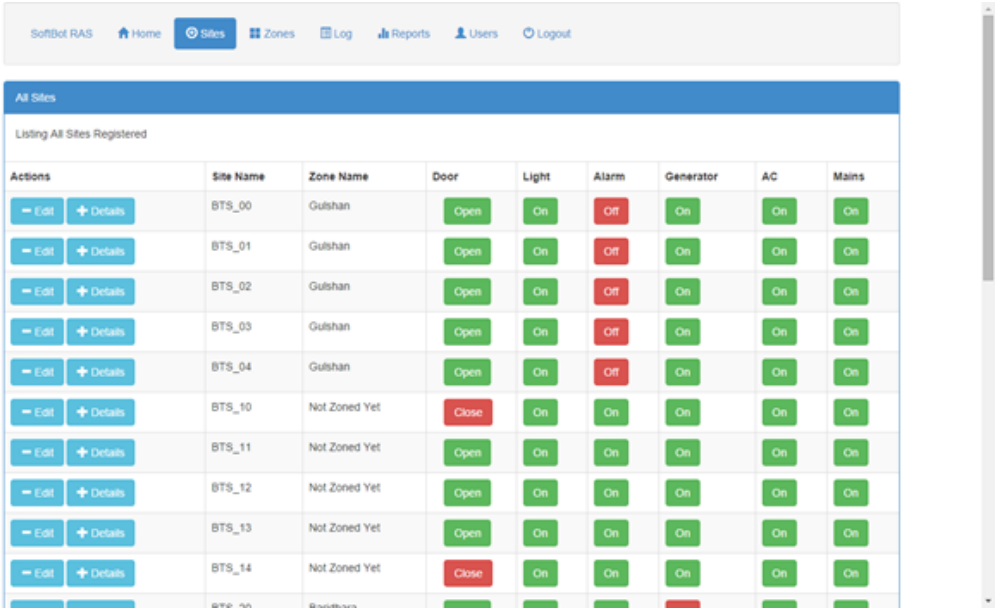

Figure 5.2.1.2 Switch Panel

Once logged in, the administrator can control individual switches via the interface shown on Figure 5.2.1.2

The administrator also has the option to grant or deny access privilege to multiple users for multiple sites or zones and is shown on Figure 5.2.1.3

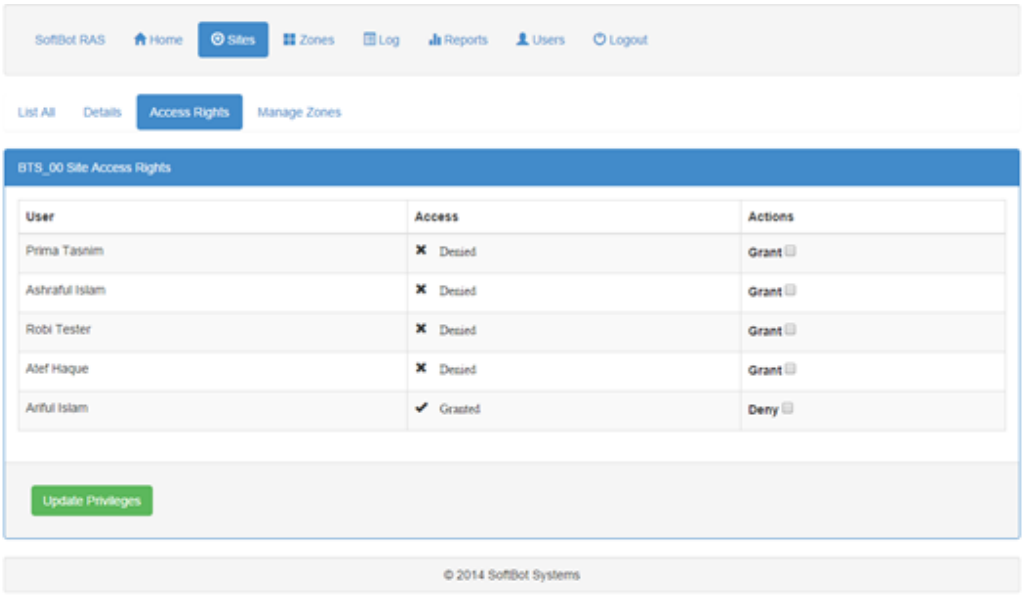

Figure 5.2.1.3 Access Rights

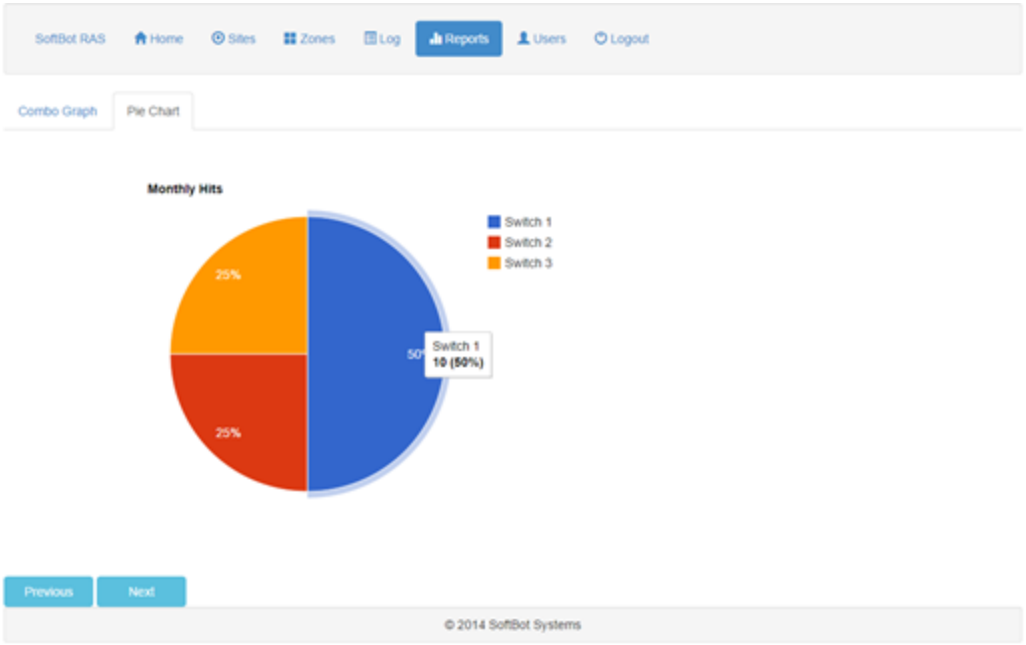

Figure 5.2.1.4 Reports Panel

The administrator also has the option to generate reports based on the individual user access, on Figure 5.2.1.4

5.2.2 Mobile Application Functionalities

The mobile application is another part of our security system. The user needs to log in to control the devices. There is a login option but no sign up option in the mobile application because only the people who are authorized by the web application can log in, otherwise a notification will be sent to the central server.

The user will login and select the assigned sites from the *site* option. After selecting a site from list, a users can control the switches, which is turning it on/off. The Sites option displays a list of sites that the user has been granted access to.

Zone, displays a list of zones that the user has been granted access to, after selecting a site from list, a users can control the switches, which is turning it on/off..

Log, displays a list of sites that the user has access to. Upon clicking on a site, all the logs generated by that site is listed for the user to view.

Map, shows the geo- location of the assigned sites.

The interface of mobile application is shown below (Figure 5.2.2.1). The user needs to log in and if the id passwords are authorized only then the user can log in and switch the switches.

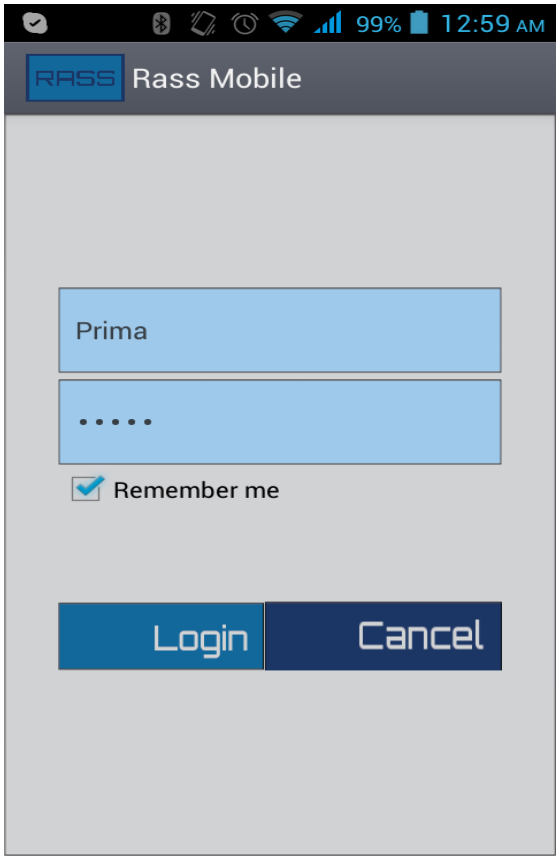

Figure 5.2.2.1 Login Activity

Figure 5.2.2.2 Home Activity

In the Figure 5.2.2.2, we can find the four options, *Sites, Log, Zones* and *Map.* The logout button will logout the user so that s/he can log in with another username and password.

In the Figure 5.2.2.3, the lists of assigned sites are shown (the list changes dynamically because assigned sites will change according to the authorization decision and those sites which are selected by the authorization for this particular user will be shown in this mobile application)

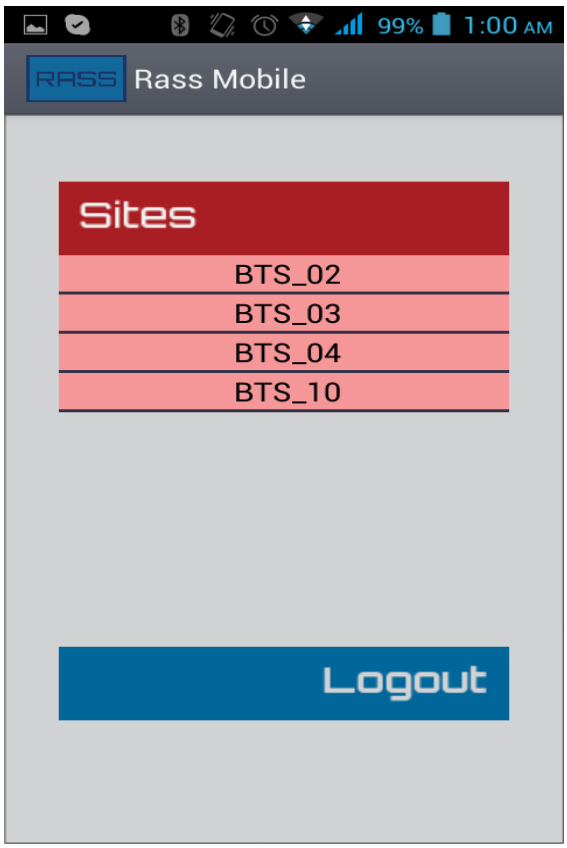

Figure 5.2.2.3 Site List Activity

Figure 5.2.2.4 Control Switches Activity

After selecting a site, in this case, i.e BTS_02 the device under this site will be shown(Figure 5.2.2.4). The user can now turn on/off the switches of any devices of this site.

Chapter 6

After having completed a successful test run of the system we ran some basic analytics to benchmark the system. Most of the analysis were based on reflex time so as to measure the latency of commands being responded with accordingly. Aside from these time related tests we also ran some electrical power consumption tests. Thermal output was measured and benchmarked accordingly to get an approximate overview of the system.

6.1 Basic Analysis

After several iterations in the process of testing the system, we have found that it responds to any command from the web within fifteen seconds at best. However, the worst scenario is somewhat alarming as it has taken more than one hundred seconds at certain times. Nevertheless, the average response time for the system is approximated at thirty seconds.

The system responds to any local commands instantly. The RFID card triggers instantaneously and has a very low probability of malfunctioning.

The power consumption is approximately fifteen watts for the entire system and it still needs much more in depth testing and analysis.

The thermal output is around three hundred and fifty joules per minute.

6.1.1 GPS Modem Analytics

The GPS Modem is somewhat problematic as it can fault and hang. However, the system can detect such a fault and reboot the GSM shield altogether in such a case. It is worth mentioning

that the probability of this failure is very low but not negligible. This happens in cases when the GPS service provider itself is busy and can not respond. This is independent of the system and in no way falls in its domain.

6.1.2 RFID Access Analytics

The RFID reader is very fast and reads a card instantly. However, rarely so, the reader takes a couple extra seconds to read depending on the status of the master unit. This happens due to a core part of the architecture of the system itself as the I2C communication protocol between master and slave units are not fast enough for the Wiegand32 RFID reader. In such a situation when the master unit is transmitting packets to the slave unit, the RFID reader will have a delay effect and the card may need to be swiped again for access.

6.2 System Redundancy

The system though providing security might be able to turn into a nuisance if it were to lock and hang. In such cases only breaking and entering to reset the system would be a viable solution. However, the system will never be able to fail as it was architected in such a manner so as to have a failsafe. Described below are the methods that were used to provide this failsafe.

6.2.1 RFID Unit

The RFID reader is connected to the slave unit, which has no chance of hanging as it is not in any way connected to the GSM shield. Thus, as long as a user with genuine access privilege is available, the system can be unlocked through the RFID card. This is vital and is the primary method of guaranteeing the failsafe.

6.2.2 Micro SD Unit

The micro SD card shield was used in order to retain user access privileges in case of a system reboot. Hence, even if no internet connectivity is available after the system automatically rebooted for any internal issue, the RFID access privileges would also be restored along with the state of the system. This architecture not only verifies the system's unconditional change but also makes sure the primary failsafe is available.

Chapter 7

No security system can be claimed to be completely invulnerable to penetration unless it is a black box itself. Similarly, RASS also has vulnerabilities and are mentioned below.

7.1 System Vulnerabilities

The system primarily has only two prime vulnerabilities and they can both be easily accounted for. The system can be made offline by spamming its ports using a **DDOS** attack or by simply disconnecting the power to the system.

Any DDOS attack can be avoided by encapsulating the whole system into an intranet by using IPs that are privately owned and have ping disabled.

An auxiliary DC power source can be introduced to the system as a source of backup power in order to avoid system shutdown due to power severance.

7.1.1 DDOS Attacks

DDOS, A Distributed Denial of Service attack is a cyber attack, it is an attempt to make an online service unavailable for a period of time by overwhelming it with traffic from multiple sources. In such cases, the attackers target a large variety of important resources, and throw a major challenge to make sure that people can publish and access important information, it can be from banks to news websites etc. Attackers build networks of infected computers, known as 'botnets', they spread malicious software through emails, websites and social media in order to

make the web server slow and to some extent to crash the entire database where the important data is kept of any web service.

7.1.2 Power Drainage

If the system is not powered, all the relays are deactivated resulting in a unlock of the electromagnetic lock, thus removing any and all measures of security. A current of one ampere is needed to keep the electromagnetic lock active. If no power is provided to the system, the lock would thus be inactive.

7.2 System Improvements

Although the system is efficient and powerful as it is designed, there still remains scope for further developments. The system suffers from major packet loss due to the nature of GPS communication. Thus it would be a great improvement if the primary mode of communication were to be changed from GPS to 3G or any other more reliable source. Also the process of synchronization with the central server is very inefficient as the number of **polls** affect the performance of the central server and will certainly downgrade performance if the number of nodes were to increase. Using RPC based communication would eliminate this inefficiency completely.

7.2.1 RPC Communication

RPC, Remote procedure call is a protocol which allows to request a service from one program to another program located in another computer in a network without understanding the internal

network details. It is also known as function call or subroutine call. RPC uses client/server model, where the requesting program is a client and the responding or service providing program is the server. It is a synchronous operation which blocks the requesting program until the result of the remote procedure are returned. The lightweight processes or threads that share the same address space allows multiple RPC calls to be performed concurrently.

Chapter 8

This is the complete system, including the device at the remote site and the central server located at the Head Quarters. This system is called **RASS** in short and is responsible for maintaining remote sites.

The system can be costly or cheap based on the scope of the solution. An analysis can be performed in order to calculate an exact costing. The system described above is in its preliminary stage. All the tools and data used to train and build the system has been discussed throughout the whole thesis report.

To make further improvements to the system a detailed understanding of networking protocols and communication protocols between master and slave using serial buses is required.

8.1 Deployment

As mentioned in the thesis, deploying the setup is made very simple through the segregation of the system into the three parts. Setting up the hardware is the only gap that needs to be filled by an electrician with a very basic understanding of electronics. Apart from this any remote unit can communicate with the central server that is already hosted at heroku for this thesis and is free of charge. The mobile application can also be found on the Google Play Store, and is also free of charge. Using the system is almost plug and play.

8.2 Limitations

The system can not be used in any location where internet connectivity in unavailable.

It can not be used in an area without electricity.

8.3 Risks

The system is not approved by any international organizations trading in the business of security according to international standards and is thus not applicable to such standards. It is only in it's development phase and much testing is needed to fine tune all the ends. Currently the system can be used by an enthusiast to test its features only.

Git Repository Links

---

https://github.com/Prima09/rass_mobile/tree/develop

https://github.com/atefth/rass_enterprise/tree/develop

https://github.com/atefth/rass_prototype/tree/develop

References

Books:

1. Power Programming with RPC By John Bloomer Publisher: O'Reilly Media, Final Release Date: February 1992,Pages: 522

2. Programming Web Services with XML-RPC By Simon St. Laurent, Joe Johnston, Edd Dumbill, Dave Winer,Publisher: O'Reilly Media,Final Release Date: June 2001,Pages: 236
3. Microsoft RPC programming guide (c1995) Author: Shirley, John; Rosenberry, Ward,Keywords: Electronic data processing -- Distributed processing; Client/server computing,Publisher: Sebastopol, CA : O'Reilly,Language: English. Book contributor: O'Reilly & Associates, Inc.,Collection: opensource

4. Microsoft RPC Programming Guide Author(s) John Shirley, Ward, Rosenberry Publisher: O'Reilly Media; 1st ed edition (March 8, 1995) Hardcover/Paperback: 245 pages eBook: Multiple Formats: PDF, ePub, Mobi, Daisy, DjVu, TXT Language: English ISBN-10: 1565920708 ISBN-13: 978-1565920705

5. Database System Concepts, *Sixth Edition,* Avi Silberschatz, Henry F. Korth, S. Sudarshan

6. FUNDAMENTALS OF Database Systems, SIXTH EDITION Ramez Elmasri, Department of Computer Science and Engineering,The University of Texas at Arlington,Shamkant B. Navathe,College of Computing,Georgia Institute of Technology

Internet:

1. http://support.balsamiq.com/customer/portal/articles/1335124
2. http://developer.android.com/index.html
3. http://www.androidhive.info/2012/01/android-json-parsing-tutorial/
4. https://developer.android.com/training/index.html
5. https://developer.android.com/training/building-content-sharing.html
6. http://www.codelearn.org/android-tutorial
7. http://xmlrpc.scripting.com/

8. http://golang.org/pkg/net/rpc/
9. http://arduino.cc/en/Guide/HomePage
10. http://arduino.cc/en/main/software
11. https://learn.sparkfun.com/tutorials?_ga=1.48314470.1511772886.1421091613
12. http://www.arduino.cc/