



DTU Compute
Department of Applied Mathematics and Computer Science

IoT based Autonomous Office Alarm and Access Control System

Master's Thesis

Luqman Perviz Akhtar (s142196)

Kongens Lyngby 2017

DTU Compute

**Department of Applied Mathematics and Computer Science
Technical University of Denmark**

Richard Petersens Plads

Building 324

2800 Kongens Lyngby, Denmark

Phone +45 4525 3031

compute@compute.dtu.dk

www.compute.dtu.dk

Summary

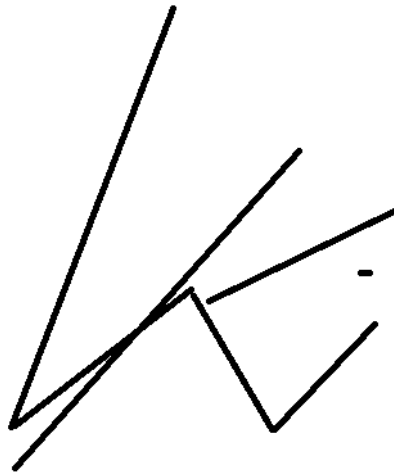
The project presents an IoT based autonomous alarm and access control system that can automatically switch on the alarm system when the last person leaves the premises. Most of the alarm systems used in the organizations are manual and human involvement is required to arm them. The manual arming of the alarm system adds an extra layer of responsibility on the user of the system and gives birth to lot of other problems that are discussed in the thesis.

However, the system proposed by us helps to overcome problems faced in the manual alarm systems. Our system is also like any other alarm system but is powered by the latest wave of computing known as Internet of Things. Our IoT based system is cost effective and provides much more functionalities than the normal alarm system. A prototype is also developed and tested in the real environment to prove the feasibility of the system.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring an M.S.c in Computer Science and Engineering.

Kongens Lyngby, September 20, 2017

A handwritten signature in black ink, consisting of several overlapping, slanted lines that form a stylized, abstract shape. The signature is located in the lower right quadrant of the page.

Luqman Perviz Akhtar (s142196)

Acknowledgements

I would like to specially thank my supervisors Prof. Christian Damsgaard Jensen and Jesper Nielsen for giving me the opportunity to work on such an innovative project. The guidance and support provided by the supervisors helped me in successfully completing the thesis project.

Beside my supervisors, I would to express my sincere gratitude and affection to my family members for their invaluable support throughout my educational career.

Contents

Summary	i
Preface	iii
Acknowledgements	v
Contents	vii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Definition	2
1.3 Project Goals	3
1.4 Project Scope	3
1.5 Thesis Outline	4
2 Background	7
2.1 Electronic Access Control System	7
2.2 Electronic Security Alarm System	12
2.3 Distributed Computing	15
2.4 Third wave of computing	18
2.5 Internet of Things	19
3 Analysis	27
3.1 Problem	27
3.2 Decomposition of Problem	28
3.3 Requirements	46
4 Design	49
4.1 Hardware	50
4.2 Software	60
5 Implementation	71
5.1 Software Tools	71
5.2 Interfacing Electric Lock with Controller Board	71
5.3 Interfacing Reader with Controller Board	73

5.4	Interfacing PIR Sensor with Controller Board	74
5.5	Interfacing Buzzer with Controller Board	76
5.6	Interfacing LED with Controller	76
5.7	Connectivity	76
5.8	Central Server	78
5.9	Mobile Application	84
6	Evaluation	89
6.1	Autonomous Alarm System	89
6.2	Access Control System	92
6.3	Firmware	95
6.4	Central Server	100
6.5	Mobile Application	102
7	Conclusion	105
7.1	Future Work	105
A	An Appendix	107
A.1	Server Code	107
A.2	Mobile Application Code	121
A.3	Firmware	128
	Bibliography	145

CHAPTER 1

Introduction

The third biggest wave of information technology (IT) is dominating everywhere at a rapid pace by connecting the world in a completely new way. This revolutionary wave is known as IoT, which is the phenomenon of connecting objects containing embedded system to the internet so they can speak with each other.

According to Cisco it is projected that by 2020 there will be fifty (50) billions physical devices connected to the internet [1]. Rapid decrease in price and size of electronic devices over the years has made it possible to connect related devices together over the internet. These devices when connected together can share, analyse and process valuable information thus forming an intelligent system of systems. IoT make Things (devices/objects) more efficient than they really are and open up a complete new way of interaction with the world.

The figures [2] clearly depict that the market potential of IoT is huge and will grow to 3.7 billion dollars by 2020 as it is being deployed in diverse range of applications that mainly includes health care, Industry 4.0, Automotive and Smart Homes/Buildings. In the near future, IoT will make every aspect of our life automated, convenient and intelligent. In this project the power of IoT will be combined with two of the important components of Building Automation i.e Alarm and Access Control System. This will give us insight of how, using an IoT technology, a simple system can be turned into an intelligent one that can provide much more features than the original system.

1.1 Motivation

The first and foremost important factor which inspired me to do this project was my own personal interest in the field of IoT and Automation. Secondly, was the desire to build a system that can help to solve real world problems being faced on regular basis by the users of the system.

It didn't take long to discover the problem domain as it is common and can be seen in every other organization. The problem was the need to switch on or off the alarm system of the buildings. I was surprised to see that still lot of medium to large scale organizations have to do it manually. This job is being done either by the security personal of the company or the last employee leaving the company. I myself have encountered such a problem at my workplace and you can imagine how difficult

it is to search each and every corner of the organization to see if everyone has left the workplace or not before activating the alarm.

The strong need to build the autonomous alarm and access control system was truly realized when I met Jesper Nielsen the former CTO of AdeoOS Aps Company. There he told me that how often they had to face such a problem on daily basis. His business involves providing private and shared office rooms to the companies which means there are plenty of different companies working under one building. He has to ensure that each of the office alarm is activated and it is difficult to do so because every company has its own office timings. He also highlighted the issue of employees forgetting to checkout and turning alarm on while leaving. He is aware of the fact that we humans tend to make mistakes and such kind of scenarios are likely to happen. So we can ignore the human error but cannot afford the cost associated in case of burglary.

He told me that advance access control and alarm systems out in the market are way more expensive to afford and still they dont mark up to the expectations of the user thats why they are still facing such a problem. He has the vision of solving the above mentioned problem by building a cost effective autonomous alarm and access control system that is user friendly, flexible, reliable and scalable. As we both were on the same track so he provided the opportunity to supervise me and allowed me to carry out the project in his company.

Last but not least the idea was discussed with the professor of DTU Christian Damsgaard Jensen who has done tremendous amount of work in this field. He liked the idea and agreed to supervise me in the project. To sum up, I would like to say that having a perfect team of supervisors on board and keeping in view the challenges mentioned above motivates me well to build an autonomous alarm and access control system that can create value for the users.

1.2 Problem Definition

The project originates from the problem to have the office buildings, automatically switch on the alarm when the last person leaves the premises. Currently, in most of the organizations the last employee leaving the building after work needs to activate the alarm manually. For companies that are big in size it often becomes difficult for the last employee to search the whole building and to see if he exactly is the last employee leaving the building. The worst case happens when the last employee turns on the alarm and actually he is not the last one. So whoever checkout afterwards will result in triggering the alarm. This situation might end up calling police or security companies that might charge way more for such kind of scenarios.

The alarm and access control system installed in the building of AdeoOS Aps works similar like any other security systems in the market. The problems faced with the currently installed alarm system are as following.

- User forgets to set the alarm
- User thinks another person is in the building
- User sets the alarm when another person is in the building (that triggers the alarm)
- Semi-Autonomous
- High Cost
- No connectivity to the internet
- Limited memory size of hardware to store user credentials

The problems mentioned above clearly depict that most of the alarm and access systems installed in the companies are not user friendly and are not convenient to use for most of the users. A novel smart and autonomous alarm and access control system must be designed to provide optimal solution.

1.3 Project Goals

The main goal of the project is to design and implement cost effective IoT based autonomous alarm and access control system prototype that can overcome limitations faced in the current system. The designed prototype will be capable of collecting room data through installed sensors and send it to the server. The server will serve as a commander that will store and evaluate data to take decisions and issue commands to operate alarm and access control system autonomously. The prototype will minimize the percentage of false alarms, burglary, and human error by providing implicit interaction.

1.4 Project Scope

As the project is quite vast consisting of many disciplines thereby not all the parts of the system would be touched due to the man power and time limitations. The work is mainly focused on the highly prioritized requirements set by the company. Therefore, the scope of the project is to implement access control system and automate the process of setting the alarm by

1. Designing and developing an autonomous alarm and access field control system using
 - Micro-controller
 - Access control components

- Alarm system components
 - PIR sensors already installed, to detect any presence in a room
 - Compatible network device to make it an IoT solution
2. Developing an alarm and access field controller firmware, which will help to interface the hardware components of both systems together.
 3. Developing main control software on the server that will be able to
 - Set an alarm in a specific room or for the whole building
 - Control the states of the alarm (Unarmed, Warning and Armed)
 - Lock and Unlock the door
 - Monitor the state of the door (Open or Close)
 - Send and receive instructions from the Mobile App

It the end, an overall developed prototype will be tested by installing it at the AdeoOS office to prove the feasibility of the system. This will in turn help the company to proceed into the production and business phase.

1.5 Thesis Outline

The thesis consists of seven chapters in total and remaining thesis is structured in the following order.

Chapter 2 provides necessary background knowledge related to our problem domain. The chapter starts by introducing the alarm and access control system along with their components. Afterwards, the chapter throws light on the concepts of distributed and ubiquitous computing. The relevant architectures, advantages and challenges present in these fields are also discussed. The main aim of this chapter is to help reader get familiar with the project domain.

Chapter 3 analyse our problem by breaking it down into the sub problems. Afterwards, each sub problem is analysed and possible solutions that can cope with problem are presented. In the end, a list requirements is created which will help us to design a system that can overcome problems faced in current system.

Chapter 4 presents the overall proposed architecture satisfying the requirements listed in the analysis section. Detail explanation of each components and their connection to each other is also discussed in this chapter.

Chapter 5 shows the implementation details of the proposed architecture. The implementation chapter leads to the development of prototype as a proof of concept.

Chapter 6 shows the evaluation of the prototype against the requirements set in chapter 3. The evaluation will help us to prove the feasibility of the system. Few test results that helped to validate the prototype are also shown in this chapter.

Chapter 7 finally concludes the thesis work and some of the ideas that are not implemented in the current project but have been left for the future work are also suggested in this chapter.

CHAPTER 2

Background

The concept of access control is as old as the humanity itself. As humans started to live in communities, there raised a need to protect themselves from external threats and they started to build big walls around their dwellings or dig up artificial lakes around. This in fact was the access control system in its simplest form, its main purpose was to protect their possessions by keeping the threats away.

Over 4000 years ago, ancient Egyptians first introduced simple keys and locks as an access control mechanism to safeguard their personal valuables [3]. Since then we have seen considerable amount of improvement done in the keys and locks technology. Over the years, this technology has gotten quite mature and is being widely as an access control mechanism in the entire world. On the other hand, as the security started becoming more and more vital it became clear that we cannot solely rely on this technology as it carries lot of flaws with it. The problem with the keys is that it can be duplicated, stolen and misplaced which means afterwards it becomes necessary to replace all the locks associated with those keys. As far as the locks are concerned they are mostly exposed to the outside world and can be picked by the intruder easily. There is no practical way of determining if someone has been trying to tamper with the lock in your absence. Additionally, the locks doesn't provide the ability to control when and by whom the keys should be used.

In order to overcome issues associated with the keys and locks technology manual systems were introduced. Manual system basically included watchmen now known as security guard as an elementary form of access control system. On one hand this method provided some solutions for the problems associated with the key and lock technology, but on the other hand it introduced new complications. As mentioned earlier, we humans tend to make mistakes so it involved the possibility of human error and not to mention skills of guards are sometimes questionable.

2.1 Electronic Access Control System

To overcome the drawbacks of the above mentioned system it became clear that an improvised and advanced system was need of the hour. Thereby leading to the era of electronic access control systems. These systems use the power of computing to gain control over determining who goes where and when.

An electronic access control system is a combination of hardware and software components that are used to control access at the entry points of the premises. The electronic access control system replaces the traditional keys with the wide range of user credentials which can be used to verify legitimate users to grant access. Whereas, the locks are replaced with the electrical ones which can operate by automating the mechanical behaviour of locking and unlocking. Overall, electronic access control system provides much better and faster access to the authorized personnel as compared to the key and lock technology.

Apart from controlling access, the system can also log transactions and provide detailed reporting of recorded transactions for each entry points. This was of course one of the major limitations of the lock and key technology.

2.1.1 Components of Electronic Access Control System

In this section we are going to look into the essential components of an access control system.

2.1.1.1 Controller

Controller is the heart of the access control system. It is also known by the names of access field controller or intelligent controller. All of the access control peripherals are interfaced to the main controller.

Controller is the one that is responsible for receiving inputs, processing data and producing outputs. It acts as a communication gateway between access control software and access points . Controllers have the capability to store the user information and system configuration in its internal memory but it is normally not recommended as the memory is quite low as compared to the computers. Therefore, controller mainly connects to the software on PC or server to send or retrieve the information of user credentials and control configuration.

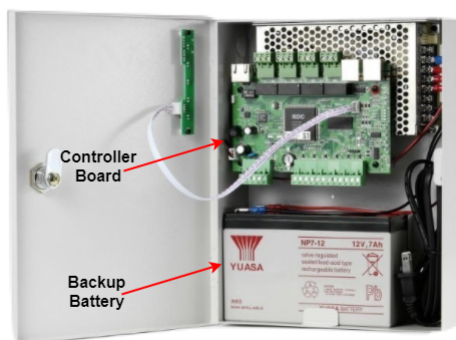


Figure 2.1: Access Controller box consisting of controller board and power unit [4].

Lastly, it also contain a power unit that is responsible for powering up the controller and the connected peripherals. Sometimes an additional backup battery is also attached with the controller to keep the access control system running in case of power failure.

2.1.1.2 Locking Device

Locking device is basically an electric hardware lock which is used to control the access points based on the instructions from the controller. These access points are basically doors, gates or any other types of physical barriers. There is a huge variety of access control locking devices being used in the world. Some of the popular examples are magnetic, door strikes and mortise locks. The access control locking devices comes up with two typical configurations

- **Fail-safe** The locking device will get unlocked when power is removed.
- **Fail-secure** The locking device will get locked when power is removed.



Figure 2.2: Door fitted with electronic lock [5]

Generally, fail-safe configuration is preferred over fail-secure configuration, so as to prevent locking up of anyone in the building in case of emergency.

2.1.1.3 Access Control Software

Access control software is known as brain of the access control system. The access control software is a piece of computer program which is used to manage access privileges, controller configurations, user credentials and record transactions. It receives data from the access point controllers and based on the system configuration and data received it makes decision whether to grant or deny access. It can take counter measures against detected suspicious activities and can instantly inform the person

in charge of a system about the critical condition. The access control software can be installed on a computer, dedicated servers or on cloud.

2.1.1.4 Reader

Reader is an electronic device which is used to read credentials provided by the user. The readers are installed near the access points. The readers are not responsible for making an access control decision but sends the credentials provided by the user to the controller which verifies it against the information present in the database. There are basically two types of reader available in the market which are

2.1.1.5 Physical Contact Reader

Physical contact reader is a type of reader which has to be touched in some form to interact with it. Popular examples are magnetic strip (obsolete) or contact chip readers.

2.1.1.6 Contactless Reader

Contactless reader is a type of reader with which we can interact without having any physical contact but within some defined range. Examples are proximity, long range and some forms of biometric readers.



Figure 2.3: Two in one Contact and Contactless Reader [6].

Sometimes mix combination of contact and contactless reader along with the numeric keypad is used to achieve multi factor authentication.

2.1.2 Access Credentials

Access Credential is the identity information of the user that is used to get an access to the resource. These credentials could be in the form of

- Physical entities like key fobs, magnetic strip or smart cards.
- Knowledge based like password, PIN etc.

- Unique Characteristic or behaviour like biometrics.

These are three categories of credentials that can be used either as single or in combination for authentication. Although authentication based on single category of credential is highly deprecated and it is therefore suggested by security researches to verify credentials from multiple categories.

2.1.3 Access Control System Operation

The main operation of the system starts when the user presents his credentials to the reader in order to verify himself to get access through the access point of the premises. The user credentials are then sent to the controller by the reader. The controller compares the credentials with the one that are already stored in the database. If the information is matched, the controller will send the signal to the access point to unlock itself to grant an access to the user. The whole event will also be logged by the controller in the database along with the user provided credential information.

If in case the provided credentials are wrong, the controller will only log the event but will not send any signal to the access point so that it can keep on maintaining its locked state. The controller also sends an additional signal on the readers side to provide an audio or visual feedback to the user about the access state.

In most of the cases an exit is uncontrolled which means REX buttons are normally used to unlock the access point from inside. However, if someone wants to control exit as well, then an additional reader would be required on the exit point. This means user will have to go through the verification process again to exit the premises.

2.1.4 Advantages of Electric Access Control System

The electronic access control systems have tremendous amount of advantages over the key and lock technology. Some of major advantages are highlighted below

- **User Friendly**

Quick and easy to check in and access premises.

- **Remotes Access**

Provides the ability to control the system remotely by using computer or mobile phone.

- **Customizable**

It provides the ability to personalize access rights and revoke credentials rights anytime so no need to change locks in case of stolen or lost credentials.

- **Accountability**

The ability to provide detailed audit report which can help the owner to see who went where or who tried to access unauthorized premises.

2.2 Electronic Security Alarm System

Electronic security alarm system is an electrical device that is used to detect intruder or unauthorized entry into the premises and alert the surrounding people or/and the owner of the system [7].

The security alarm system is often used in combination with the access control system to enhance the level of security. Where access control system is used to restrict the access, the security alarm system is used to detect the unauthorized access in case if someone manages to breach into the premises. The security alarm systems have proven to be helpful in deterring and capturing the burglars.

2.2.1 Components of an Electronic Alarm System

The most important and popular components of a security alarm system are as following

2.2.1.1 The Control Panel

This component of alarm system is similar to the access controller component of an access control system. It is known as the brain of an alarm system. The panel comprises of the controller, power unit and an additional backup battery.

All other components of an alarm system are connected to the controller. It serves as the central processing unit and all the decisions are taken by the controller based on the data received from the connected components. Plus, it also has the responsibility to alert the law enforcement or alarm monitoring companies if in case an intruder is detected.

2.2.1.2 Keypad

The keypad is basically a simple pad consisting of buttons that may represents digits, symbols or sometimes alphabets [8].



Figure 2.4: Alarm Control panel with keypad [9].

In alarm systems the keypad is used to provide external inputs to the control panel. The inputs normally comprises of arming and disarming the alarm. It is also

used to configure the settings of the control panel. Normally, alarm panels are placed close to the secure side of the access points so that it is easier for the owner of the system to arm or disarm the alarm while leaving or entering the premises.

2.2.1.3 Sounding Device

It is an output device of an electronic alarm system. This device could be any audible equipment that can make loud sound in order to alert the neighbours and to startle away the intruder from the scene. Examples are siren and horns.



Figure 2.5: Alarm Siren [10]

2.2.1.4 Sensors

Sensors are one of the core components of security alarm system. Sensors are the one that are used to detect the suspicious activity in the premises. Sensors can monitor the door or window opening, glass breakage, movements, noise and many more similar things to detect the intruder. As soon as the intruder is detected the information is sent to the control panel which initiates the process of sounding the alarm and alerting the alarm monitoring or security services. Some of the widely used sensor are motion sensor, door sensor, glass break sensor and many more.



Figure 2.6: A magnetic contact sensor [11].

2.2.2 Electronic Security Alarm System Operation

The operation of an electronic alarm system starts once the alarm system has been installed in the premises and the zone have been secured by means of detection sensors. For the sake of explanation let us consider a fictitious room which has one window and door opening. The room has magnetic sensors installed for window and door open detection and one motion sensor installed to secure the premises.

Once the system is armed and running, the sensors will start doing their jobs. If the intruder tries to get into the room by opening or breaking the door, the sensor contact between the door and the frame will be broken which will indicate that the intruder has managed to get into the room. This event is going to be sensed by the controller which will then take counter measures like activation of alarm, calling the emergency services and sounding the alarm to deter or capture the intruder.

Another case could be if intruder manages to dodge the door and windows sensor, for example entering the room by making hole into the wall. In that case a motion detector sensor would be able to detect the movement of the intruder inside the room and controller will trigger similar action as described above. This is how an alarm system operates generally but of course can have more operations and security by based on the components attached to the system.

2.2.3 Advantages of an Electronic Security Alarm System

The alarm system is an important security device that is used to protect you and your investment against the intruders. Apart from that it also provides numerous other benefits which are as following.

- **User Friendly**

Easy to turn on or off by simply pressing few buttons. Nowadays, designed in a way that is easier for user to install around their properties.

- **Protection of Valuables**

The alarm system can help to protect you and your personal goods by alerting the home member and the law enforcement agencies which can arrive at the scene instantly.

- **Remote Surveillance and Control**

If combined with the CCTV camera can provide one with the live coverage of the scene from any location of the world. Plus can provide the ability to remotely arm or disarm the alarm.

- **Insurance Discount**

Most of the insurance companies will give discount to the customers that have alarm systems installed at their properties.

- **Deter Burglar**

One study showed that about 83 percent of the criminals will try to look whether an alarm system is installed or not before entering the target place. About 60 percent of the criminals said that they would prefer an alternative target if an alarm system is present [12]. The figures clearly depict that alarm systems demotivates the criminal to attempt burglary in a secured zone.

2.3 Distributed Computing

As our project involves combination of hardware and software components which means computations are going to be distributed across multiple components at different levels. Therefore, it is important to get a general overview of the field of distributed computing.

Distributed System is a collection of independent computers that are working together to achieve common goal and appear to the user as one computer system [13, 89]. In simpler terms a type of system where processes are not only present on one but multiple machines and communicate with each other by passing messages.[14].

2.3.1 Goals of Distributed Systems

- **Resource Sharing**

Simple and easy to access and share remote resources among systems. The resource could be anything like shared file server, printer, web-page etc. Simple example is shared file server where the file present at one server can be accessed from any computer we log in.

- **Scalability**

The ability of system to grow and manage without compromising on performance in case if there is an increase in

1. Number of resources and users.
2. Distance between computers.

- **Openness**

Offering services according to specific set of defined standards so that

1. Different implementation can interact with each other (Interoperability)

2. New services can be added and removed easily (extensibility)

- **Transparency**

The system should appear to the user as a single unified system rather than showing in terms of collection of resource and processes distributed across multiple computer. Examples are transparency in terms of access, location, failure etc. [77].

2.3.2 Distributed System Architecture

There are two main types of architecture in distributed systems.

2.3.2.1 Client – Server Architecture

The client server architecture splits the processes into two groups

Server is a process that listens to the request and provides service according to the received request. The provided service can be data from database, file, printing service etc.

Client is the one which sends the request to the server for service and waits for the response.

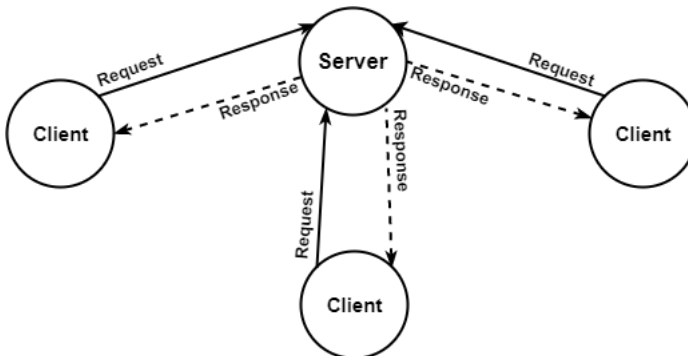


Figure 2.7: Client-Server Model

When the client needs the service it sends the request to the server known as invocation. Afterwards, waits for the response from the server and on the server side as it receives the request it responds back with the particular service to complete the interaction. This interaction as a whole is known as remote invocation. Servers can serve multiple clients and are not interested in how the information is going to be used [15].

The example of client server can be seen in interaction between web browser and web server. The web browsers (like chrome) act as a client which is used to invoke web server. Lets suppose that the browser client sends the request to the Wikipedia server. The server will reply with the HTML code which the browser is going to render and display the document in readable format.

Server can serve multiple clients. Server and client can be presented on one or multiple machines [16]. Additionally, server can also act as a client if it invokes other server. The terminology of client and server is nothing more than a role of a process. The drawback of the client server architecture is that there is single point of failure which means if the server fails than the whole system fails.

2.3.2.2 Peer to Peer Architecture

The peer-to-peer model is also known as decentralized model because this model does not contain any server as a central system [17]. In peer-to-peer model, all nodes act as both server and client. This means each node is a service provider as well as requester so everyone can share resources with each other directly. In P2P system, if a particular node fails the overall systems is not affected [18]. Bit Torrent is a classic example of this, if client realize that the peer is not responding, it will try to search and connect another peer and will start continue to download its file where it currently was without interruption.

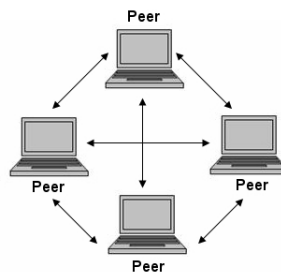


Figure 2.8: Peer to Peer Model [19].

The drawback of peer to peer model is that it is difficult to maintain security and to manage administrative control of the system.

2.3.3 Advantages of Distributed Systems

Distributed systems provides us with lot advantages as compared to single system. Some of the advantage are mentioned below.

Economic

At much lower cost collection of computers can reach the same power level as that of the expensive supercomputers.

Performance

It can help to overcome the computational load by spreading the labour across multiple computers. Each computer will be able to perform partial task efficiently and faster. Thus increasing the overall performance entire system.

Incremental Growth

The systems can be extended according to the requirements of user. For example, if a system has two machines working on a particular task and the task is complex that cannot be handled by these two machine you can just simply add more machines according to the needs.

Reliability

Multiple collection of systems provide much better fault tolerance and allows the machine to fail without effecting the overall operation of the system. Of courses, the performance will be reduced. Not to forget that if components of the distributed system are centralized a single point of failure might occur causing halt to the overall system.

2.4 Third wave of computing

The first wave of computing started roughly about 30 to 40 years ago in the form of mainframe computers. The computers at that time were bulky and occupied the entire room. Also being expensive to operate, each mainframe computer was shared among many trained professionals [20]. Afterwards, came the wave of computing when PCs were introduced where the interaction was one on one. The dramatic shift from first to second wave of computing made technologist realized the emergence of third wave of computing [78]. So, Mark Weiser from Xerox Palo Alto Research Center and IBM proposed the terms ubiquitous and pervasive computing respectively as the third wave of Computing [85].

The basic idea behind these two technological phrases is about embedding computing everywhere and anywhere in the environment in such a way that it becomes virtually invisible or unnoticeable to the user [21, 92]. This technology has introduced new patterns of interaction i.e., one to many or many to many. Ubiquitous computing will allow the users to interact with the computers in much more natural or implicit manner [22, 23]. The researchers believe that reduction in price, size and increment in processing power and storage along with the robust networking of computers will enable us to integrate them in every day object of our life whether its chair, clothes, cars or human body[24] .

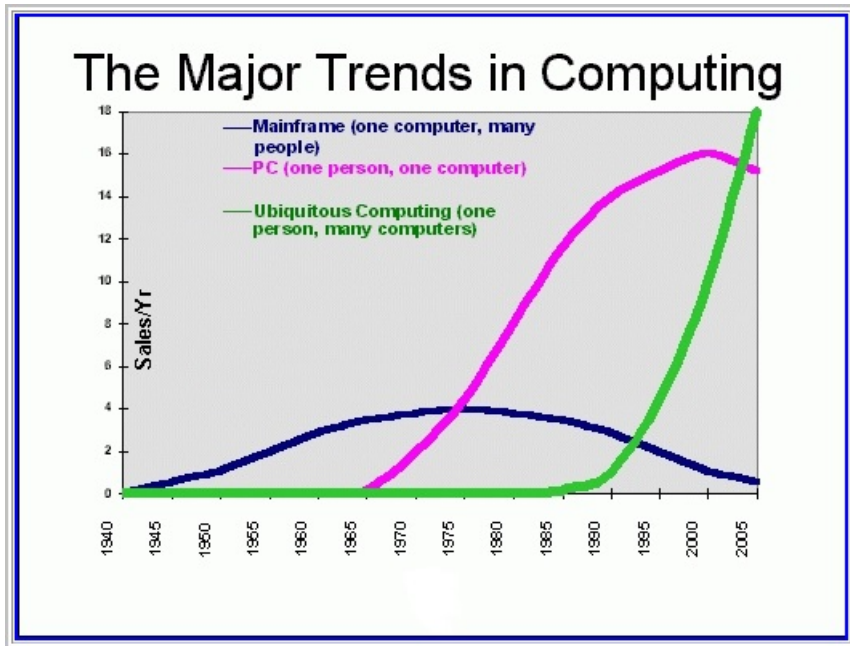


Figure 2.9: Graph showing major trends in computing [20]

This concept has led to several other related research areas like ambient intelligence, context-aware systems, mobile computing, IoT and many more. One thing to note down is that all of these contribute to the same vision. We will look into the field that is particular of interest and related to the scope of the project.

2.5 Internet of Things

The future on Information Technology known as IoT is the concept of creating a web of objects that are connected together over the internet [94]. The term IoT was introduced by Kevin Ashton in his presentation about RFID at Procter and Gamble in the year 1999 [73]. The term *Thing* in an IoT Technology refers to any physical object that incorporates embedded technology and network connectivity [25].

IoT Technology provides the *Things* with the ability to speak so that they can share the valuable information collected within their surroundings to the internet where it can be analyzed and processed so that it can be used to make better and smarter decisions. These physical devices consist of combination of one or more electronic components like actuators, sensors, processors, transmitters and receivers that are working together to link the physical world with the digital world [84]. The example of one such device is an IoT refrigerator which can do several things like

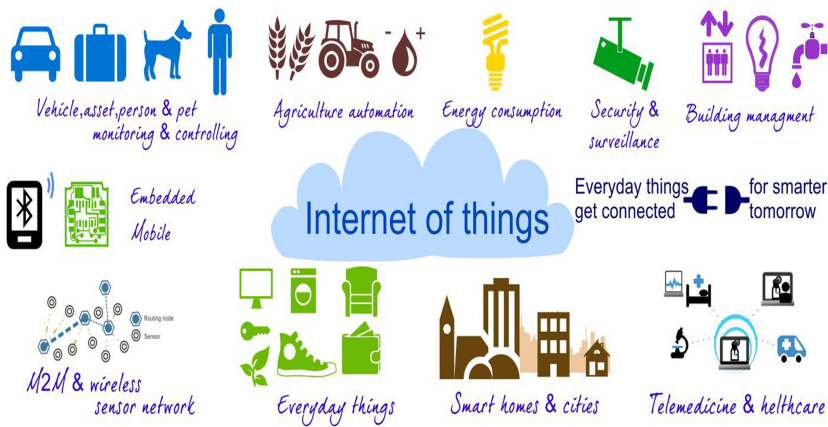


Figure 2.10: Various fields of Internet of Things

tracking our usage habits, counting number of products left in the fridge, adjusting temperature efficiently to save power, ordering products autonomously and many more similar things.

According to one of the report from Intel there are still 85 percent devices present in the entire world that needs to be connected [26]. There is a famous saying by Jacob Morgan in his article about IoT technology [27] on Forbes website that

“Anything that can be connected, will be connected”

This statement is not far from reality because the components of IoT technology are getting cheaper, portable and faster due to which it becomes easier to integrate anything in everything with minimal possible cost. Moreover, we are already blessed with the tools that can add sensing and communicating capabilities to almost all of the things in the world. The IoT technology is providing us with new types of advance innovative application and services that can help us to improve every aspect of our life whether its health care, business, energy management, education, transportation or industry etc. After several years of progress the IoT Technology has finally reached its take off stage and is going to be a boon to mankind in the coming future.

2.5.1 Applications of IoT

In order to realize the true values created by IoT we need to look at some of the uses cases and application where IoT technology id in operation or currently being developed. Some of the popular examples are given below.

2.5.1.1 Health Care

This is one of the important field where everyone is working on to create solutions which can be used to monitor disease and predict early warning signs to save the patient life from the deadly consequences of the disease on time. One of company known as Rest Devices [28] has actually made a device embedded in baby pajamas which is capable of detecting *Sudden Infant Death Syndrome*. If respiratory rate of the baby reaches critical stage the device will immediately text or call the parents and emergency service.

Similarly, IoT based insulin pumps are available that upon usage send the information remotely to the doctor without having the patient to bother about paying a visit to him. Additionally, it also reminds the patient about the time and amount of dosage intake so that not even a single dose is missed [29]. Full time patient monitoring systems will help doctors understand disease better to reduce misdiagnose rate and increase life expectancy. IoT is making efficient and better health care system by revolutionizing the industry with smart innovative health care application.

2.5.1.2 Smart Homes and Buildings

Smart Home and buildings is one the most popular [30] and successful field of IoT Technology. It is the concept of connecting various appliances in our home and buildings with each other so they can monitor and manage our place efficiently and provide the owner with ability to control these devices remotely. One of the popular example is the latest IoT device by Google known as Nest thermostat [31]. This device is linked with your heating system, fans or air conditioners and then based on your needs adjust the temperature of your home. It can also sense you and if you are not at home, it can turn off the appliances attached to it automatically to save power and cost.

The lights, washing machines, refrigerators, windows, doors, garage and all other appliances are going to become intelligent which will be able to manage and work according to our needs. The lights will turn on and off automatically, washing machine will be able to tell you once it has done cleaning and turn off itself, your refrigerators are going to update about the stock in the fridge and your doors and windows will be able to lock and open autonomously. Your garage will know when you are about to come and will start the car and open the gate for you. In short, our entire home will become autonomous.

The smart buildings will be able to automatically detect authorize residents and will provide smart access to them. It will be able to detect the air quality in the surroundings and will try to improve in a best possible way with its intelligence. It will be able to tell about its physical condition so proper reparation can take place. Will provide assistance to the emergency services for example in case fire is set, it will inform the related services immediately about the critical situation and will help firefighters by providing necessary details like the origin of fire, number and location

of people in the building. Apart from that, it can track occupancy within the building and can help people to find the free spots immediately.

Smart homes and buildings will help in optimizing resource and energy consumption in an efficient manner, which will in turn save cost and contribute towards a green environment.

2.5.1.3 Smart Cities

The IoT technology is not limited to smart home and buildings but will also transform our city into smarter city. Interesting focus area of smart cities are traffic monitoring, street and traffic lights, air quality, pollution and waste management, highways and parking systems. The traffic management system that can provide important updates about routes like traffic situation, roadblocks and accidents to your car which afterwards can plan alternative route. The IoT parking lights will be able to tell us about unoccupied parking spaces within the city so we do not waste much of our time on searching free spot. The smart lights will not be limited to providing illumination only but will also be able to detect air quality and monitor people and vehicle around their surroundings. This information will help in making a sustainable environment. The waste containers will be able to provide information whether it is ready for collection or not so that we can save our fuel, time and money associated with waste collection.

Bigger companies like IBM, Cisco, Google, Siemens and many more have started providing such solutions and countries have started making projects and budgets that are going to adopt these solutions. The real world example of IoT in smart city can be seen in the London city where a traffic management system with the name SCOOT [32] is present. This system is capable of adjusting the green traffic light time by getting the feedback from the road traffic. Thus helping in saving fuel, cost, time and traffic congestion.

2.5.1.4 Smart Industry

Internet of Things is going to help in overcoming errors that affect the outcome of the production process. For example, it will be able to track the performance of the machines by looking certain parameters like accuracy, downtime and health of the machine. It will help to eliminate downtime and provide smooth production flow, thus increase in output. IoT enable machines to schedule the resources according to the needs of the product. Moreover, IoT technology will also track the performance of labour force in the factory. One such example can be seen in the company of Denmark known as Nemlig where labour performance is tracked to see if labour is working sincerely or not. Three different coloured lights namely green, yellow and red are installed which represent the labour performance. The green lights represent that the employee is working at the required pace whereas red light shows that the employee is working below expectations and needs to speed up. They track this by counting the number of order processed in a specified time. IoT will help to identify

and overcome the holes in the production process, which in turn will help to optimize operation and cost to facilitate the growth of the business and the Industry.

Other examples of IoT Technology includes automotive, agriculture, media, payments etc. In short, we can say that IoT has created its place in every field of technology.

2.5.2 Architecture of IoT

There is no standard or single architecture for IoT. Over the years, different architectures have been proposed but we will look at one of the basic architecture, which is also a base in every other proposed architecture. The basic architecture depends one 3 to 5 layers [80, 86].

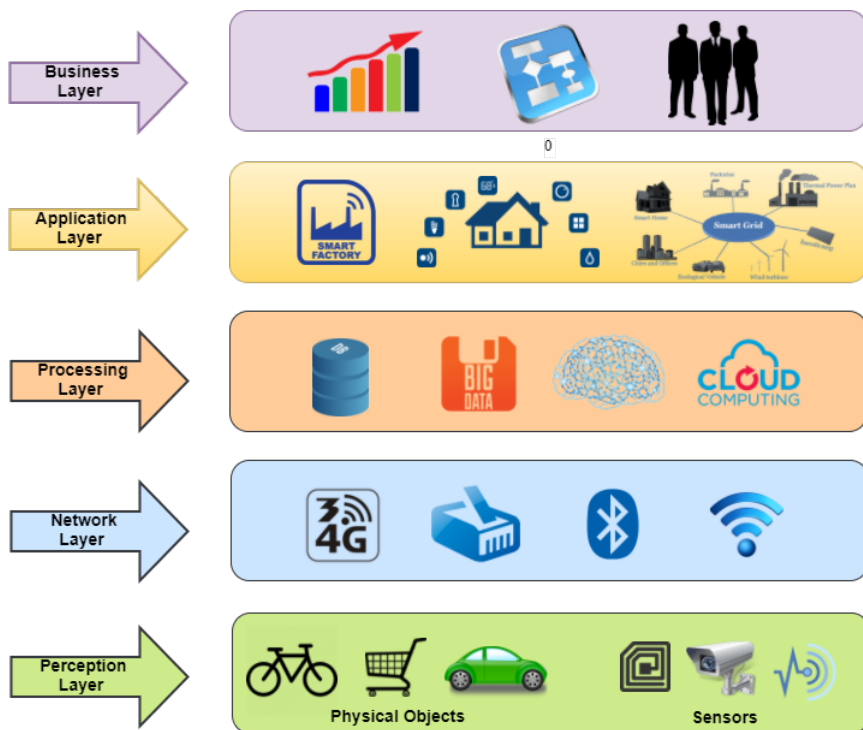


Figure 2.11: IoT Architecture

- **Perception Layer**

This is the lowest layer of IoT architecture it consists of Things like physical objects and sensors. Two things are done at this stage, one is identification of physical objects based on the sensors attached to it and other is to gather

information about the object or their surroundings [72, 86, 91]. In short, we can say that the layer is responsible for connecting the physical world with the digital world.

- **Network Layer**

It is the second layer of IoT architecture also known as Transmission Layer. It works as a bridge in between perception layer and the processing layer. The network layer is responsible for sending the data received from various sources to the processing layer and vice versa [86]. Both wired and wireless technologies like Bluetooth, Wi-Fi, GPRS, 3 or 4 G, Ethernet, ZigBee etc. can be used for transmitting and receiving data [81].

- **Processing Layer**

As the name suggest this layer is responsible for processing the data of perception layer received through the network layer. It is also known by the name of middleware layer. In this layer the data is analysed, processed and stored using intelligent technologies like big data, cloud computing, database management and Artificial Intelligence [84].

- **Application Layer**

The application layer is responsible for providing intelligent services to various fields and users according to their needs [83]. For example, if the IoT technology is used in the vehicle it will give information necessary for the driver. This layer also helps in determining the fields where IoT applications can be used for example Smart Homes, Factories, health agriculture and many more [84, 86]. Application layer interface can be used by the users to access IoT device [88].

- **Business Layer**

The business layer is about managing the IoT systems. This layer helps to generate profits from the applications and services provided to the market. It also deals with the user privacy. It analyses data received from the previous layer to carry out research that will help to create future innovative products and business opportunities[84, 93].

2.5.3 Security and Privacy Challenge in IoT

The security is one of the hot topic and current challenges of IoT technology. Since large amount of data is collected and shared over the internet it raises questions about the confidentiality, integrity, authenticity, authorization, availability of data and privacy of the users. The issues related to privacy and security of the data needs to be addressed and solved if we want to see the adaptation and development of this technology in near future. First, let us see what these terms really are

- **Privacy**

Privacy assures appropriately gathering, using, protecting and destroying data [33].

- **Security**

Security is about maintaining CIA of information [33].

In order to realize the consequences of compromising privacy and security let us consider a scenario. Suppose that a person is a heart patient and uses IoT pacemaker connected to the internet directly. Now imagine what will happen if someone gets access to the information provided by the IoT device probably he can play around with that information. Moreover if he gets access to the patients device then he can turn it off which will lead to sudden death of the patient.

A real world example of hacking IoT device was also demonstrated live at Infosecuirty Europe 2015 Conference where the Team of PenTest Company hacked the IoT kettle to get the password of the Wi-Fi network [34]. Getting password means an open gateway to all the other devices connected on the network. They connected the kettle machine with their own unencrypted access point and then dig into the memory of the embedded device to extract password in plain text. They also highlighted the issue about laziness of the people where they dont tend to change the IoT device default password.

Researches have highlighted the security and privacy challenges faced at each layer of IoT architecture which are following.

- **Perception Layer**

Since installed sensors are designed for specific purpose due to which they are tightly constrained [82] in terms of processing power and memory. They are not like regular computers which means it is difficult to implement resource demanding encryption algorithms for security purpose at this level[88].

Secondly, there is a danger of Denial of Service attacks. The attacker can connect the sensors to their own network and can push the limitation of these devices. Another way could be that the attackers can integrate their own sensor in the current setup and start flooding the network. Therefore, the confidentiality, integrity and authenticity of data must be ensured [91].

- **Network Layer**

The network of IoT is quite similar to that of standard internet layer but still is not as efficient and reliable as the standard one. Variety of communication protocols are available to ensure security like (e.g SSL, TLS) but the problem lies in implementation of these protocols. The information being sent over the network could be interpreted by Man-in-the-middle which can alter or diverge the information [72]. Protection from eavesdropping, network traffic congestion,

relay attacks etc and authentication of information being received and sent are core challenges in the network layer [91].

- **Processing Layer**

A large amount of data is going to be processed at this layer which means it is highly important that the data received is authentic. Malicious or junk information affects the intelligent processing and will generate unexpected and wrong results [90]. Implementing secure cloud computing, proper virus protection and encryption algorithms are some of the challenges of this layer [72].

- **Application Layer**

At this layer, the securities are developed according to the risk involved in different types of application [88]. For example , if you have an app that tells you the temperature of your home and afterwards you can adjust the temperature according to your needs. The risk in this kind of application might not be high but in case if the applications are related to self driving cars, medical services or any other similar service it would need high security as the risk is high.

The main challenges faced in application layer are about ensuing secure collection and sharing of data because it raise the concerns about the privacy, resource and access control and information disclosure [79].

It can be seen from the challenges mentioned above that implementing security and protecting privacy is highly important and secure infrastructure should be provided in order to use and get benefits from the IoT Technology. Government legislation should also play important role in encouraging and promoting development of IoT security as it may involve the question of national security.

CHAPTER 3

Analysis

AdeoOS Aps is a middle scale company located at Alberstlund in Denmark. The building of the company has two floors and the area is about $84 \times 24 \text{ m}^2$. It deals with the business of providing office space to multiple organizations.

The safety of the entire property is one of the top priority of the company. In order to achieve that the company has installed an electronic alarm and access control system. The aim of installing the system is to protect the property and assets from burglary or theft. Currently installed system is able to control access on the main door entrance of the building and the alarm system is securing all the rooms within the premises by using motion sensors. The system costed the company about 4000 danish kroner per room and there are approximately 50 rooms in the entire company. The system operates by manually turning it on when all of the staff have left the building.

3.1 Problem

The problem lies in the need of arming the alarm system manually when the last person leaves the building. As multiple companies are working under one roof and every company has its own office timing which means it becomes difficult to determine whether everyone has left the premises or not. One might think that it could be done by hiring a security personal to do this job but still he has to search the entire building and not to forget the never ending cost associated of hiring guard and probability of human error involved with this type of solution. After installing a super expensive alarm and access control system and then hiring a security personal does not seems to be an efficient and economical solution.

This problem is not limited to the companies only but have been seen everywhere whether they are shops, homes etc. We have to turn on the alarm manually before leaving the premises. Of course, for small scale properties like homes and shops it might not be a problem to search the premises but what if someone forgets to turn on the alarm unless it is not a remote alarm. However, coming back to our problem the current solution adapted by the organization is the duty assigned to the last person in premises to inspect the entire premises before arming the alarm and leaving the building. This again is definitely not an effective and convenient solution and has its own drawbacks.

The other main problem is more generic one and is related to the implementation of proper access control system for the employees within the building. As mentioned earlier, different companies are working under one roof and every company has its own working time. For example, one company might leave earlier than other companies and in that situation the alarm system wont be active which means anyone can enter (if not locked) or break into that particular office of the company. Moreover, no company would ever want any unauthorized person sneak around their vicinity during their absence.

Therefore, implementing access control system outside the building might be beneficial to the owner of the building but not to the companies working inside as they have their own enclosed space that needs to be protected from unwanted access.

3.1.1 Example

One real world burglary example with the current system was observed when cleaning lady was doing her duty late night in the building. The alarm was turned off for that specific amount of work time. As she was doing her job on the second floor, what happened was that the intruder managed to enter in one of the office room in the first floor and robbed valuables without being noticed. Therefore, it is highly important to build a robust and flexible alarm and access control system for each room in order to protect them from such scenarios in future.

3.2 Decomposition of Problem

Following is the list of the problems that have been encountered and identified while using the current alarm and access control system. Possible solutions that can be used to cope with identified problems are also discussed.

3.2.1 Manual Arming of Alarm System

The problem with the manual alarm also known as active alarm system is that it adds an extra layer of responsibility to the operator of the system. It bounds the operator to remember the activation of an alarm system. In this case, there is a possibility of human error that he might forget to turn on the alarm. Consequences of such error can result in unnoticed burglary or theft. There are following methods that can be used to reduce the human error.

1. **Reminder**

The risk of the user forgetting to turn on the alarm can be reduced by creating a method of regularly reminding the user about the event. This can be achieved by using various methods which are as following.

- a) Creating a system that can send a reminder sms or place a call to the owners mobile phone. The drawback of this solution is that the user must have mobile phone and the system must have GSM module in it to send message or place a call.
- b) Sending push notifications to the smartphone of the owner. For this solution we need to have an app of the system that can be installed on the owner's smartphone. In addition to that the system must have an internet connectivity.

Sometimes the method of reminding can be annoying to the user and he might end up turning it off completely. Therefore, flexibility must be provided to the user in terms of setting the interval of the reminder.

2. Passive alarm system

This is also known as an automatic alarm system. This type of alarm system turns on itself automatically without any human involvement. Automatic arming of the alarm system depends on several parameters set in the system. For instance, it could be a time parameter that would activate the alarm once specified time is reached. The main challenge in this type of alarm system is to ensure that the considered parameters are met otherwise it may not activate or may result in false alarm.

3.2.2 Lack of Remote Access

This is an addition to the above mentioned problem. The current system has no remote access that means if the user has left the building and on his way back he realizes that he forgot to turn on the alarm then he has to come all the way back to the premises in order to activate the alarm system. Therefore, it is important that the owner should have the functionality of remotely turning on or off the alarm system. The remote access can also help in providing valuable information to the owner like the status of the alarm.

This type of solution could be implemented using previously mentioned methods like internet or GSM. The real challenge in implementing this feature is that the remote access is prone to hacking. Therefore it should be highly secured and does not allow the hackers to break into the system and take control.

3.2.3 Presence Detection

Presence detection is one of the main problem in activating the alarm system. The AdeoOS current alarm system can only be activated once it has been confirmed that everyone has left the building. Now the question arises how to confirm that everyone has left the premises. As mentioned earlier, what companies do is that they assign the duty of the last person in the premises to search the entire building to see that

everybody has left and he actually is the last person in the building. Searching the premises is complex task and contain high probability of error.

Suppose the person conducting the search does not find anyone in the building and concludes that he is the last person remaining in the building, thus activates the alarm and leave the premises. However, unintentionally someone happens to be present in the building that went unnoticed during the search. This scenario will result in the activation of false alarm when the actual last person leave the premises. Each false activation of the alarm will result in fine of about 1000 danish kroner by the security firm. According to a report of US Department of Justice, the ratio of falsely triggered burglar alarm is in between 94 to 98 percent in US [87]. Improper arming and disarming of the system is one of the reason of false burglar alarm.

Another problem is that the search for the last person might be repeated by multiple people within the organization. Near the closing hour, everyone present in the building will start to think that they are the last one and will begin to search the premises. This will create a sense of confusion in everyone present in the building. Additionally, some people might start to guesstimate that they are not the last one and leave the premises without proper checking and turning on the alarm.

A proper solution must be implemented to detect the presence of personnel in the building. Following solution and technologies are proposed that can be used to achieve this task.

3.2.3.1 Counting

One of the easiest and simple approach is to count the number of people entering or exiting the building. These counters can be placed at the entrance point of the premises from where everyone has to pass at least once. This will help to count people and set up the alarm system once everyone has left the premises. One such device is known as turnstile.

1. Turnstile

Turnstile is a type of barrier or gate that allow only one person to pass at a time thus preventing tailgating or piggybacking. They are used in variety of places like airports, stadium, amusement parks and office lobbies. In terms of office use, normally the employee has to present an access token to pass through the barrier. They are used for access control and gives accurate counting of the people. There are different types of turnstile barrier known as mechanical, optical, full height and video turnstile. Different types of technologies are used in them to count people passing through the barriers. The drawback of the turnstile gate is that it is expensive, big and needs extra installation on the building entrance. Also in case of emergency exit it can be a bit problematic [35].



Figure 3.1: Tripod turnstile barrier

2. Infrared Beam

Infrared beam comprises of two parts, which is transmitter and receiver. The transmitter emits infrared light and receiver picks up the light. The transmitter and receiver are placed opposite to each other with distance apart. A straight path of light is created from transmitter to the receiver as shown in figure 3.2. When someone passes through this infrared line the connection is interrupted and thus signal is triggered that can be registered as count. IR beam can be used to count people entering or exiting the premises. Different types of combination and settings can be used to differentiate between human, pets or to detect direction. Mostly new IR beams are lightweight, low powered, inexpensive, weatherproof and insensitive to temperature [75].

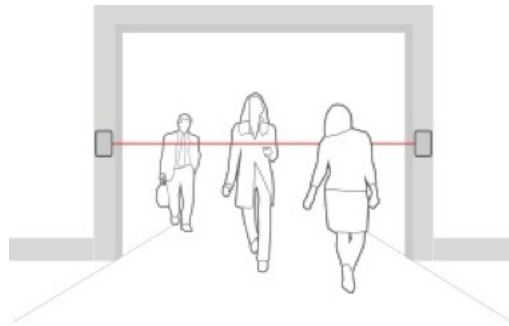


Figure 3.2: Demonstration of Infrared Beam Counter

The limitation of such technology is that it won't be effective in counting multiple people entering side by side in the room [74]. The adversary can emit his or her own beam of the light to the receiver to dodge the system. This can be prevented by using efficient algorithms where transmitter constantly changes the frequency of light on the receiver. This makes it difficult for the intruder to judge and dodge the system. [36].

3. Camera

The video camera uses advance image processing techniques to detect people and count them. The camera is mounted in an overhead position and is faced towards the ground. It has the ability to count multiple people passing through the entrance at once. This solution is more suitable for the places where there is a large flow of people. The accuracy of counting is highly dependent on the quality of image processing software[37].

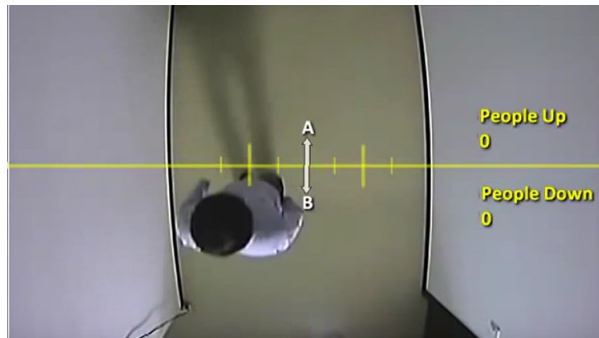
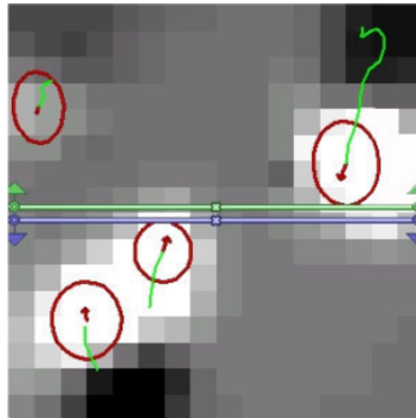


Figure 3.3: People counting using camera and image processing algorithm

Video counting is an expensive technique and require complex algorithms to detect people. Various parameters like lightening conditions, colours or shadows can affect the results. Camera can also sometimes raise the privacy issues[75].

4. Thermal Camera

Thermal camera detects the heat emitted by the bodies to produce an image of the environment and uses image processing algorithms to detect people and count them. The thermal camera solves the problem of illumination and privacy to some extent that was main problem in normal video camera. It has the ability to count multiple people simultaneously. The camera can detect the object even in the darkness.



The sensor 'sees' heat emitted from a person and tracks them as they walk

Figure 3.4: Thermal camera counter view

The drawback of thermal camera is that it is more expensive as compared to simple video camera. Also requires complex algorithm in order to perform counting.

Other relatively new technologies are Wi-Fi and Bluetooth counters [38]. These technologies are immature at this stage and only works if user have smart phone with necessary services enabled and applications installed.

3.2.3.2 Occupancy sensor

These types of sensor does not count the actual number of people in a particular area but helps to determine whether particular space or room is occupied or unoccupied. It can also help to ease the process of activating the alarm by reporting to the owner once the building gets empty. Following types of sensors can be used to detect the presence of a person in the premises.

1. Ultrasonic sensor

Ultrasonic sensor consists of two transducers, one is used to transmit and other one is used to receiver sound waves [39]. The sensors send out high frequency sound waves(inaudible to humans) which are reflected back when hit by a solid object and are sensed by the sensor. The reflected waves received from the static object will always be of same pattern. However, the sound waves reflected from a moving object will lead to disturbances in the sequence of waves and will be received in a different pattern from previous one, this change will help to

determine that a moving body is present in the environment [40]. The ultrasonic sensors are highly sensitive and can provide great coverage of an area including corners, gaps and blind spots. They can even detect the minor range of motion like hand movements [41].



Figure 3.5: Ultrasonic Sensor

The limitation of ultra-sonic sensor is that it cannot detect a stationary object and cannot differentiate whether a moving body is of a person or any other object. High sensitivity of the sensor can sometimes result in false alarm by detecting the non-occupant movements. For example, movements from high-level vibrations or flow of air through open windows or HVAC [41]. Carefull calibration and design is needed to cope with these issues. The temperature and medium can also affect the detection as the sound wave varies upon these factors[76].

2. Microwave sensor

Microwave sensor works in a similar way as that of an ultrasonic sensor. However, instead of using sound waves this technology uses radio waves to detect changes in the environment. This type of sensor is best suited for large areas and in places that have irregular shapes. They are also highly sensitive and have more coverage as compared to the ultrasound sensors, can even detect motion through glass, thin walls and plastic [42]. They are also not affected by the temperature[43].



Figure 3.6: Microwave Sensor

The limitation of microwave sensor is that they are expensive as compared to the ultrasonic sensors. High sensitivity can also pick up small movements like swinging of a curtain and will result in false alarm [44]. They may also pick up motion where not desirable like motion behind the walls. These sensors also cannot detect the stationary object.

3. Passive Infrared Sensor

PIR sensor is one the popular and widely used sensors for presences detection. PIR sensor consists of pyro-electric sensor which is used to measure infrared radiations. PIR sensor constantly measure infrared radiation emitted by an objects (like human and animals) and looks for changes in the pattern of the received radiation. Once pattern is disturbed the action is considered as a motion [45, 46]. PIR sensors are normally configured to look for infrared radiation in between the wavelength of 7 μ m to 14 μ m as human emitted infrared radiation lies in this range [47]. The sensor is called passive because sensor inside is not emitting any infrared radiation but only receiving it. Figure 3.7 shows a basic type of PIR sensor.

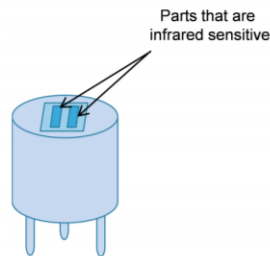


Figure 3.7: PIR Sensor

The PIR sensor is split into two portions each of which is sensitive to the infrared radiations. In an idle position, both portion will sense the same amount of radiation emitted from the environment. When someone enters the zone one portion will receive higher radiation than the other one. Similarly, when the person leaves the zone the amount of radiation on high portion drops thus this resulting change is detected as a motion [48]. The figure 3.8 shows how the signal is changed when human body enters and exit the field view of the sensor.

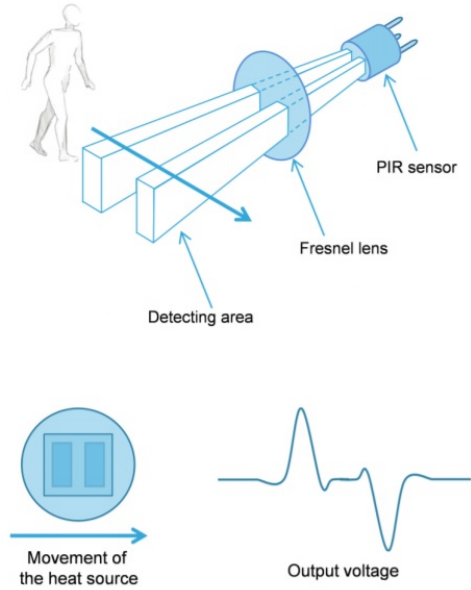


Figure 3.8: Working principle of PIR sensor

As it can be seen in the figure 3.9 that a Fresnel lens is used to focus the radiation coming from the source to the pyro-electric sensor [49]. The Fresnel lens help to increase the area of sensitivity. The concept is similar to that of camera lens which condense the large area onto their film. Specially designed plastic cover is used in front of the sensor that is split into small multiple sections consisting of Fresnel lens to create multiple small fields of detection as show in figure 3.9 [50].

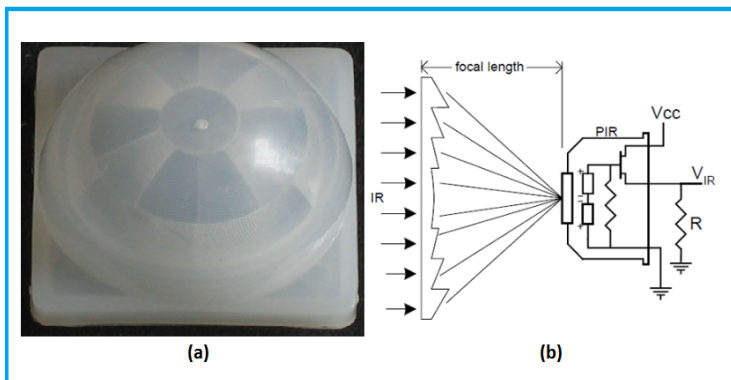


Figure 3.9: (a) Special PIR sensor cover (b) Basic function of Fresnel lens

The limitation of PIR sensor is that it can be easily affected by the sudden temperature and light changes [75]. The coverage range is in line of sight which means it cannot see around corner and sensitivity is affected if there is an obstruction between person and the sensor. The sensor should not be installed near HVAC or in front of glass window which might trigger false alarm [51]. The technology can be dodged by blocking the body heat emission. The PIR sensors also cannot detect stationary and slow moving objects. The sensitivity is higher if movement is parallel to the sensor but lower if the direction of movement is towards or away from the sensor[52].

4. Dual Technology Motion Detectors

Dual technology motion detectors are nothing but simply a combination of PIR sensor with the ultrasonic or microwave sensor to overcome the limitation faced in both technologies [53]. Combination will help to make both lateral and forward motion detection efficiently.

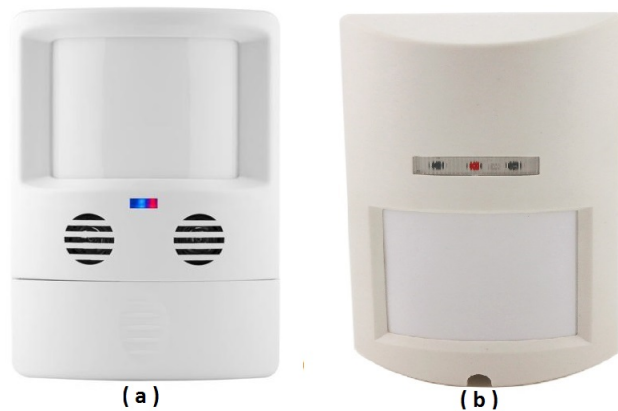


Figure 3.10: Dual Technology PIR sensor (a) with ultrasonic sensor (b) with microwave sensor[54]

Highly efficient algorithm must be designed to get benefits from both technologies altogether. Some program in a way that if both the sensors agree then the detection is registered otherwise not. One should remember that both have different ranges so it reduces the probability of activation [55].

3.2.4 High Cost

The cost of the current alarm system is quite high as told by the CTO of the company and still it does not mark up to the expectation of the user. Whereas, in the business of office hotel where multiple companies are working under one roof, each of them

must have their own independent local alarm system. As we can see that the current alarm is quite expensive and has many issues. We believe that a better alarm system can be made at much lower cost which can cope with the problems and challenges faced in the current alarm system. The new system will be cost effective, flexible, user friendly and will incorporate much more functionalities than the current one.

3.2.5 Individual Access Control System

The current access control system is only implemented at the main entrance of the building but individual offices inside the building lack access control system. An individual access control system for each office is important because each company have their own personal space and resources that needs to be protected from unauthorized people inside and outside of the building. Therefore, an access control system for each office would be implemented to manage access privileges and restrict unauthorized access.

The access control systems will be used to control the entry points of the premises. In our case that would be the individual office door which will be locked to restrict access and can only be opened by the authorized users. Following technologies can be used in order to control the entry point of the offices.

3.2.5.1 Locking Device

A locking device is basically a lock which is used to hold the door in its closed position so that no can enter into the premises. Mechanical keys can be used to open and close the mechanical door locks but they have lot of limitations as discussed in chapter 2. Therefore we will use electronic locks which can be made smart and then can provide more valuable information like who, when and how many times accessed the entry point etc.

Additionally, these locks can be interfaced with computers to make them smart. Following are the popular electronic locks used to secure the doors.

1. Electromagnetic Lock

The electromagnetic lock comprises of electromagnet plate and an iron plate. The electromagnet is attached to the frame of the door and the armature plate is attached with the door in such a way then when door is closed both the components get in contact to each other [56].

When power is supplied to the electromagnetic plate a temporary magnet is created which attracts the iron plate and holds it tightly so that the door remains closed [57]. The holding force can be from 300 to 1200 pounds depending upon the strength of the magnet [56]. Similarly, power is cut to demagnetize the plate and both the plates are released.

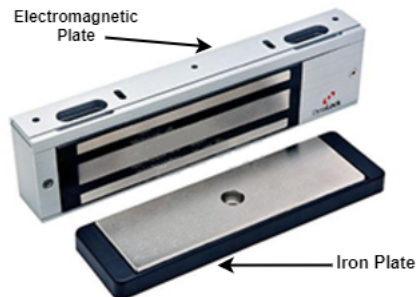


Figure 3.11: Electromagnetic Lock

2. Electric Strike

Electric strike is one of the popular lock and can be installed easily in most of the door frames without needing major modification. The electric strike lock is not a complete lock and doesn't fully secure the door alone as magnetic lock does but allows the door to be opened without unlocking the lockset of the door.

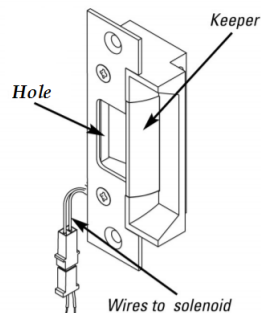


Figure 3.12: Electric Strike

The normal door consists of two major parts one is the handle and other one is the latch or bolt that sticks out from the side of the door. This latch sticks into the hole present on the side of the door frame to prevent door from opening. The hole is made up of metal plate and is known as strike. The electric strike is similar to that of simple strike but have a moveable piece of metal known as keeper, which can swing or stay firm depending on the power provided. This moveable part allows the door to open and close without moving the latch [58].

3. Electric Drop Bolt Lock

Electric drop or dead bolt lock as the name suggest consist of a solid bolt that sticks out of the locking plate when power is applied to the lock. This bolt is directed into the strike plate installed on the doorframe to keep the door locked. The power is cut off in order to unlock and release the door [15].

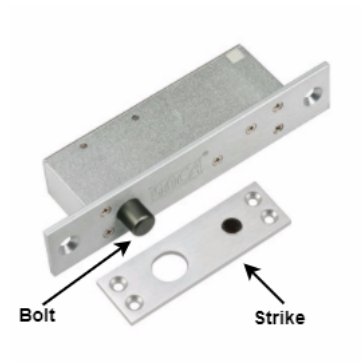


Figure 3.13: Deadbolt Lock

3.2.5.2 Reader

Reader is a device that plays similar role as that of a keyway in traditional lock and key system. This is where user present his key (credentials) for authentication. Following types of readers comprising of different kinds of technologies are used normally to take input from the user.

1. Coded Lock Reader

This type of reader consists of numeric or alphabetic keys which require user to enter PIN or password directly to the access reader [59]. The advantage of using this type of reader is that the user does not need to carry any additional equipment like keys or tokens which can be lost easily [60]. PIN and passwords are the most simple and popular credentials. However, the drawback associated with this type of method is that anyone who knows the password can get access through the entry point. In addition to that anyone can shoulder surf while someone is entering the password on the reader. Passwords may be stolen if used over the network by the hackers.



Figure 3.14: Simple Numeric Keypad Reader

2. Magnetic Stripe Reader

The magnetic card reader is used to read special type of cards consisting of a stripe of magnetic tape on it where user credentials are encoded. The card needs to be in contact with the reader and works by swiping the card magnetic stripe through the reader. The technology is cost effective and has been widely used. However, the strip is susceptible to dirt and magnetic field [60].



Figure 3.15: Magnetic Stripe Reader

3. Proximity Reader

Proximity reader is used to reads the proximity tokens or cards consisting of a coil of wire and microchip inside them where user credentials are stored. When the card is brought to the proximity of reader, it absorbs the Radio waves emitted by the reader to power up the chip. Once the chip is powered up, it sends the stored credential information to the reader. This type of technology is used in almost every other organization because it is cheap and fairly easy to use [61]. The cards and tokens dont require any direct contact with the reader, just have to be in the proximity of the reader. The problem with the proximity cards is that they can be stolen, lost or forgotten easily. However, cards can be deactivated easily to restrict access [62].



Figure 3.16: Proximity Card Reader

4. Biometric reader

Biometric readers are used to identify the unique physical characteristic of a person such as fingerprints, palm, face, voice, and iris to grant access. They are also used to detect behavioural characteristics like hand-written signature or gait style. It overcomes the issues faced with the pin and card technologies. However, this type of reader is quite expensive and provides slower access as compared to others. [62]. The accuracy of biometric readers is still not efficient and can cause nightmares to the administrators.



Figure 3.17: Biometric Fingerprint Reader

The sensor is placed on the reader that is used to sense particular characteristics of the user and sends the input to the controller which compares the sample with the stored template in order to verify the legitimate user. As this technology is becoming more mature day by day, the companies are starting to move towards this type of authentication mechanism [63].

Most of the time several types of readers consisting of different technologies are combined together into one reader to increase the level of security.

3.2.5.3 Controller

Controller consists of a single-chip computer known as a microcontroller. It is called a single-chip computer because the CPU, memory, input, and output ports are all embedded together into one single chip. These kinds of chips are mostly used for embedded

applications because of their features like low power consumption, low size, low cost and high flexibility. Chips can be programmed using low level language like assembly or high level languages like C. Some of the popular chips are PIC16F877A , Atmega8 etc. It can be used to interface different kind of electronics devices together like in our case it would be sensors, readers and door locks etc. It can process data and can take decisions according to its programmed software.

Manufacturers of the microcontrollers have made it quite simple to use the chip by creating an embedded development boards and softwares that allow the user to easily program, flash and debug controller through computers. The most popular boards are as following.

1. Arduino

The Arduino board is a small open source development board consisting of microcontroller and some electronic components. The hardware and software of the board is made in such a way that it can be easily used by anyone. It can be connected with physical objects (like sensors, lights, actuators) to manipulate data and perform actions according to the program in the chip. Nowadays, it is also known as an IoT tool because it act as a gateway for connecting physical world with the digital world.

The Arduino boards comes in various models and sizes consisting of different specification and functionalities based on the chip installed on the board. However, the good thing about Arduino board is that all of them can be programmed in a similar manner. One of the popular Arduino board is known as UNO and it consists of following main components as shown in the figure 3.18.

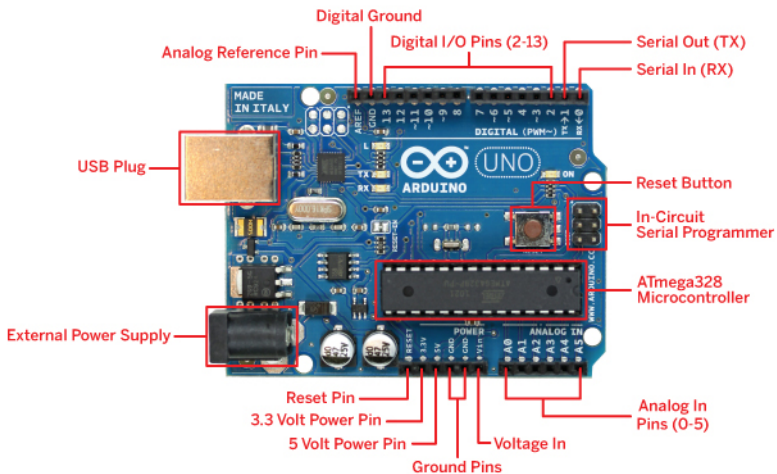


Figure 3.18: Arduino Board

The *External Power Connector* is used to turn on the device and can be powered up with the voltage range of 7 to 12 volts. The *USB Plug* is used to upload program from the computer and can also be used to power up the Arduino board.

The *Atmega328 Microcontroller* chip is used in the Arduino board where all the processing takes place and also known as the brain of the board. The red colour *Reset Button* and *Reset Pin* are used to reset the controller board in case if it hangs up or for some other reason.

The black pin headers consist of *Voltage* and *Ground Pins* providing power of 3.3 to 5 volts which can be used to power up external attached devices. Five *Analog Pins* can be used to read analog data. *Serial In and Out* pins are used for serial communications. The *Digital I/O Pins* are used to input and output data in digital form. The *Analog Reference Pin* is used to set the voltage level of analog pins externally should be in the range of 0 to 5 volts. Last but not least the *In-Circuit Serial Programmer* is another type of serial interface known as Serial Peripheral Interface.

The open source software environment for programming Arduino is known as integrated development environment (IDE) which supports languages like C and C++.

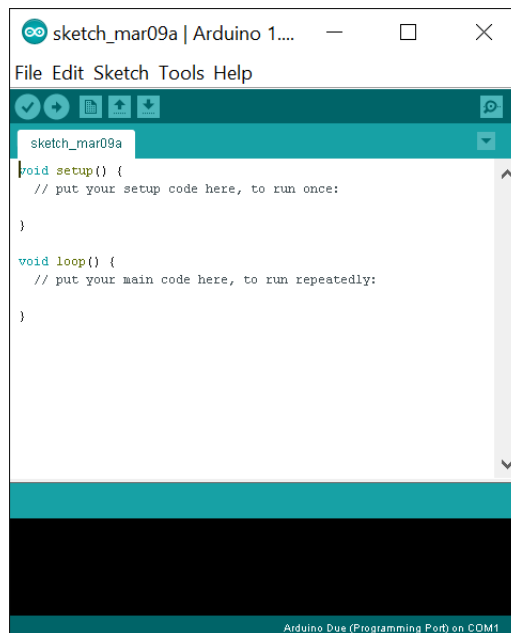


Figure 3.19: IDE for programming Arduino boards

The IDE consists of graphical user interface that is used to write, compile, debug and burn the code into the microcontrollers of the boards. It consists of multiple examples and libraries that can be used to ease the process of programming. You can also pull additional libraries from the Arduino server by using the application library manager. The interface is fairly simple and makes easier to program the microcontrollers.

2. Raspberry Pi

A Raspberry Pi is not a pure microcontroller but credit card size small computer board that provides the functionality of computer as well as that of a microcontroller to some extent [64]. It is an inexpensive piece of board that runs specially designed Linux operating system known as Raspbian. It also has the capability of running other operating system like Android, Fedora, Firefox OS, and Windows 10 etc. There are different models of raspberry pi that comes with the different specification and functionalities. Some well-known models are Raspberry pi A+, 1, 2 etc. Similarly, like Arduino the raspberry pi consists of GPIO pins to interact with the physical objects but to a limited extent. As the GPIO ports features are not equal to that of Arduino. However, Raspberry Pi supports almost all programming languages to interact with the GPIO pins. Additionally, it has the powerful capability of networking. The raspberry pi board typically consists of following components as shown in the figure 3.20.

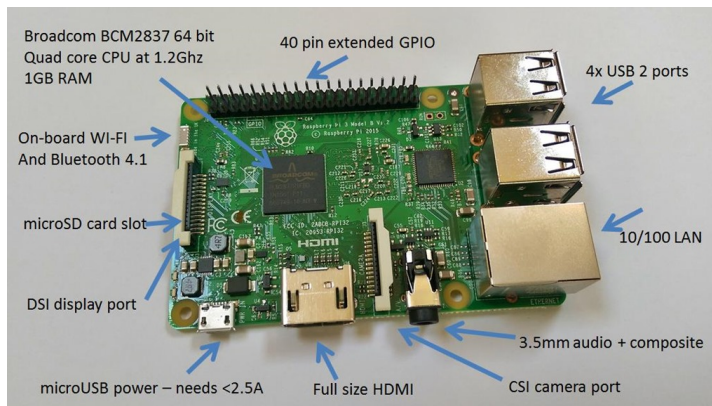


Figure 3.20: Raspberry Pi Board

The board consists of *MicroUSB slot* to power up the device with 5 volts. The *DSI Display Port* is used to connect external 15 pin connector LCD for visual display. Also, *HDMI Port* is available to connect screens and monitors directly to the board. *MicroSD Card Slot* is used to support mcroSD card which act similar to that of a harddisk in the computer. The operating system and other programs are installed in the SD card memory. One *3.5mm Audio Jack* is available to attach speaker or headphones with the system.

The board is powered by *BoardcomBCM2837* 64 bit Quad core processor chip consisting of 1 GB Ram. It includes 40 pin *GPIO* ports which includes digital pins, USART, SPI and I2C communication support. Four *USB Ports* are provided to attach keyboard, mouse, and many more compatible devices with the board. *Ethernet port* is also provided to connect Ethernet cable for internet connectivity. Additionally, *On-board WiFi and Bluetooth* connectivity is also provided. A *CSI camera port* can be used to connect external 15 pin header camera.

Overall, board is a complete computer and also serve as one of the candidate of IoT tools. It can be used along with the Arduino to overcome the limitations faced in both devices and turn the board into a powerful device.

3.2.6 Limited Hardware and and Memory Capacity

The current access control system and all other access control systems where processing is done at the controller level has one huge problem. That is the hardware and memory limitation which is usually fixed and if your needs and requirements increases then the current system might not be able to accommodate it. For example, the current system might be able to accommodate 4000 users and suppose that your business scales up and your work force has increased. How the system with limited capacity is going to accommodate it ?

The access control system also records valuable data which grows day by day. The processing and storage of this data is another problem. It is clear that the day by day growing data at some point will become problematic for the controller based systems due to their less memory and processing power. The best solution is to use these controllers for local processing and filtering of unnecessary data. Afterwards, useful data should be push to the powerful units like computer, server or cloud. The data will be efficiently processed as those devices have super-fast processor power and huge memory units.

3.3 Requirements

After carefully analysing the problems faced with the current system and needs of the company a list of requirements is created that the final prototype should have in order to provide optimal solution.

1. Developing an autonomous office alarm system using alarm system components that can perform the following tasks.
 - a) Arm the alarm autonomously in individual room.
 - b) Ability to arm the alarm manually (optional).

- c) Occupancy detection using currently installed PIR sensors from the old system.
 - d) Once armed it should be able to sound the alarm in case of intrusion or unauthorized entry.
2. Developing an access control system using access control system components that can perform the following tasks.
 - a) Can take input from the user for authentication.
 - b) Providing access to the premises for authorized members.
 - c) Restricting access to the premises for unauthorized member.
3. Developing a firmware for an alarm and access control system that can perform following tasks.
 - a) Interface components of alarm and access control system with the controller board and read data from sensors, door locks and readers.
 - b) Analyse data and perform local processing to filter out useless data received from the components.
 - c) Get configuration and instructions form the server and update the controller board accordingly.
 - d) Control door lock based on the instructions received from the server.
 - e) Ability to send data received from sensors, door locks and readers to the server.
 - f) Alert server about critical situations.
4. Developing main control software on the server side that will be able to perform the following tasks.
 - a) Recieve data and send response from and to the controller board respectively.
 - b) Analyse data received form the controller board and issue orders to the board accordingly like Open or Close the door etc.
 - c) Provide configuration settings to the controller board.
 - d) Authenticate credentials to provide or restrict access to the premises.
 - e) Control the sates of the alarm (Unarmed, Warning and Armed).
 - f) Store data received from the controller board into the database.
5. Developing a simple mobile app that can allow users to Arm or Disarm the alarm system remotely.

It the end all parts of the final prototype will be evaluated based on the requirements set in this chapter in order to prove the feasibility of the system.

CHAPTER 4

Design

An overall autonomous alarm and access control system architecture is proposed based on the requirements mentioned in the previous chapter. This architecture will help to reduce and overcome the challenges faced in the current alarm system. The entire architecture can be seen in figure 4.1.

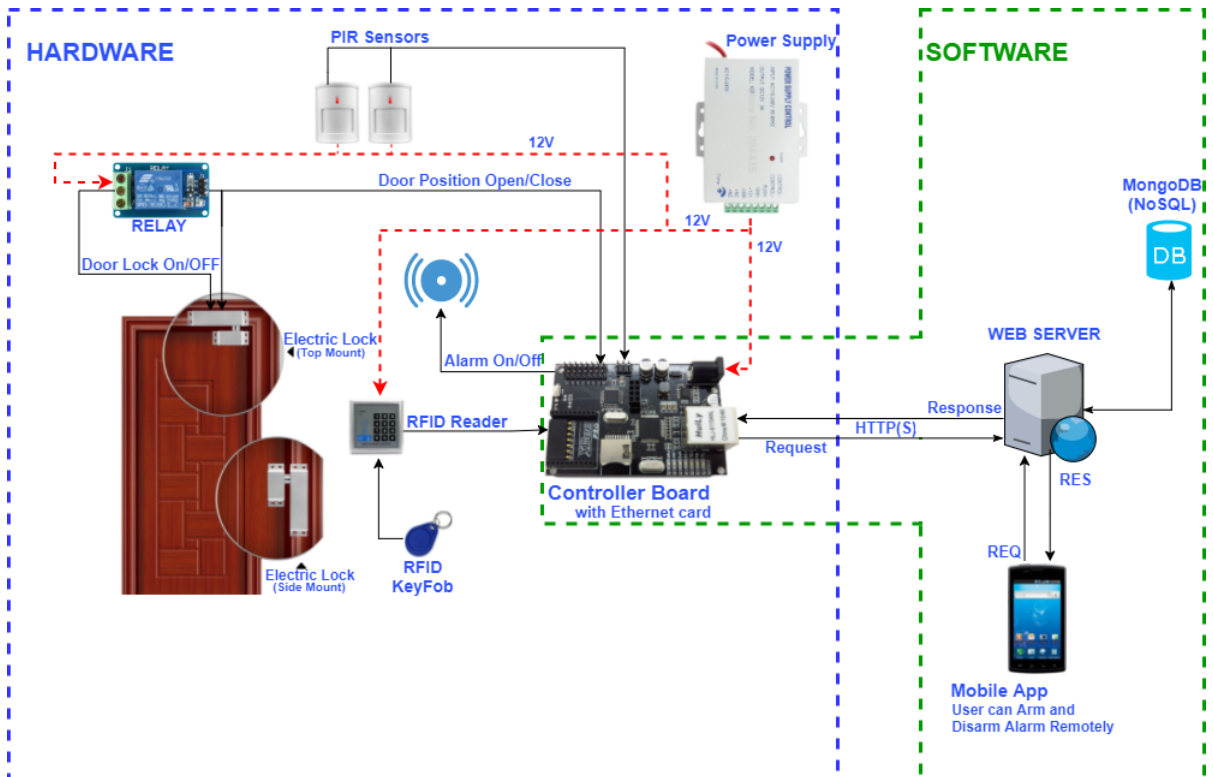


Figure 4.1: Proposed Architecture Diagram

The proposed architecture diagram shows all the components of the system and

their relationship between each other. As it can be seen from the architecture diagram that the system consists of two major parts hardware and software.

4.1 Hardware

The hardware part is the one that is going to be installed in each room and will be responsible for collecting, pre-processing and sending data for final processing to the server. The role of each hardware component within the system is as follows.

4.1.1 PIR Sensor

The PIR sensors are installed in each room for motion and occupancy detection. PIR sensors used in our design are simple one as they were already installed in the building but one can replace them with the dual technology motion sensors to increase precision. The PIR sensor main board consists of three 2-pin terminal connectors as shown in the figure 4.2.

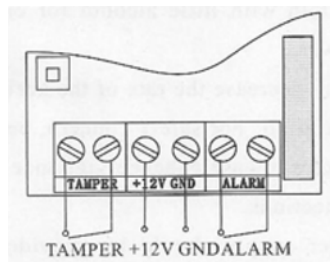


Figure 4.2: PIR sensor Box Circuitry

Tamper

Tamper is used to prevent people from opening the cover of the sensor box. Tamper generates the signal to trigger the alarm system in case someone tries to open the plastic cover of PIR sensor. The tamper connector is basically connected to the micro switch which consist of a spring on the top of the button. When the cover is placed on the box, the spring is compressed which in turn presses the button and keep on holding it to that position as long as the cover in on.

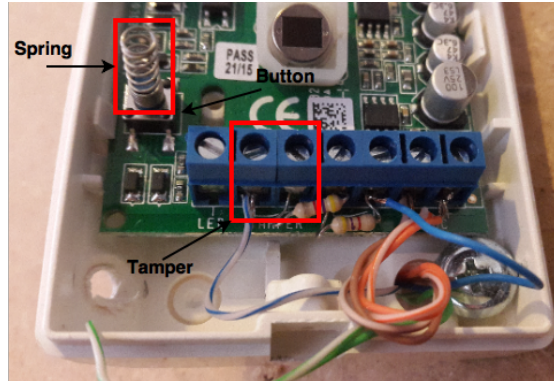


Figure 4.3: PIR sensor without cover

If someone removes the cover the spring will be decompressed which in turn will generate the alarm trigger signal.

Power Connector

The (+12/GND) connector is where we supply power which is 12 volts in this case to turn on the PIR sensor.

Alarm Connector

The alarm connector consist of a magnetic switch which is normally closed when no motion is detected and opens up when a motion is detected. These two states (Open and Close) help us in determining whether someone is in the room or not. A simple circuit demonstrating the working principle of Alarm connector is shown in the figure 4.4.

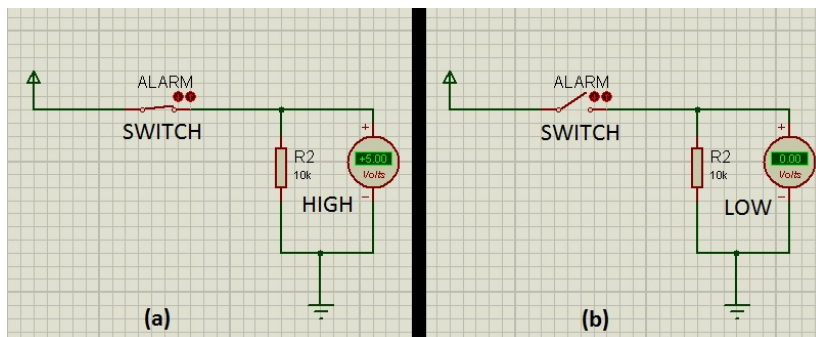


Figure 4.4: Demonstration of Alarm Connector

The figure (a) show that when the sensor is not detecting any motion, the switch will remain close and whatever signal we send from one end will be received at

the other end. Whereas, looking at figure (b) when a motion is detected the switch will open and we will not be able receive any signal at the other side. The switching is done automatically using relay. The operation of relay will be discussed in next section.

The working principle of tamper connector is also same as that of an alarm connector. Thus, the PIR sensor provides two functionalities one is motion detection and another one is tamper detection. However, there is one huge flaw in the above mentioned wiring design that can be exploited by the hardware hackers. The hack is shown in the figure below.

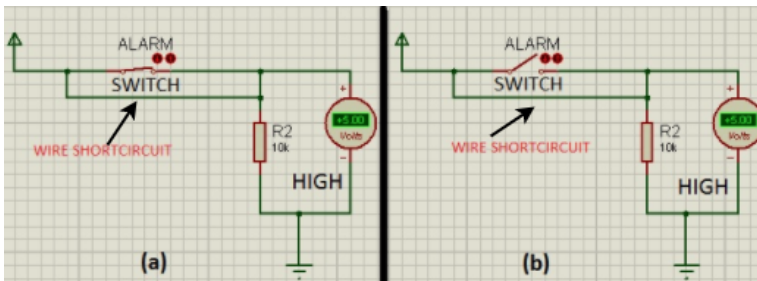


Figure 4.5: Shortcircuit Hack

The hack is that the intruder can simply bypass the alarm connector signal by short-circuiting the wire entering and exiting the PIR sensor box. As it can be seen from the figure 4.5 that once the switch is bypassed using wire short-circuit the switch will have no effect on the signal output. This means the system will think that there is no motion within the room and intruder can simply walk around without being detected. However, this hole can be patched using *End Of Line* resistors.

A resistor is an electronic component that is used to limit the flow of current within the circuit. This means that if the resistor is used in the circuit, the sensor HIGH signal will be bit lowered than the maximum value. For example, if sensor gives 5 volts when closed it might start giving 4 volts max when resistor is connected to it. Now we have created three different signals which are 0 volt for motion, 4 volt for no motion and 5 volt for short circuit detection. Now the controller system will be able to make precise decision based on these three different defined signals.

This method would only work if the resistors are placed at the end of the line which means inside the box after the alarm connector terminals. The hacker will not be able to remove the resistors as they are kept in the box which is tamper proof.

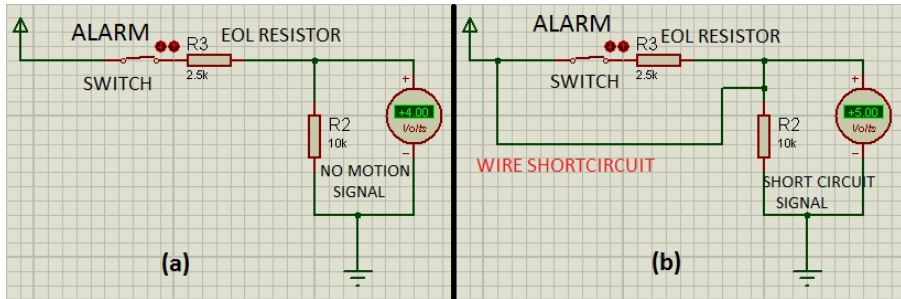


Figure 4.6: EOL resistors to acknowledge short circuit

The figures 4.6 shows how by placing an EOL resistor we can simply detect short circuit within the system. Similarly, additional signal are created to different between motion detection and circuit open detection. Thus, with this configuration the PIR sensor will be able to provide us with four different signal that are open circuit / Tamper, short circuit, motion and no motion.

4.1.2 Electric Lock

Electric locks should be installed at each office room that can only be opened by an authorized person. As per requirements of the company, we are using electric drop bolt lock but one should remember that any lock mentioned in the analysis can be used. The purpose is to just secure the entry point.



Figure 4.7: Electric Drop Bolt Lock

The lock consists of four wires red, black, yellow and white. The red and black wires are used to power up the lock. Whereas, white and yellow wires are signal lines that helps in determining whether the door is open or closed. The lock has fail safe

configuration it means when the power is applied to the device it will lock itself and when power is removed it gets unlocked. One thing to remember is that the door should only be locked if the signal wire shows that the door is properly closed.

The lock operates at 12 volts and microcontroller is a low power device operating at 5 volts. It means that the microcontroller doesn't have enough power to operate the lock by providing on and off signal. Therefore, an additional component known as relay is used to bridge the gap between different voltages.

Relay

Relay is an electromechanical switch that uses low power signal to control relatively high power devices. It consists of an electromagnet that when energized will open or close the switch [65].

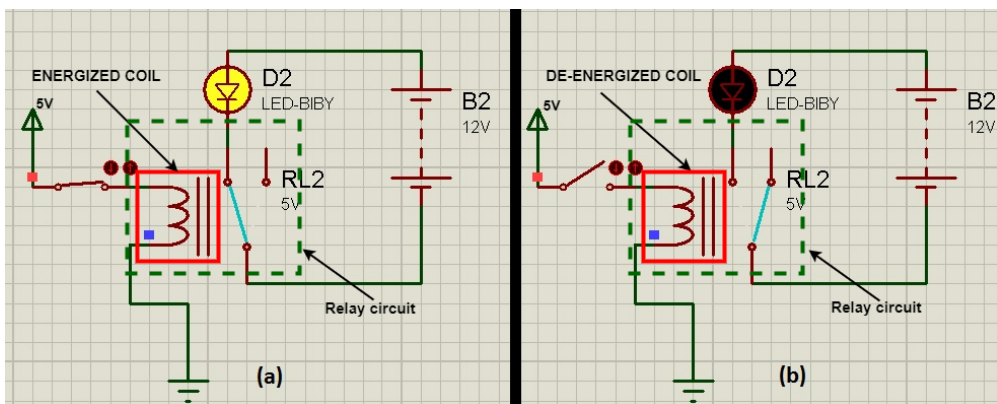


Figure 4.8: Working Principle of Relay

The figure 4.8 shows the inside circuitry of the relay device. It consists of a coil which creates a magnetic field when low power signal is passed through it. This temporarily created magnet will push the switch on the other side towards the close position. Thus, completing the overall circuit at the other end. In our case the led shown in the figure 4.8 will be replaced with the lock that will be powered up using high voltage. Similarly, in order to break the circuit the high voltage signal from the relay is disconnected which in turn moves the mechanical switch back to its original position.

Now with the help of relay the controller can easily control high power devices.

4.1.3 Reader

The reader is going to be installed in front of each secured entry points. The user will be able to present his credentials for authentication. Once authenticated the door

will be unlocked providing access to the office. The reader used in our prototype is proximity reader along with the keypad. These types of readers are cost effective and double combination in them provides the capability of multi factor authentication. However, advance reader like biometric ones can also be used but of course it will impact the cost of the entire system.

There are variety of communication protocol present to send data from the reader to the microcontroller like Wiegand, RS232, RS485 and data/clock. All of these protocols serve the same purpose of transmitting data from the reader to the controller board. The reader that we are using provides support for the first three mentioned communication protocols. We are going to use the Wiegand Protocol as it can be easily attached to our controller board. For other ones we would need to use the converters.

The reader that is being used while developing the prototype is shown in the figure 4.9 along with its wiring connection table.



Figure 4.9: Reader along with the Wire Connection Chart

The *Red* and *Black* wires are used to power up the device. *Blue* wire is used to control lights of the reader and yellow one is used to control the buzzer present inside the reader. The remaining two wires *D0* and *D1* are used for communication using Wiegand protocol.

Wiegand Protocol

The Wiegand protocol is used to transmit data from the reader to the controller board [66]. It basically consists of three wires known as *Data Zero* (*D0-green wire*), *Data One* (*D1-white wire*) and the ground *GND* (*black wire*). The *Data Zero* is used to send binary 0 and *Data One* is used to send binary 1 from the line.

The two separate data lines are provided with the 5 volt signal to keep the lines voltage high all the time. In order to send binary 0 the *Data Zero* line is pulled to low voltage whereas the *DATA One* pin remain stable at 5 volts [67].

Similarly, if we want to send binary 1 the voltage on *D1* pin is lowered and *D0* is kept stable at 5 volts.

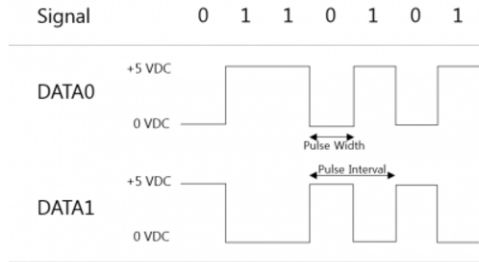


Figure 4.10: DATA0 and DATA1 lines sending binary 0110101 [68]

The figure 4.10 shows how the data is transmitted over the data lines. The pulse width of each signal is about 50 microseconds wide and the pulse interval between two adjacent signals is 1 millisecond. Last but no least the data packets are separated by the time interval of 500 milliseconds [69].

Using this protocol we can send the data present at the reader to the controller board for authentication.

4.1.4 Power Supply

The power supply used is 12 VDC as all the components within the system requires at most 12 volts. The power is supplied to the following devices.

1. Sensors
2. Readers
3. Relays
4. Electric Locks
5. Controller Board

Each hardware unit installed in the room will have its own power supply to avoid overloading and single point of failure.

4.1.5 The Controller Board

The controller board is the central unit of the entire architecture that connects all the components of the system together. It is also responsible for controlling all other hardware components within the system. The controller board used in the design is known as IBoard [70]. It has enough pins and ports to connect all of our design components together.

The board is basically an Arduino board but is more flexible and have Ethernet module already embedded in it. Whereas, in case of Arduino a separate extension of an Ethernet shield is needed to connect the board to the internet. The connection diagram of the board with other hardware components is shown in the figure below.

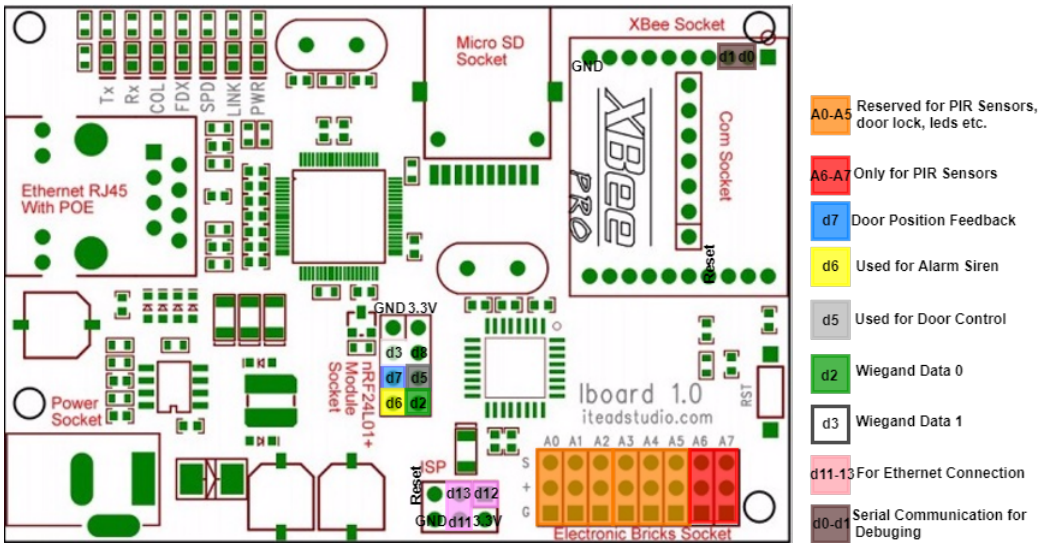


Figure 4.11: IBOARD with connection configuration

One unique feature of this board is that it can be powered over Ethernet cable. PoE is a technology that allows the network cable to carry both power and data lines in one cable to the end device. The end device (should support PoE) then can use this power to turn itself on rather than using an external power source. However, our board is not purely POE therefore we need to use an extra splitter at the end device that will separate the PoE data line and power line so that the cables can be connected easily to the board. The benefit of PoE technology is to reduce extra wires and power supplies cost.

The drawback of the technology is that not all the network switches support PoE and the one that supports are expensive one. However, we can use PoE injector instead of power supply to get the same functionality. For this project the company

has its own PoE switch already installed in the building so it is easier for us to power the board using PoE.

The program loaded on the controller board helps it to collect data and perform local processing on it. Once controller is done with the processing it communicates with the server to exchange data using http protocol over TCP connection.

- **HTTP**

HTTP is a stateless application layer protocol that is used to enable communication between client and server. It helps to transfer data in different formats like Hypertext, plain text, multi-media files etc. between client and server. It functions as a request and response protocol between client and server. A TCP is used to set up a connection between client and server and once its established the client can send the http request to the server. The http protocol provides following methods to specify what actions to be performed on the data present at the server.

Method	Action
GET	Retrieve data from the resource
POST	Submit data to the resource
PUT	Update data of the resource
DELETE	Delete a specified resource

Table 4.1: HTTP Methods

The format used for the http request and response messages is shown in the figure 4.12.

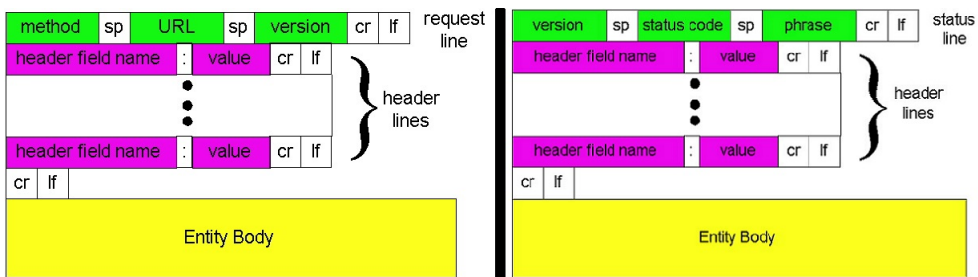


Figure 4.12: On the left request message format and on the right response message format

The first line is known as the request line which consists of three fields specifying http method, URL and http version. The header consists of the information like host name, language type etc. The last one is the body containing the message.

The response format is quite similar to the first one but its first line consists of http version, status code and reason. Status codes are used to define whether the communication was successful or there was any error.

The http request and response messages are text based which means information sent over the network can be read by the Man in the Middle. For the sake of prototype and the limitation of the board we are going to use HTTP but in final product implementation encrypted HTTPS protocol should be used.

4.1.6 Block Diagram

To sum up the hardware architecture a block diagram is shown in figure 4.13 which gives an overview of the general connections and interaction of components with each other.

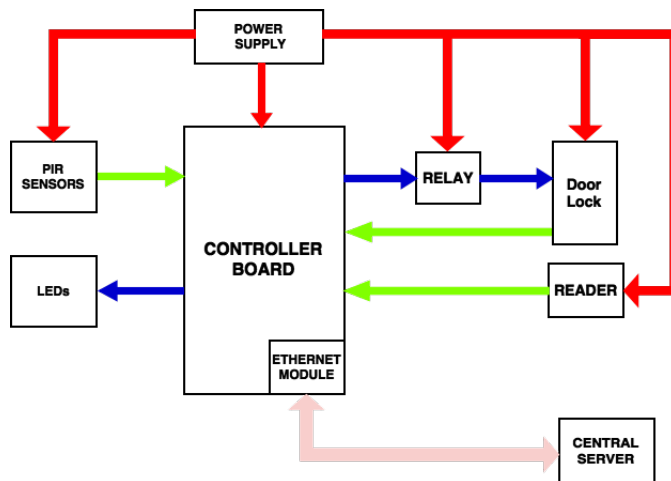


Figure 4.13: Block Diagram of the Hardware System

The directional arrows represent the flow of data to and from the controller board. The data coming to the controller board is usually input of the controller and the data going away from the controller is basically output. The two way communication is only between controller board and server to send collected data and receive instruction respectively.

4.2 Software

The software is divided into two parts. One is the software for the controller board known as firmware and the other is the software for the central server which act as a main controller of the entire system.

4.2.1 Firmware

The microcontroller itself cannot perform any action. A firmware is needed to be installed on the flash memory of the microcontroller which enables it to control and communicate with other devices. It also enables the microcontroller to perform processing on the data received from all those devices. A firmware is simply a computer program that is installed on the microcontroller to make it functional. Based on the requirements of the system an overall flowchart of the proposed firmware is shown in the figure 4.14 present in the next page.

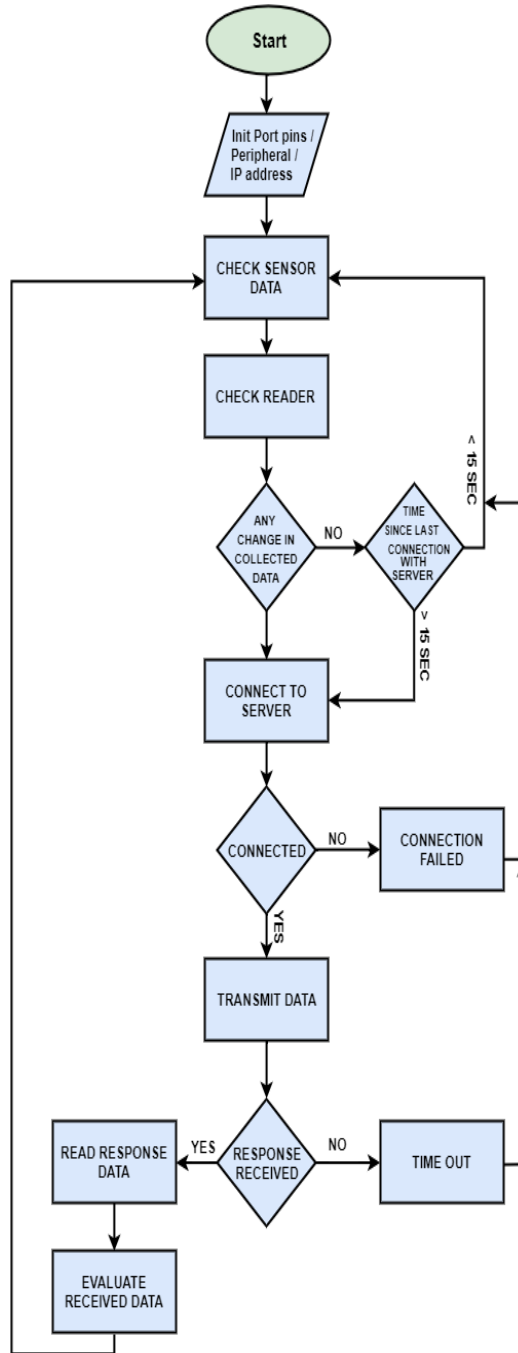


Figure 4.14: FLOWCHART OF FIRMWARE

The flow of the program start by initialization of all the ports, pins, IP addresses, serial number of the controller and peripherals so that the controller can communicate and control all of the devices connected to it. The controller basically polls in a constant loop where it checks for any data being received from the sensor or the reader. If the PIR is triggered or reader is used the controller will immediately notify the server about the event and server response back with the instructions that should be executed by the controller.

The instructions received from the server could be like unlock the door, arm the alarm system or turn on the alarm siren etc. However, if there is no change in PIR sensors then the controller will send the data after every 15 seconds as we dont want to push unnecessary data to the server every second. This will help the server to log only important data. The controller keeps on running infinite loop forever in order to maintain working of the alarm and access control system in the room.

However, there is one exception in the design that if in case the controller never connects to the server it will keep on running in an internal loop forever. The solution for this one is quite simple that is implementing a watch dog timer that will restart the board and log this action in the EEPROM of the board. If in case after several restarts it still doesn't work as expected than it should inform the owner of the system about failure. Fault tolerance is not in the current scope but of course is kept in mind for future work.

4.2.2 Central Server

The central server is the main commander and backbone of the entire system architecture. All of the data gathered from the sensors by client controller boards reaches to the central server. When a data is sent to the server it is stored (in database), processed and instructions are issued based on the decision made by the central server. In the first version of the prototype the server comprises of simple Web APIs to facilitate controller boards and the mobile app users. The Web APIs exposes endpoints to the resources which are separated so that the perspective clients can easily consume desired services. The clients interact with the resource using HTTP methods and APIs define how methods are operated on the particular resource. In the project our Web APIs is centered around two main resources which are as following.

- **Boards**

The boards resource is designed especially for the controller boards where they send data for processing. In the current design POST method is used by the clients independently for posting data. After receiving request, the server wraps the processed data in the response body which basically consist of the next possible instructions for the controller board. A sequence diagram is shown in the 4.15 which represents the above mentioned interaction between the board and the main server.

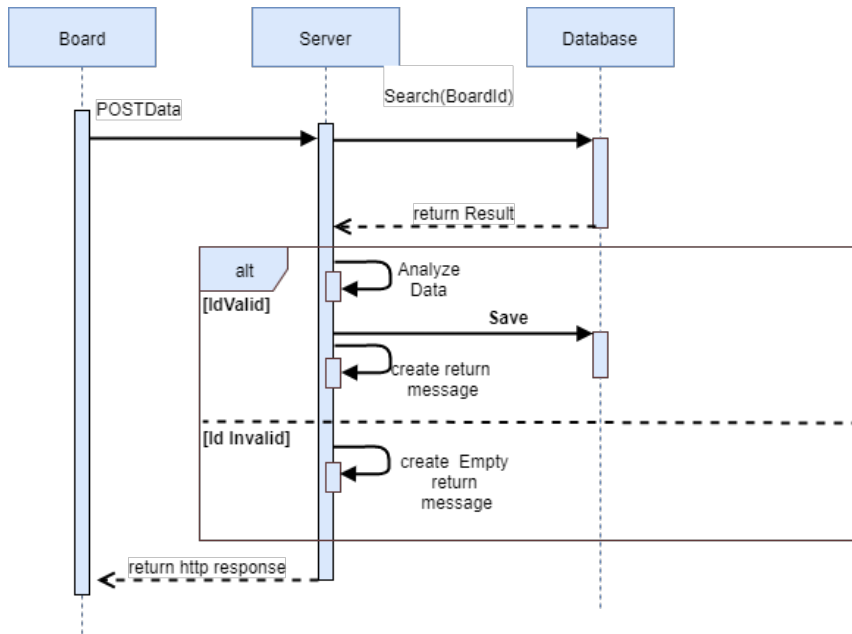


Figure 4.15: Interaction between Board and Central Server

- **RClients**

The RClients resource is consumed by the mobile application users. Currently, POST method is supported and all other actions are defined in that one method. The clients can use it to log into their mobile application by providing credentials. Once logged in the client can get the list of rooms assigned to him and can remotely arm or disarm the alarm of the rooms. A sequence diagram that shows the possible interaction between mobile users and central server is shown in figure 4.16.

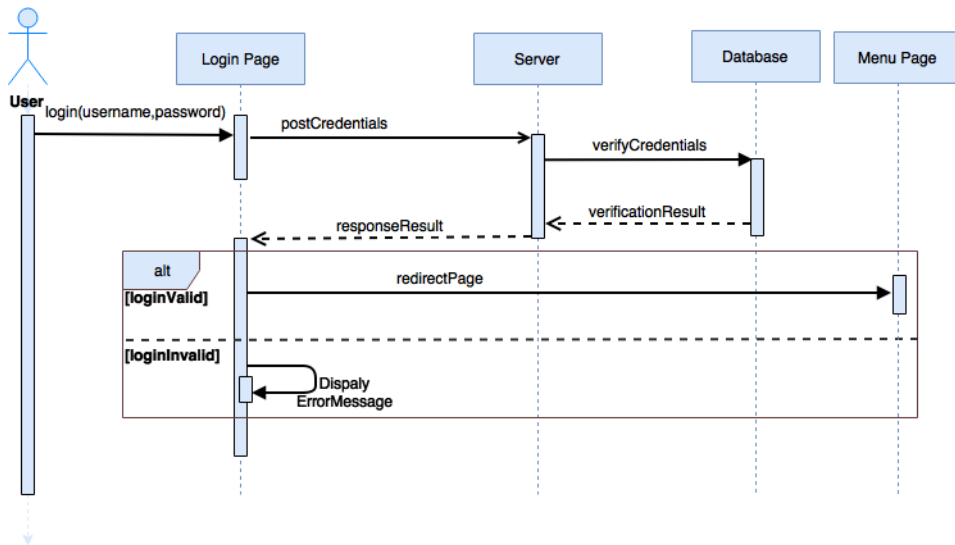


Figure 4.16: Interaction between mobile user and Central Server for Log-in

As it can be seen from the figure 4.16 that the client needs to Login before he can get list of his rooms. The credentials provided by the user are not only used for authentication purpose but are also used to query database so that we can easily find the rooms assigned to the user.

After successful login the client is presented with the menu page where he can retrieve the list of his rooms and can remotely arm or disarm them. This interaction is also shown below in the form of sequence diagram.

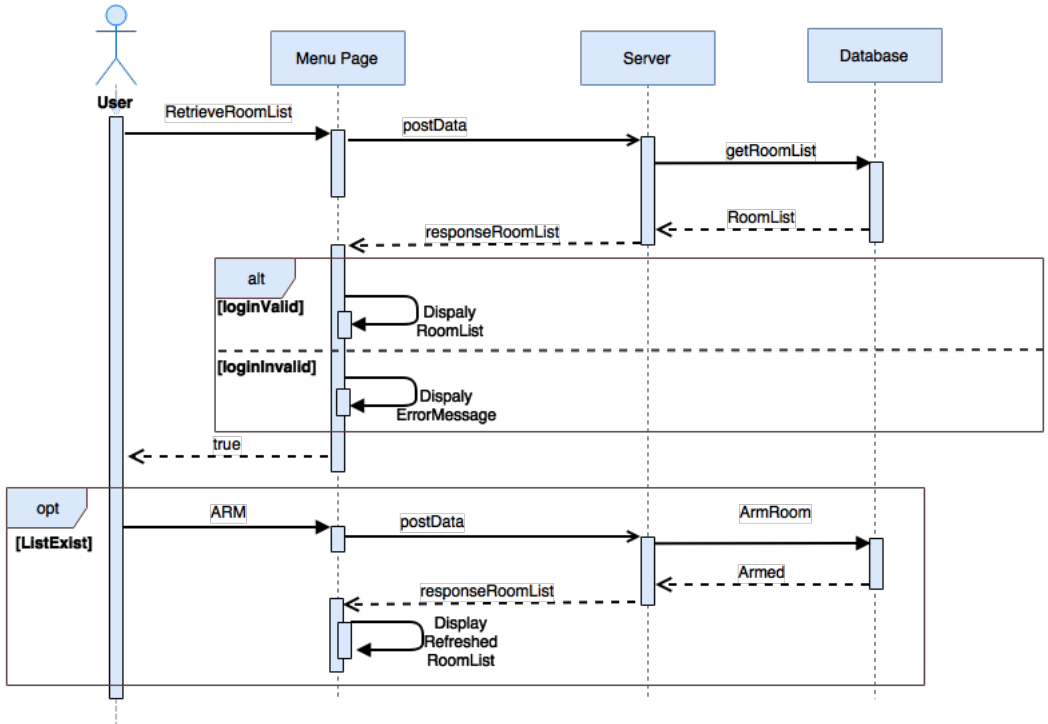


Figure 4.17: Interaction between Mobile User and Central Server for retrieving room-list and arming room

4.2.3 Algorithm

The algorithm running on the central server compares newly arrived data from the controller boards with its previously stored data. The algorithm is time based which means once the algorithm starts getting similar signal from the PIR sensors like no motion for a specified amount of time it starts to predict that the room is unoccupied and then will change the state of the alarm. Otherwise, if the motion is reported by the controller board at frequent intervals this will indicate that the room is occupied and alarm will remain in unarmed state.

Once predicted that the room is unoccupied the server will turn the state of alarm to *Warning* mode. The *Warning* mode is basically an alert signal in form of buzzer tone. In case if sensor missed to detect presence then this signal will create sound to alert user present in the room that system is going in alarm mode. The user can trigger the sensor to reset the system back to its *Normal* state. The resetting in the algorithm is right now done by triggering sensor but for real system should be bound to some type of authentication. For example, the user can reset the state through his

computer or by tapping his RFID to the reader. Additionally, counter sensors can be used to reduce the triggering of false warnings. This solution will of course have impact on the cost of the system.

However, if no one is in the room and warning period has been completed then alarm will be turned into the *Arm* mode. The lock will also be activated along with the arming of alarm to restrict the entry point. Now the system is autonomously armed which means if someone tires break into the premises alarm will be triggered.

In order to unlock the door user has to present his RFID and PIN to the reader. The information will be sent to the server where it will be verified by comparing it with stored credentials. If verified the door will get unlocked and along with that the state of alarm will be switched back to *Normal* mode. The access control system is designed in such a way that only authorized user can unlock the door using his credentials. In this way both the algorithm present on the boards and on the server work together to provide autonomous alarm and access control system. The flowchart of the server algorithm is shown in the figure below.

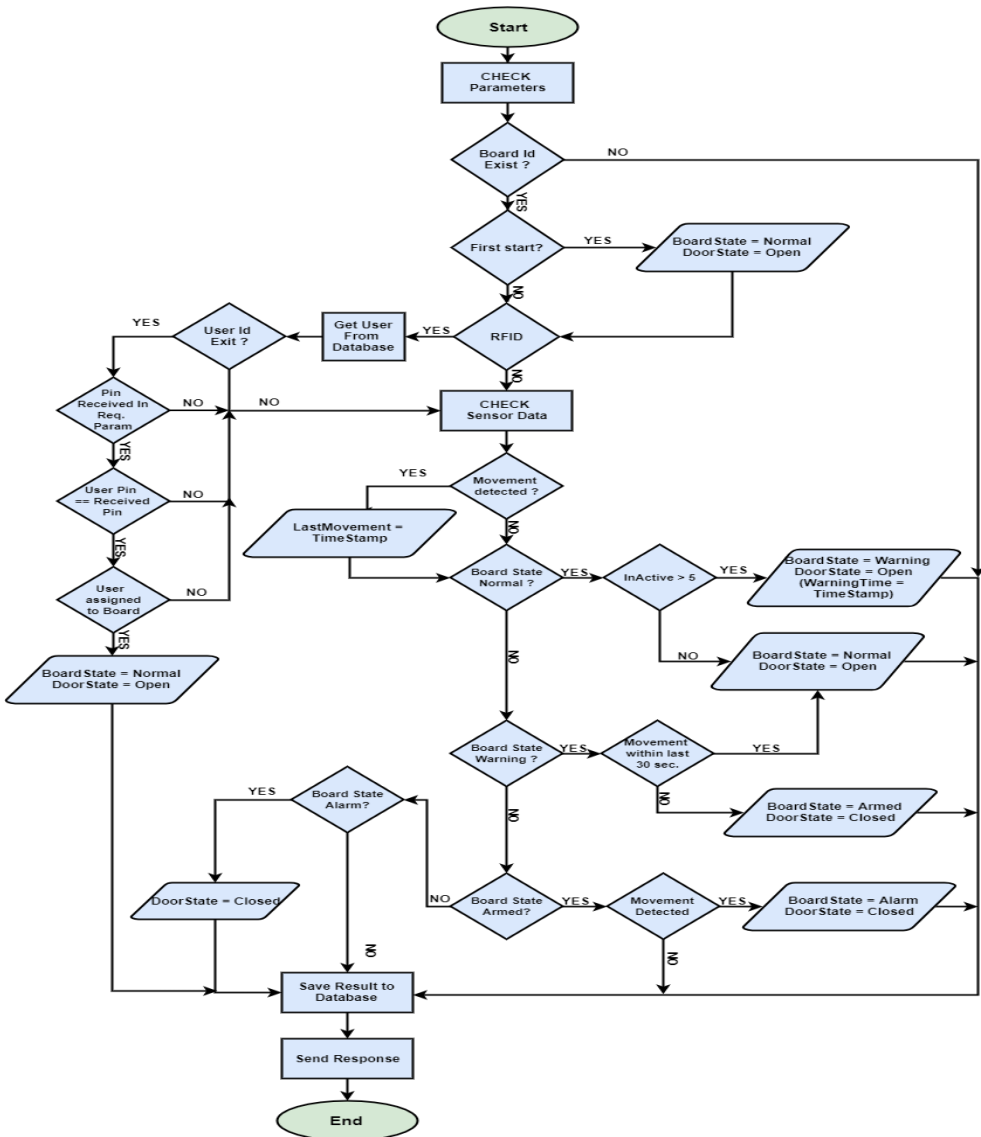


Figure 4.18: Flowchart of Server Software

4.2.4 Database

A database is used to log transactions of each system connected to the server. Apart from that it is used to manage controllers and user information. It provides the capability of adding new controller boards and users in the system. In order to

keep things simple, flexible and faster NoSQL MongoDB is used in the project to serve the purpose of storing and retrieving data. It consists of collections comprising of documents which is equivalent to rows in relational database. The document structure of MongoDB consists of fields and value pairs. The field value can contain other document or array of documents. It is not necessary for the documents to follow same structure each document can have different types of fields. The document store information in BSON format which is similar to JSON but has some additional data types. Based on the requirement of our design we have created three different types of collections.

Controller

The controller collection represents all the information about the controller boards. The layout of the controller document is shown in figure 4.19.

```
{
  "_id" : "1234",
  "ControllerType" : 0,
  "Name" : "new Room x.xx",
  "Location" : "Albertslund",
  "PortConfig" : [
    "N",
    "N",
    "N",
    "N",
    "N",
    "N",
    "N",
    "N",
    "N"
  ],
  "LastMovement" : NumberLong(0),
  "LastPing" : NumberLong(131480332696468595),
  "LastWarning" : NumberLong(131431535182139147),
  "ControllerState" : 3,
  "DoorState" : 1,
  "UpdateConfig" : true
}
```

Figure 4.19: Controller Collection Document

The *PortConfig* field defines what kind of sensor data should be read by the controller board and these setting can be changed during the runtime of the system which in turn will make controller board to adopt new configurations immediately.

UserVsController

This collection is used to create a relation between user and controller. The relation is that the user is assigned to the controller board. This is done by creating a document consisting of username and board serial number. The layout of the collection is shown in figure 4.20.

```
{
  "_id" : "db93bc76-7e39-4569-9d42-4288777eeb36",
  "Username" : "adeoOS",
  "SerialNumber" : "12345"
}
```

Figure 4.20: UserVsController Collection Document

User

The user collection consists of all the information about user credentials. The layout of collection is can be seen in figure 4.21.

```
{
  "_id" : "adeoOS",
  "Password" : "1234",
  "rfID" : "1624659",
  "Pin" : "1624"
}
```

Figure 4.21: User Collection Document

In the prototype as we have to do lot of testing because of which the data is being stored in the clear text form but for the final product the precautionary measures should be taken like hashing the data before storing it in the database.

CHAPTER 5

Implementation

A prototype is developed as a proof of concept in order to validate and evaluate our design. The functionalities of the prototype are developed according to the highly prioritize requirements set by the company. The chapter will go through the process of explaining how different components of the system are implemented and connected together to build our prototype that can cope with the problems discussed in the analysis chapter.

5.1 Software Tools

Variety of tools were used for the development of prototype. The first software used was *Proteous* to verify connections of the electrical components safely. A *Proteous* software is a simulator which helps to draw and diagnose electrical connections of the hardware devices on the software. As we were dealing with high voltage so it is always preferable to verify connections using simulator rather than directly working on the real electrical components.

In order to program the controller board firmware two development tools were used. initially, the work was carried out using *Arduino IDE* but later realized that there is a plugin available for *Visual Studio* known as *Visual Micro* which supports development of controller boards firmware. The plugin provides the possibility of using rich features of *Visual Studio* like debugging and intelligence which was limitation in *Arduino IDE*.

For the server side programming *ASP.NET WEB* APIs framework was used which helped to build WEB APIs in an easy and flexible manner. The mobile app was developed using *Cordova platform* which helped us to build a mobile application that can run on multiple platforms.

Last but not least, the programming languages used for making firmware, server side program and app were C++, C#, HTML, CSS and JavaScript respectively.

5.2 Interfacing Electric Lock with Controller Board

As stated in the design chapter that the power wires of the electrical lock are connected to the relay module which is powered using 12 volts. Now in order to interface the lock

we actually have to interface the relay module. This is done by connecting the relay module input pin $+$ with the digital pin $d5$ of the Controller Board. The digital pin means that the controller board can be programmed to either output HIGH (3.3V) or LOW (0 V) on that particular pin. This signal will be then read by the relay module and lock will be operated accordingly. The settings of the controller pins that are specified in the program to operate relay are shown in the figure 5.1.

```
pinMode(5, OUTPUT);    // Setting pin number 5 as OUTPUT
digitalWrite(5,HIGH); // The pin 5 will generate HIGH (3.3V) Signal
digitalWrite(5, LOW); // The pin 5 will generate LOW (0V) Singal
```

Figure 5.1: Relay activation and deactivation setting in Firmware

The first statement is used to set the Data Direction Register of pin number $d5$ which means that the board will reserve this pin for only producing outputs. The second two statements depict how we can program HIGH or LOW signals at the pin $d5$. The other part of the lock is interfacing feedback sensor consisting of yellow and white wires. The feedback is simply a 3.3(V) signal provided by the board to one of the sensor wire and waiting for the signal to return from the other wire back to the board. This signal will only be returned if the door is closed and lock pieces are aligned together. In order to do that we have to configure Data Direction Register of pin $d7$ to input. It can be done in the program as shown in the figure 5.2.

```
pinMode(7, INPUT_PULLUP); // set pin to input
digitalWrite(7, HIGH);    // The input will remain HIGH by default
```

Figure 5.2: Lock Feedback Programming

The first line sets the Data Direction Register of pin $d7$ but one thing to note down is that we are using `INPUT_PULLUP` rather than `INPUT` this is because when the pin is set to input it turns into a floating state. Floating state means that it can randomly accept any input generated from the noise in the environment and will produce unexpected results. Therefore, in order to avoid that we fix its value to HIGH and whenever it will be lowered we will treat that as an input signal.

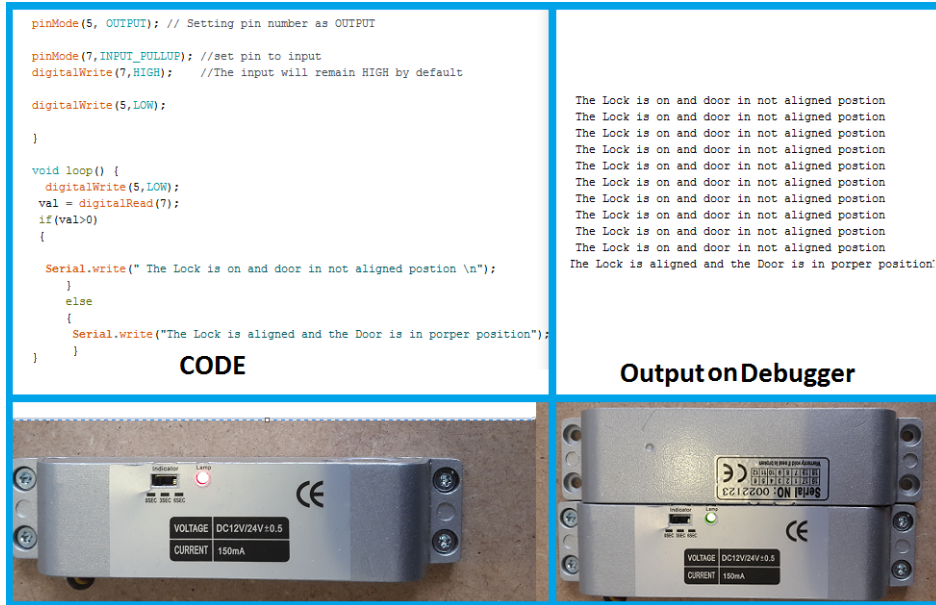


Figure 5.3: Lock Program Demonstration

The figure above shows that when the lock and the metal piece are not aligned together it is giving us negative feedback. As soon as the lock and metal piece are aligned the door is locked and we can see that on the debugger window. It basically means that the microcontroller has acknowledged the door is properly locked.

5.3 Interfacing Reader with Controller Board

The reader is connected to the special pins of the controller board i.e *d2* and *d3*. The reader is interfaced using the library *wiegand* [71] present at github. The *d2* and *d3* are interrupt based pins which means when they are triggered the program halts its activity and facilitate the data present of those pins first. The interrupt pins are attached to the reader and mimic the wiegand protocol. The pins are pulled HIGH and whenever the data is sent at the particular pin of controller from the reader that pin is pulled low and this action is logged. In the end total bits are combined to produce readable input data. The code for reading the data is shown in the figure 5.4.

```

#include <Wiegand.h> //Library

WIEGAND wg;

void setup() {

    wg.begin(); // intialization
}

void loop() {
    if(wg.available()) //if data is available show
    {
        Serial.print("Wiegand HEX = ");
        Serial.print(wg.getCode(),HEX); //Show data as Hexadecimal
        Serial.print(", DECIMAL = ");
        Serial.print(wg.getCode()); //Show data as Decimal
    }
}

```

Figure 5.4: Wiegand Controller Code

The `Serial.print()` instruction is used to send ASCII characters serially to the computer so that the output can be viewed on the computer screen. However, we have created a function `CheckReader()` which can capture the data from the RFID tag and Keypad in decimal and hexadecimal format.

5.4 Interfacing PIR Sensor with Controller Board

In order to interface the PIR sensors first we have to look at the schematic diagram of our sensor design shown in the figure 4.13.

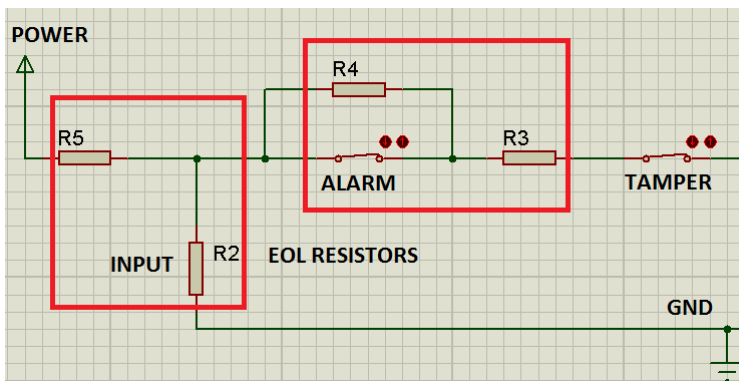


Figure 5.5: PIR sensor Schematic Diagram

We have three wires coming out of the PIR sensor namely *POWER*, *GND* and *INPUT*. The *POWER* and *GND* pin are connected to provide a signal of 3.3(V).

Afterwards, the third pin *INPUT* is used to read analog values coming out from the sensor. Basically these values represent four different states of the sensor that is open / tamper, wire cut, motion and no motion. Analog values are simply alternating voltage values like 1.5(V), 2.5(V) etc. The controller board has an Analog to Digital converter circuitry inside to read analog values and convert it into numerical values between 0 to 1023. The value 1023 is because of the 10 bit register of controller used in analog to digital conversion. The formula for conversion is provided by the board manufacturer.

$$\frac{\text{ResolutionOfADC}}{\text{SystemVoltage}} = \frac{\text{ADCReading}}{\text{AnalogVoltageMeasured}}$$

The *Resolution Of ADC* is 1023 and *System Voltage* is the voltage of our controller board. Whereas *ADC Reading* is the analog input value we feed to the pin and the *AnalogVoltageMeasured* is the numerical values we get within the range of 0 - 1023. There are seven *A0-A7* analog pins provided by the board where we can attach our PIR sensors. Once the circuitry is made the next step is to setup the code in order to specify meanings of the captured input values. The Arduino library provides built in function known as *analogRead()* to read the values from the sensor.

```

char checkSensor(int sensorInput)
{
    //Serial.println(sensorInput);

    char state;

    int sensorReading = analogRead(sensorInput);
    //logfile(String(sensorReading));
    if (sensorReading < 400) {
        state = 'W'; // Wire shorted. Possible tampering.
    }
    else if (sensorReading >= 400 && sensorReading < 590) {
        state = 'N'; // Normal state, sensor not triggered
    }
    else if (sensorReading >= 590 && sensorReading < 800) {
        state = 'T'; // Sensor triggered.
    }
    else {
        state = 'C'; // Open circuit. Cut or tamper triggered.
    }
}

```

Figure 5.6: Interfacing PIR Sensors Code

A function *checkSenors()* is created by us where we pass the analog value and in return we get single character value. The meaning of each character is specified in the comments of the program.

5.5 Interfacing Buzzer with Controller Board

The buzzer is used to make loud sound when the alarm is triggered. For our project we are using buzzer built in the reader. The interfacing of buzzer is quite easy as reader has one specific wire that outputs 5(V) all the time. We just need to pull it LOW to turn on the buzzer. The pin *d6* of the board is used to activate and deactivate the buzzer.

```

739 void Beeper()
740 {
741     {
742
743
744     if (beeper)
745     {
746
747         if (millis() - beepDuration >= beepInterval) // Change state after every specified time
748         {
749             beepDuration = millis();
750
751             if (beeperState == LOW)
752             {
753                 beeperState = HIGH;
754             }
755
756             else {
757
758                 if (beepInterval > 10 )
759                     beeperState = LOW;
760
761             }
762
763             digitalWrite(6, beeperState); // Fluctuating the output in order to play sound at different interval
764
765         }
766     }
767 }

```

Figure 5.7: Interfacing Buzzer with the Controller

The *Beeper()* function is made to control the buzzer. A timing delay have been added in the program to play sound at various intervals.

5.6 Interfacing LED with Controller

The LEDs are used to show the status of the system. It helps to indicate whether system is operational or have been stopped due to some problem. This type of visual feedback can help the owner to quickly identify the problems with the system. In order to interface an LED we just have to simply power it up. This can be done using any digital pin of the controller board. In our case we have created a *Blink()* function and is using the pin *A5* of the board set as digital output to control the LED.

5.7 Connectivity

In order to keep things simple we are using wired internet connection. The library *Ethernet.h* is provided by the Arduino community to connect the board to the internet. In the program, a function is declared to call DHCP server to assign an IP address

to the board. Additionally, IP Address and port number of the server is also defined in the firmware in order to exchange data between the board and server. Once all of the data is gathered by the board from the connected devices, the next step is to create HTTP request to send data to the server. In the current design POST method is used to send all the data present at the board. Figure 5.8 shows how to create HTTP request using Post method in the firmware.

```
Serial.println("connecting...");
if (client.connect(server,8080)) {
  Serial.println("connected");
  client.println("POST /api/board HTTP/1.1");
  client.println("Host: 10.0.0.35:8080");
  client.println("Connection: close");
  client.println("Content-Type: application/x-www-form-urlencoded; charset=utf-8");
  client.print("Content-Length: ");
  client.println(PostData.length());
  client.println();
  client.print(PostData);
  client.println();
}
else {
  Serial.println("connection failed");
}
}
```

Figure 5.8: Creation of HTTP Request in the Firmware

The *PostData* is actually the body of the HTTP request consisting of values like *PORTDATA*, *RFID*, *PIN*, *SerialNumber* which can be processed by the server. In the end last function named *EvaluateReturnValues()* is created which parse the response wrapped in the form of tags. Each tag consists of message for all the components connected to the board. The structure of the function *EvaluateReturnValues()* is shown in the figure 5.9.

```

516 void EvaluateReturnValues()
517 {
518     // Find config
519     char newSetting[8];
520
521     String doorStatus = GetValue("doorStatus"); // get keyword doorStatus from the response
522     String state = GetValue("state"); // get keyword state from the response
523     String tempnewSetting=GetValue("config"); // get keyword config from the response
524     String pinStatus = GetValue("pin"); // get keyword pinStatus from the response
525
526     logfile("Check out the Status");
527     logfile(pinStatus);
528
529
530     tempnewSetting.toCharArray(newSetting, 8);
531     setupPorts(newSetting);
532
533     Serial.print(state);
534
535     if (state == "Normal"){ ... }
541
542     if (state == "Warning"){ ... }
562
563     if (state == "Armed"){ ... }
569
570     if (state == "Alarm"){ ... }
577
578     if(doorStatus == "Open"){ ... }
583
584     if (doorStatus == "Closed"){ ... }
588
589
590     if (pinStatus == "On"){ ... }
---
```

Figure 5.9: Structure of EvaluateReturnValues() Function In Controller Firmware

The collected keywords are compared with *if* statements. When match is found that particular instruction is executed by the controller board. One important thing to highlight is that we have a *config* keyword that is used to change the settings of the controller board without uploading new firmware. This is because we have defined all the possible inputs and outputs of the controller board so that we can change the pin settings dynamically in real time.

5.8 Central Server

The server interface consist of two simple web APIs created using ASP.NET WEB API project. In the current version of the prototype two some simple calls are created to facilitate the board and mobile user clients separately.

API	Description
POST api/boards	Consumed by the controller board
POST api/RClient	Consumed by the mobile App

Table 5.1: HTTP Methods To Consume Resource

As we are going to receive plenty of valuable data from various clients for processing. Therefore, the first step is to setup the database for the storage of data. In the code we need to create a `MongoClient` object which consists of connection to the database. Afterwards, we need to specify the name of our database. If it is not present the `MongoClient` will make one for us. Once the database is created we need to get the collections by specifying the name and if they are not present in the database new ones will be created automatically. The steps needed to setup the *Mongo* database in *Visual Studio* are shown in the figure 5.10.

```
MongoClient client = new MongoClient("mongodb://localhost");
database =client.GetServer().GetDatabase("AdeoAlarm");
CollectionControllers = database.GetCollection<Controller>("Controllers");
CollectionUsers = database.GetCollection<User>("Users");
CollectionUserVsController = database.GetCollection<UserVsController>("UserVsController");
```

Figure 5.10: Setting Up Mongo Database in Visual Studio

5.8.1 Boards Class

Once the database is set then we need to create a class which can handle HTTP requests, define methods and return response to the clients. For the controller boards a class named boards is created which also specifies the name of the path and inside it POST method is implemented . In this method all of the calculations are taking place for the controller boards. Once http request is received from the board the parameters enclosed in the request body are read and copied for processing. The following steps are taken before returning the response back to the client.

5.8.1.1 Controller Board Identification

The first important parameter received is serial number of the board. The program compares the received serial number with the ones stored in the database. If match is found then further received parameter are evaluated.

```
// Query Database with the received serialNumber
Controller controller = WebApiApplication.CollectionControllers.FindOneById(serialNumber);
```

Figure 5.11: Searching Database with Board ID

5.8.1.2 Initialization of New Board

The second parameter is the *firststart* which is received only when the board starts for the first time. This is used to give some time to the board so that it can get stable and have proper configuration.

5.8.1.3 Authentication

The two parameters *RFID* and *pin* are used for authentication but are only received when the user tries to interact with the reader. So, if the data is present in these parameters then it is compared with the stored ones in the database to find if the ID is legitimate or not. One verified the next step is to see if the user is authorized to the room or not. If everything goes smooth the access is granted.

```

if (user.Pin == RequestPin) //if pin is accepted
{
    // Check if user is registered with the board
    var qr = Query.And(Query<UserVsController>.EQ(us => us.Username, user.Username.ToString()) ,
        Query<UserVsController>.EQ(us => us.SerialNumber, serialNumber));
    UserVsController usr = WebApiApplication.CollectionUserVsController.Find(qr).FirstOrDefault();

    if (usr != null) //user and board match found
    {
        controler.DoorState = Enums.DoorState.Open; //Open the door
        controler.ControllerState = Enums.ControllerState.Normal; // Move the PIR to nomarl state
        WebApiApplication.CollectionControllers.Save(controler); //Save the trasction in DB
    }
    else
    {
        // Pin Ok but not authroize to the controller room board
    }
}

```

Figure 5.12: User Authentication

5.8.1.4 Sensor Data Evaluation

The *portInfo* parameter is compared with several statement to see the status of the room. If we receive the letter *P* it means PIR triggered and we note down this transaction as the *LastMovement time* in the room. Apart from that we track if someone tried to sabotage the sensors.

```

if (PortInfo[i] != 'N')
{
    switch (controler.PortConfig[i])
    {

        case "P":
        {
            controler.LastMovement = nowStamp; //Motion Deceted
            break;
        }
        case "B":
        {
            controler.LastMovement = nowStamp;
            //Open door either be returning a open command or a portId to go high
            break;
        }
        case "W":
        { //Wire cut
            controler.LastMovement = nowStamp;
            controler.ControlerState = Enums.ControlerState.Alarm; //set Alarm
            break;
        }
        case "T":
        { // Tamper
            controler.LastMovement = nowStamp;
            controler.ControlerState = Enums.ControlerState.Alarm; //set Alarm
            break;
        }
    }
}

```

Figure 5.13: Sensor Data Evaluation at Server End

If none of the condition satisfies then it is considered as no motion. In this scenario only present time is logged and *LastMovement time* is not updated.

5.8.1.5 Setting Alarm Status

The second last step is to set the state of an alarm system. This is done by comparing the *Present time* with the *LastMovement time* stored in the board profile. As mentioned earlier that the *LastMovement time* is only recorded if motion is detected in the room. The boards are normally set to report back their room status after every 15 seconds if no motion is detected. This means that if we get regular calls of no motion detection, the *LastMovement time* variable will not be updated by the system. Thus, the time difference between *Preset time* and *LastMovement time* will increase. Then all we need to do is to set a suitable threshold (in our case is five minutes) to confirm that there is no movement in the room and it is time to safely change the state of the alarm to warning or armed mode. We have enclosed these calculation in the class named *APIFunctions.cs* in our code.

5.8.1.6 Response

The final step is to save newly generated results in the profile of the controller board present in the database and then send back the response to the client board. The response consists of tags with the keywords representing orders generated by the server for the controller board like open/close door, set alarm mode and configuration updates. This flow of program keeps the system running autonomously in real time.

5.8.2 RClients

The RClient class is used to handle HTTP requests coming from the mobile App clients. This class also consists of one POST method that takes four parameters i.e. username, password, serial number and action. In the current prototype to speed up the process everything is wrapped up into the single POST method but in future separate method would be implemented to avoid wasting additional processing cycles. The action parameter actually specifies what kind of resource the user wants to consume. The main functions of the RClient Class is as follows.

5.8.2.1 Login

When the user opens up the mobile app he is presented with the login page where he needs to enter his username and password. Once he enter his credentials the request is sent to the server. The server compares the parameter with the ones stored in the database for authentication. On successful authentication the user is logged in where he get access to his menu screen.

```
//checking ID
var user = Global.CollectionUsers.FindOneById(userId);

    if(user.Password==password)    // if ID exist the next step is to check passowrd
{
    responseLogin.MessageId = "100";
    responseLogin.MessageText = "UserOk";
    string response = JsonConvert.SerializeObject(responseLogin); // Covertng .NET Objects into JSON format.
    Response.Write(response);
}
}
```

Figure 5.14: Highlighting main events in mobile user authentication

5.8.2.2 Rooms Retrieval

The class is responsible for handling requests related to the retrieval of room list. The request with action parameter *getControlers* is sent to the server from the mobile client. The server queries all the controller boards serial number present in the

database associated with the username and make an object consisting of found rooms. Once found the .NET object is serialized into the JSON string format using Newton-JOSN framework. Afterwards, the response is sent to the APP.

```

case "getControllers":
{
    ResponceGetController responceGetController = new ResponceGetController();
    responceGetController.Rooms = new List<Controller>();
    var query = Query<UserVsController>.EQ(us => us.Username, userId);
    var listOfController = WebApiApplication.CollectionUserVsController.Find(query);
    foreach (var userVsController in listOfController)
    {
        var controller = WebApiApplication.CollectionControllers.FindOneById(userVsController.SerialNumber);
        responceGetController.Rooms.Add(controller);
    }
    string response = JsonConvert.SerializeObject(responceGetController);
    return response;
}

```

Figure 5.15: Code to provide list of room to the mobile app users

5.8.2.3 Arming and Disarming the Alarm System

The last two actions are *ARM* and *DISARM* which are used to arm or disarm the room linked to the user. The disarming is done by simply changing the time difference between *Present time* and *LastMovement time* to zero and changing the alarm state to Normal. Whereas, the arming of alarm is done by simply increasing the time difference.

```

case "DISARM":
{
    var controller = Global.CollectionControllers.FindOneById(sno);
    if (controller != null)
    {
        controller.LastPing = DateTime.UtcNow.ToFileTime();

        if (controller.ControllerState != Enums.ControllerState.Normal)
        {
            controller.ControllerState = Enums.ControllerState.Normal;
            controller.DoorState = Enums.DoorState.Open;
            controller.LastMovement = Now();
            Global.CollectionControllers.Save(controller);
        }
    }
    break; }

```

Figure 5.16: Statements to DISARM the room

5.9 Mobile Application

A simple mobile application is developed using Evothings Studio Workbench which provides platform for developing IoT based mobile applications. The platform enables one to build mobile applications using popular programming languages like HTML, CSS and JavaScript. It is a cross platform development tool which means that apps written in one programming language can run on multiple platforms. The mobile application will allow smart phone users to remotely arm and disarm the controller boards of their office. The application consists of two .html files; one is the index.html and other one is the menu.html. The application starts by loading the first index.html file which is basically a Log-in page as shown in the figure5.17.

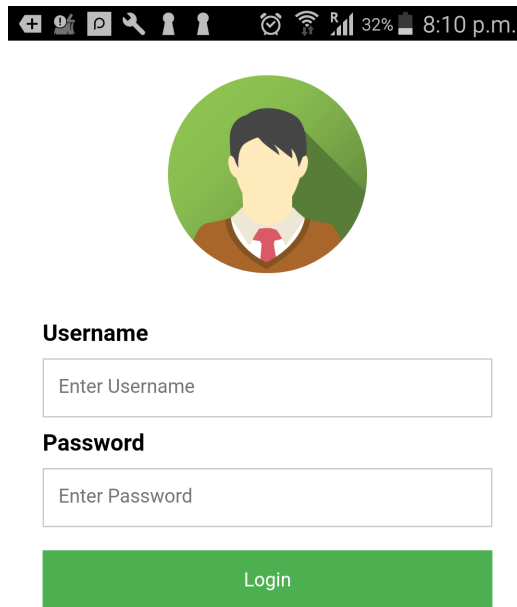


Figure 5.17: Mobile App Login Page

At this page the user is asked to enter his credentials in order to get access to the menu page. Two input tags are created one for entering username and other one for password.

```

<div class="container">
  <label><b>Username</b></label>
  <input id="us" type="text" placeholder="Enter Username" name="uname" required>

  <label><b>Password</b></label>
  <input id="ps" type="password" placeholder="Enter Password" name="psw" required>

  <button type="button" onClick="return CheckLogin()" >Login</button>

</div>

```

Figure 5.18: Capturing user input in .html

Once credentials are entered, the user can press the *Login* button present at the bottom of the text fields. The login button invokes the JavaScript function known as *CheckLogin()* where the processing takes place. The first thing defined in the function is to check the input fields to ensure that they are not empty. If text field is empty the user is notified via alert message.

```

if( document.getElementById("us").value.length == 0 || document.getElementById("ps").value.
length == 0 || document.getElementById("us").value.trim().length == 0 || document.getElement
ById("ps").value.trim().length == 0 )
{
  alert("Please enter User/Pswd and white spaces not allowed in text field");
  return false;
}

```

Figure 5.19: Checking User Inputs

However, if the text fields are properly filled than the HTTP request consisting of POST method is sent to the server using AJAX. AJAX decouples the data exchange layer from the presentation layer because network operation are time consuming and if executed on same layer can ruin user experience. In order to create HTTP request using AJAX first we need to create *XMLHttpRequest()* Object and then define the request structure.

```

var xhttp = new XMLHttpRequest(); // Creating new XMLHttpRequest Object
xhttp.open("POST", "http://10.0.0.35:8080/api/RClient", true); //Post Method with URI Path
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); //Headers
xhttp.send("userID="+document.getElementById("us").value + "&password="+document.getElementById("ps
").value+"&action=login"); //Request Body

```

Figure 5.20: Creating post request using AJAX

The server performs the authentication and send back the results. A callback function is created to fetch the data whenever the response is received at the client side. If the user is authenticated then he is transferred to the new menu.html page and if the provided credentials proves to be wrong then an error message is displayed on screen.

```
xhttp.onreadystatechange = function() { //CallBack Function
  if (this.readyState == 4 && this.status == 200) {
    result = this.responseText;
  }
}
```

Figure 5.21: Callback Function

The menu page consists of one single button present at the bottom of the APP. This button is used to retrieve the list of rooms assigned to the user.



Get Room List

Figure 5.22: Menu Page

When user presses the *GET Room List* button HTTP request is sent to the server and in response list of rooms is received in JSON string format. The received data is parsed into the JavaScript object using *JSON.parse()* function. Now based on the number of rooms received a small list of *DIV* is created dynamically for each room inside the *MAIN DIV*. The div consists of information about the controller board i.e serial number, alarm status and buttons which user can use to change the status of the alarm. Note that the visualization of serial number is just for testing process whereas in real App it should be replaced by room name.


```

for (var i = 0; i < jsn.Rooms.length; i++) { //loop based on rooms
//received
//alert(jsn.Rooms[i].SerialNumber);
//jsn=JSON.parse(res);
//alert(jsn.Rooms[i].SerialNumber);

    $('<div/>', {
        id: jsn.Rooms[i].SerialNumber,
        class: 'room'
    }).prependTo('#MAIN');

    if(jsn.Rooms[i].ControlerState == 1) //Translating numbers to user
    //understandable format
    {
        state="Normal";
        btn1="ARM";
    }

    else if(jsn.Rooms[i].ControlerState == 3)
    { state="ARMED";
      btn1="DISARM";
    }

    else if(jsn.Rooms[i].ControlerState == 2)
    { state="Warning";
    }
    else{}

    //Updating the newly created div contents
    document.getElementById(jsn.Rooms[i].SerialNumber).innerHTML="<p>Board
Number :"+jsn.Rooms[i].SerialNumber+ '<button class="button1"
type="button" value="+btn1+ ' onclick="+ARM("+jsn.Rooms[i].SerialNumber+',
'+"\\""+btn1+"\\"" +' ) id=m'+jsn.Rooms[i].SerialNumber+ '>'+ btn1+ '</button>'
+ "<br>Controller State:"+state+"</p>";

```

Figure 5.23: Creating list of div representing rooms

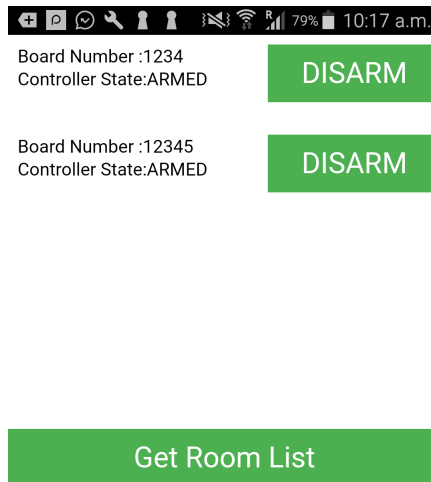


Figure 5.24: User room list of the menu page

Now the user can select which room he wants to ARM or DISARM remotely based on the buttons presented on the list. The buttons consist of function call with

parameters i.e serial number and action to be performed. When the button is pressed these parameter are passed to the function named *ARM()* which makes an HTTP request to the server to perform the specified action on the controller board. Once the specified action is performed the server sends response back to the client which in turn refreshes the list with new information. In this way using simple mobile App the user can control their office rooms remotely.

CHAPTER 6

Evaluation

In this chapter we are going to evaluate our developed prototype against the requirements listed in the analysis section. This evaluation will help us to verify the functionality and prove the feasibility of the system. The overall prototype developed consisting of alarm and access control components is shown in the figure 6.1.

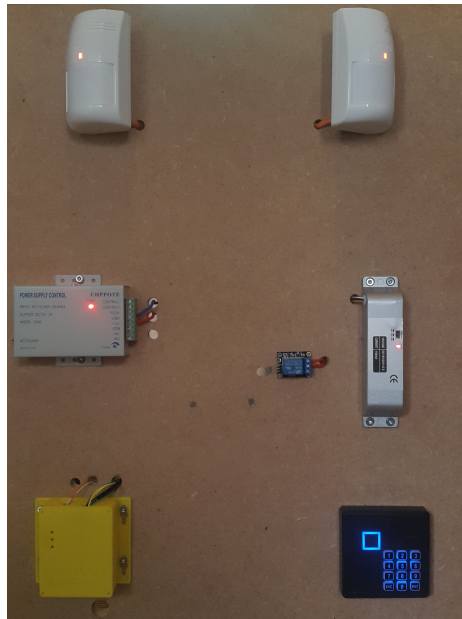


Figure 6.1: Hardware Prototype

6.1 Autonomous Alarm System

Autonomous alarm system has been fully developed and satisfy all of the requirements listed in the table. The proof for completion of each requirement is discussed below.

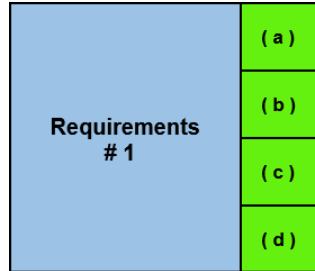


Figure 6.2: Requirements Number 1

6.1.1 RQ # 1(a)

Each room has its own controller board working independently from other controllers in the system. The system sets the alarm autonomously in each room. The stages through which the system passes before setting the alarm is shown in the figure below.

Stage 1 Normal

```
"<config>PNNNNNNN</config><doorStatus>Open</doorStatus>
<state>Normal</state><lastmovement>00:00:00</lastmovement>
<lastPing>00:00:00</lastPing><lastWarning>00:33:32.7773533
</lastWarning>"
```

Stage 2 Warning

```
"<config>PNNNNNNN</config><doorStatus>Open</doorStatus>
<state>Warning</state><lastmovement>00:05:00.7648332</lastmovement>
<lastPing>00:00:00</lastPing><lastWarning>-00:00:00
.0005036</lastWarning>"
```

Stage 3 Arm

```
"<config>PNNNNNNN</config><doorStatus>Closed</doorStatus>
<state>Armed</state><lastmovement>00:05:31.7225925</lastmovement>
<lastPing>00:00:00</lastPing><lastWarning>00:00:30
.9572557</lastWarning>"
```

Figure 6.3: Response received by Controller Board

In the *Normal Mode* the tag `<doorStatus>` shows that the lock is open and `<state>` of the alarm is *Normal*. Whereas, in *Warning Mode* the `<state>` of an alarm is set to *Warning*. This is because the tag `<lastmovement>` shows that five minutes have been passed since last motion is detected. Therefore, *lastWarning* clock will be triggered to start counting. Once it reaches thirty seconds and still no motion is detected the `<state>` of the alarm will be changed to final mode known as *ARM*. Now if someone tries to walk into the room then alarm will be triggered. All of these stages working together helps to set the alarm autonomously.

6.1.2 RQ # 1(b)

This part is optional because it was not defined in the project scope but was later decided by the company officials to be added in the project. However, our design is flexible and programming it to arm the system by a simple button press is not a problem. We just need to set one of the pin in configuration file to accept button triggers. Afterwards, the code to handle the event is already present in the current design.

```

case "B": //Button Press
{
//ARM THE BOARD
if (controller.ControllerState != Enums.ControllerState.Armed &&
    controller.ControllerState != Enums.ControllerState.Warning)
{
//controller.ControllerState = Enums.ControllerState.Warning;
controller.LastMovement = DateTime.Now.AddMinutes(-6).ToFileTimeUtc();
controller.ControllerState = Enums.ControllerState.Armed;
controller.DoorState = Enums.DoorState.Closed;
WebApiApplication.CollectionControllers.Save(controller);
}
}

```

Figure 6.4: Arm system on button press

6.1.3 RQ # 1(c)

The occupancy is detected using the PIR sensors as they can track the motion within the room. In order to test the accuracy of PIR sensor four different test scenarios were conducted and output of PIR sensor against those scenario was checked. The results of the tests are shown below.

```

Board started
Pir1: 14: 512 (Normal state)
Pir1: 14: 508 (Normal state)
Pir1: 14: 533 (Normal state)
Pir1: 14: 507 (Normal state) No motion
Pir1: 14: 502 (Normal state)
Pir1: 14: 476 (Normal state)
Pir1: 14: 517 (Normal state)
Pir1: 14: 671 (Sensor triggered) Motion detected
Pir1: 14: 728 (Sensor triggered)
Pir1: 14: 1023 (Open. Cut or tamper sound alarm)
Pir1: 14: 1023 (Open. Cut or tamper sound alarm) Circuit Open
Pir1: 14: 1023 (Open. Cut or tamper sound alarm)
Pir1: 14: 339 (Wire short sound alarm)
Pir1: 14: 294 (Wire short sound alarm) Short Circuit
Pir1: 14: 267 (Wire short sound alarm)

```

Figure 6.5: PIR sensor test

Initially when there was no motion within the room the PIR sensor was displaying *Normal state* as expected. In the second scenario when movement was made in the room the PIR sensor was able to capture it properly. The PIR cover was removed

in the third scenario which lead to the change in the output of PIR as Tamper. In the end wires were shortened to dodge the sensor but PIR sensor was able to detect and report it. The events shown in the figure 6.5 and algorithm implemented helps in detecting whether someone is in the room or not.

6.1.4 RQ # 1(d)

The alarm system is interfaced with the buzzer which means that in case of intrusion the buzzer will be activated. The figure below shows how the ControllerState *Alarm* is treated in the firmware.

```
if (state == "Armed")
{

    beeper = false;
}

if (state == "Alarm")
{
    // Start beeping...
    beepInterval = 10;
    beeper = true;
}
}
```

Figure 6.6: Requirements

The *Alarm Mode* will be activated if the movement is detected in premises during the armed mode. As it can be seen from the figure that in the Alarm mode the buzzer is activated by playing sound at a continuous interval of 10 milliseconds.

6.2 Access Control System

The access control system developed is fully functional and meet all the requirements shown in the table below.

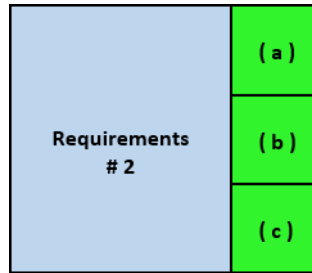


Figure 6.7: Requirements Number 2

6.2.1 RQ # 2(a)

A proximity reader with keypad is used which can be placed at the entry points of the premises for input. The user can present his credentials i.e (RFID tag and password) to the reader which are then used by the system for authentication. The figure shows the reader installed at the entry point of the CTO's office room.



Figure 6.8: Installation of Proximity Reader at one of the AdeoOS Office room

The debugger can be used to see the inputs coming from the reader. The figure below shows the inputs captured by the algorithms once the user present his RFID tag and PIN to the reader.

```

Serial | COM8
Opening port
Port open
RFID DATA: 8708681 triggered
Key Presses
1
Key Presses
12
Key Presses
125
Key Presses
1257
Final Pin Value
1257

```

Figure 6.9: RFID reader Test Results

Once RFID value is captured the keypad is activated so that the user can enter his PIN code. The key values are only accepted if provided along with the RFID tag otherwise they will be discarded.

6.2.2 RQ # 2(b)

The system allows access to only those members which are registered in the system to a particular room. The figure 6.10 shows how system allows the access to the authorized user.

Before

Key	Value
(3) 1234	{ 11 fields }
id	1234
ControlerType	0
Name	new Room xxx
Location	Albertslund
PortConfig	[8 elements]
LastMovement	131500557026155565
LastPing	13150056034381490
LastWarning	131500560033808933
ControlerState	3
DoorState	1
UpdateConfig	true

After

Key	Value
(3) 1234	{ 11 fields }
id	1234
ControlerType	0
Name	new Room xxx
Location	Albertslund
PortConfig	[8 elements]
LastMovement	131500557026155565
LastPing	131501286650982859
LastWarning	131501286655590926
ControlerState	2
DoorState	0
UpdateConfig	true

Debug Window:

```

string portString = data.portString;
string serialNumber = data.serialNumber;
string rfid = data.rfid;
string fin = rfid;
string RequestPin = data.RequestPin;

//RFID comparision
var query = Query<User>.EQ(lo => lo.rfid, rfid);
//if pin is accepted
if (user.Pin == RequestPin)
{
    // Check if user is registered with the board
    var qr = Query.And(Query<UserVsController>.EQ(us => us.Username, user.Username.ToString()),
        Query<UserVsController>.EQ(us => us.SerialNumber, serialNumber));
    UserVsController usr = WebApiApplication.CollectionUserVsController.Find(qr).FirstOrDefault();

    if (usr != null) //user and board match found
    {
        controler.DoorState = Enums.DoorState.Open; //Open the door
        controler.ControllerState = Enums.ControllerState.Normal; // Move the PIR to nomarl state
        WebApiApplication.CollectionControllers.Save(controler); //Save the trasction in DB
    }
}

```

Figure 6.10: Debugging of User Authentication

The controller board profile is shown on the top left of the figure which indicates that the alarm is armed (*ControlerState = 3*) and the door is locked (*DoorState = 1*). The figure on the right shows the the server program which compares the received RFID and PIN value with the saved ones. If the user is authentic then the access is granted as it can be seen in the controller board profile shown on the bottom left of the figure. The door is unlocked (*DoorState = 0*) and alarm status has been moved to warning mode (*ControlerState = 3*). One thing to note down is that the status is moved to warning mode rather than the normal mode. This is because the state will go back to normal mode automatically when the user steps into the room and the PIR sensors are triggered. This is helpful in the case when user authenticate himself but for some reason doesn't enter the room. The room will be armed again after thirty seconds.

6.2.3 RQ # 2(c)

Unauthorized users will not get access to the premises. If someone enter his/her credentials at the entry point where he/she is not authorized the system will simply ignore it. The process of authentication goes through all the steps mentioned earlier but in the end when the user is not found the systems simply ignore it by going into the *else* statement.

```
else
{
    // Pin Ok but not authroize to the controller room board
}
```

Figure 6.11: User is valid but not authorized

More functionalists can be implemented like restricting or blocking the user after several attempts.

6.3 Firmware

The firmware is implemented according to the design and performs all the functionalities listed in the requirements section smoothly without any major issues. However, few strange data values were seen in the debug window but luckily those values never appeared inside the program code. We concluded that this might be a problem with the debugger itself but not the firmware code.

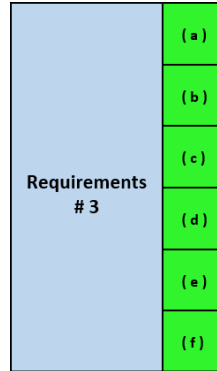


Figure 6.12: Requirements Number 3

6.3.1 RQ # 3(a)

The components of the alarm and access control system were interfaced with the board one by one and were checked regularly to ensure the proper functionality of the system. In the end all of the interfaced components worked together as a single unit. The prototype with interfaced components was placed inside the office room and used for several days to check the validity of the system. The figure below shows the installation of prototype in the AdeoOS office room.



Figure 6.13: Running prototype installed in the AdeoOS office room

The door was controlled with the electric lock and each day was locked and unlocked using the developed prototype. The system worked as it should and no problem

was faced in locking or unlocking the door. The only scenario in which the lock will not open is when the server is not responding. In order to overcome this issue fault tolerant solutions must be implemented. One of solution could be temporary creation of mini database in the EEPROM of the controller board. The controller board can then consult installed database in case of unavailability of the primary service.

6.3.2 RQ # 3(b)

The firmware performs the local processing inside the controller board before sending the data to the sever. This processing helps to ease down the load on the server by filtering out the useless data. For example, if there is no motion in the room we don't want this data to be sent to the server every second. As this data is of no interest and during the night huge amount of similar data will be generated. Therefore, non critical data is filtered out and sent after some time delay.

Another example is discarding the key presses if the user doesn't provide the RFID tag. As the system supports multi-factor authentication and sending only key presses to the server without providing RFID tag is going to result in failed authentication . Therefore, key presses can be discarded at the controller level to avoid invoking the server unnecessarily.

6.3.3 RQ # 3(c)

The response given back by the server on each call of the controller board consist of all the necessary instructions and configurations for that particular board. The example of the server response to the controller board is shown in the figure below.

```
"<config>PNNNNNNN</config> <doorStatus>Open</doorStatus>
<state>Normal</state><lastmovement>00:00:00</lastmovement>
<lastPing>00:00:00</lastPing><lastWarning>00:33:32.7773533
</lastWarning>"
```

Figure 6.14: Server Response Message

The important tags are highlighted in the red colored boxes. The first tag *<config>* is about the configuration settings of the micro controller. It consists of eight letters each of which represent pins *A0-A7* of the controller board. The pins can be configured to receive various forms of inputs like digital(Buttons), analog (Sensors) etc. The table below shows the letters indicating what kind of inputs can be accepted by our controller board.

Letter	Description
N	None which means whatever connected to PORT will not be read by the board.
P	PIR will only read inputs from motion detection sensor.
B	Button will only read input from the button connected on the particular pin.
M	Magnetic used to read inputs from reed switches.

Figure 6.15: Configuratuon Settings

Currently, we are using the first two letters whereas third and fourth one are implemented in the firmware but not used right now. The future plan is to use one for door bell and other for window sensors.

The next tag *doorStatus* is the instruction for the door lock. It is either *Open* or *Closed*. *Open* means that the controller should unlock the door whereas *Closed* means that controller should lock the door. The final important tag *state* is about the state of the alarm. The figure ?? shows the actions taken by the controller board depending on the different states provided by the server.

State	Action
Normal	The buzzer is kept off by the controller board and motion detected has no impact on it.
Warning	Activate buzzer to alert user that soon alarm is going into the arming state. If movement is detected the state goes back to normal.
Armed	Turn off the buzzer and look for any motion in the room.
Alarm	Turn on the buzzer to frighten the intruder and call the emergency service.

Figure 6.16: Alarm states and corresponding controller actions

To call the emergency service is not implemented in the firmware but is one of our future plan.

6.3.4 RQ # 3(d)

The firmware can control the door lock based on the instructions received from the server. The part of the code which controls the lock based on the instructions received from the server is shown in the figure below.

```

if(doorStatus == "Open")
{
    digitalWrite(5, HIGH)// Open the door
}

if (doorStatus == "Closed")
{
    digitalWrite(5, LOW);// Close the door
}

```

Figure 6.17: Door lock control in firmware

As the comments suggest that the HIGH signal will open the door and LOW will close the door.

6.3.5 RQ # 3(e)

The data received from the attached components is smoothly sent to the server by wrapping it into HTTP request body. The example of request body is shown in the figure 6.18.

```
postData = "serialNumber=" + sn + "&portString="+PortTransmit +"&rfid="+rfID+"&RequestPin="+pin;
```

Figure 6.18: Request Body

A simple test of client server interaction was conducted during the early development of the system where the system was left on running for one whole night. The number of successful calls completed and failed were counted in the test. The results are shown below.

Expressions on COM5			
Name	Date	Min	Max
<i>AdeoController.ino, line 539, httpRequest()</i>			
success	1938	1	1938
<i>AdeoController.ino, line 543, httpRequest()</i>			
failure	1	1	1

Figure 6.19: Successful and failed calls between client and server

Approximately 2000 calls were made out of which only 1 failed for some unknown reason. However, the overall system didn't crashed during the failed call.

6.3.6 RQ # 3(f)

There are two types of scenarios in which the system alerts the server immediately without causing any delay. First one is when motion is detected and another one is when input is received from the reader so that the user credentials can be sent for authentication. Both the scenarios are important due to which priority is given to them as compared to other situations.

6.4 Central Server

The server side software fulfills all the requirements listed in the analysis chapter and serves both the controller boards and mobile Apps connected on the network. The structure is scalable as new services can be easily added to server.

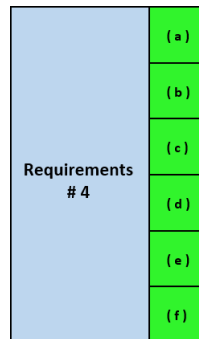


Figure 6.20: Requirements Number 4

6.4.1 RQ # 4(a)

The server is able to receive and send data from and to the clients respectively. As all the clients are depending on the instructions sent by the server therefore it is highly important that it should work nonstop. Hence, our server is able to communicate and exchange data with the clients without any hurdle.

As the server is a central unit which means it has a single point of failure. The only scenario in which the communication with the server will not be possible is when it is down or unavailable. Therefore, a monitoring service is needed using fault-tolerant techniques to protect the system from complete failure.

6.4.2 RQ # 4(b)

Each data received at the server end is carefully analysed and processed using an algorithm implemented for an autonomous alarm and access control system. After processing,

appropriate decisions are made and instructions are sent to the client.

6.4.3 RQ # 4(c)

The configuration settings for the controller boards are stored in their profile which is present in the database. These configuration settings can be changed in real time by simply editing it in the database. Each response sent from the server consists of configuration settings for the client. The first thing the client does on receiving response it to check for new configuration settings and adopt it accordingly.

The partial change of firmware over the internet is very powerful and flexible feature of our system. It was not been possible to implement it if the board had no Internet connectivity.

6.4.4 RQ # 4(d)

As shown in the implementation section that multifactor authentication mechanism consisting of PIN and RFID tag is implemented in order to provide and restrict access to the premises. The system has been tested by providing authorized and unauthorized credentials to ensure the proper functionality of the system.

6.4.5 RQ # 4(e)

The server controls the alarm state of each board individually. Class with the name *ApiFunctions.cs* is implemented to handle the alarm states of each controller board. A sample of database is shown in the figure 6.21 representing alarm states of two different controller boards set by the server.

Key	Value	Key	Value
<ul style="list-style-type: none"> ▼ (1) 12345 <ul style="list-style-type: none"> ▢ _id 12345 ▢ ControllerType 0 ▢ Name Room 0.01 ▢ Location Albertslund > PortConfig [8 elements] <ul style="list-style-type: none"> ▢ LastMovement 131443507354438302 ▢ LastPing 131495703143261708 ▢ LastWarning 131444597070880383 ▢ ControllerState 3 ▢ DoorState 1 ▢ UpdateConfig true 		<ul style="list-style-type: none"> ▼ (3) 1234 <ul style="list-style-type: none"> ▢ _id 1234 ▢ ControllerType 0 ▢ Name new Room x.xx ▢ Location Albertslund > PortConfig [8 elements] <ul style="list-style-type: none"> ▢ LastMovement 131500484592661614 ▢ LastPing 131500484592661614 ▢ LastWarning 131499588013601648 ▢ ControllerState 1 ▢ DoorState 0 ▢ UpdateConfig true 	
(A)		(B)	

Figure 6.21: Controller Profile

The *ControllerState* variables represents the state of an alarm system. In figure (a) *ControllerState* is *Armed* = 3 whereas in figure (b) the state is (Normal = 1). This

clearly shows that the server is capable of setting and managing the states of each connected board.

6.4.6 RQ # 4(e)

The data received by the controller board is always saved in the database before sending the response back to the client. The instruction used in the program to save the controller board profile is shown in the figure 6.22.

```
WebApiApplication.CollectionControllers.Save(controller);
```

Figure 6.22: Saving controller board profile in database

The storage of data is highly important because it is going to be used each time when server performs processing on data.

6.5 Mobile Application

A mobile application is developed which enables the smartphone users to ARM and DISARM the alarm of their rooms remotely. Two screenshots of the mobile App is shown in the figure??.

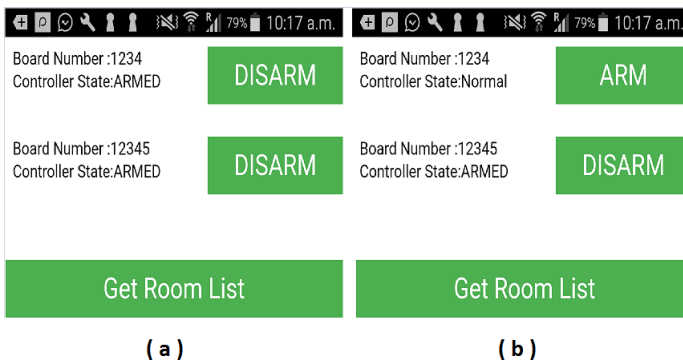


Figure 6.23: (a) Both the rooms are armed (b) user Disarm the first room

In the first screenshot the user is presented with the list of his rooms showing alarm status and board number. Whereas, the second screenshot shows refreshed list with new values. The new list is generated when the user tries to update the alarm state of a particular room by the pressing the corresponding button.

A login page is also created which appears before the menu page. Here the user have to present his credentials in order to get access to the rooms list. If a user is

logged in successfully he will be presented with the room list otherwise error will be displayed as shown in the figure below.

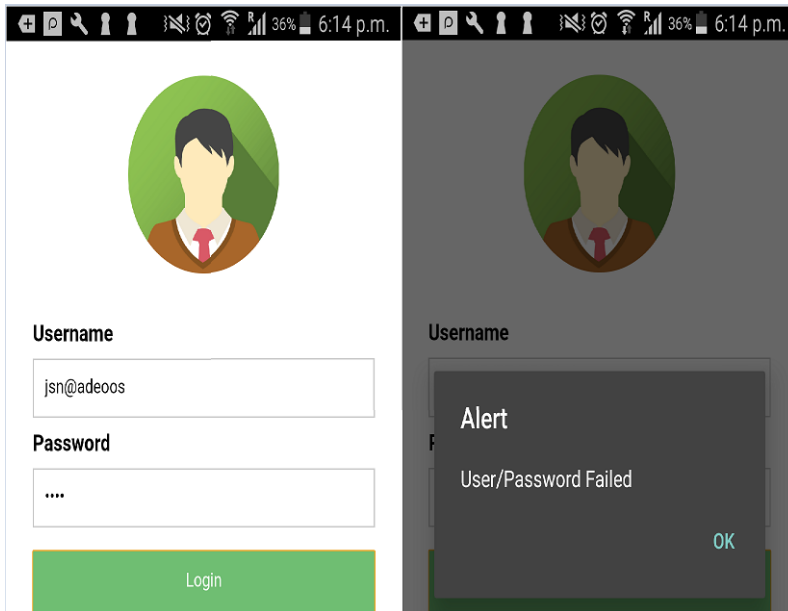


Figure 6.24: Login Failed

CHAPTER 7

Conclusion

The thesis project mainly focused on tackling the problems faced in arming the so called autonomous alarm system and controlling of access to the premises when multiple companies are working under one roof. During the initial phase of the project various technologies like alarm, access control, distributed and ubiquitous computing systems were researched that helped in finding a novel solution to cope with the challenges faced in the current alarm systems.

The theory studied was applied in practice which helped in building a reliable, scalable, efficient and most importantly cost effective solution. An IoT based Autonomous Alarm and Access Control System has been proposed which combines the power of internet to the alarm and access control modules. The IoT technology helped in enhancing the processing power and memory limitations of alarm and access control system which in turn made it possible to incorporate intelligent features into the system. These intelligent features made it possible to completely automate the alarm system which helped in reducing the rate of human error, false alarms and burglary.

The solution is a combination of hardware and software components; distributed alarm and access control modules, mobile App and a central server with database connecting every object in the system together. The alarm and access control modules are deployed in each room where they operate independently to other modules in the system. These modules can automatically arm and disarm their room alarm system based on occupancy and along with that they control access to the entry points of the room.

A prototype is developed as a proof of concept that is based on the proposed design. The prototype was installed in one of the office room of AdeoOS Aps for testing and evaluation purpose. It was capable of performing all of its functionalities that were necessary to achieve our goals and prove the feasibility of the system. Therefore, we can conclude that the aim of developing cost effective IoT based autonomous alarm and access control system is accomplished and the solution is capable of overcoming the limitations faced in most of the expensive security alarm systems present in the market.

7.1 Future Work

For the future, the plan is make number of improvements and enhancement in the current system to increase the accuracy and efficiency of the overall system. Some of

the ideas to work with in future are as following.

7.1.1 Counter Sensor

As it can be seen from our work that we used PIR sensors for occupancy detection as they were easily available and already installed in the office rooms. Whereas, it would be interesting to combine the PIR sensor along with the counter technology discussed in analysis topic. The combination will help to enhance the occupancy detection mechanism which in turn can reduce the false alarm rate and possibilities of dogging the system.

7.1.2 Fault tolerance

The proper functioning of the system is highly important and failure of such systems can result in major consequences. Therefore, it is really important to implement fault tolerant techniques so we can prevent the system from complete failure. The system designed should robust enough to keep the main functionality of the system running in case of faults. Some of the examples could be implementing watch dog timers in case communication with server fails or using EEPROM as a secondary database in case server is unavailable.

7.1.3 Security

Our project needs lot of improvements in terms of security. The data exchange and stored in the system is not encrypted or hashed. Therefore proper cryptographic algorithms should be used to protect our valuable data. Apart from that the security in IoT is major challenge. As the network security in the connected physical objects is weak and can be used as gateway by hacker to access and damage other connected devices in the network.

7.1.4 Administrator Application

An administrator application is needed so that the administrator can manage his alarm and access control system installed in the building. Right now all of administrative tasks like registering, specifying access privileges, delete profile etc. are all done by directly editing the data present in the database using Robomongo software.

APPENDIX **A**

An Appendix

A.1 Server Code

```
1
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Net;
6 using System.Net.Http;
7 using System.Net.Http.Headers;
8 using System.Web.Http;
9 using WebApplication9.Models;
10
11 using System;
12 using System.Collections.Generic;
13 using System.Configuration;
14 using System.Linq;
15 using System.Web;
16 using System.Web.UI;
17 using System.Web.UI.WebControls;
18 using WebApplication9.Models;
19 using MongoDB.Driver.Wrappers;
20 using System.Threading;
21 using MongoDB.Driver.Builders;
22
23 using Newtonsoft.Json;
24 using MongoDB.Bson;
25 using MongoDB.Driver.Linq;
26
27 namespace WebApplication9.Controllers
28 {
29
30
31     public class BoardsController : ApiController
32     {
33
34
35
36         public String Post([FromBody] Data data)
37         {
38
39             string portString = data.portString;
```

```
41     string serialNumber = data.serialNumber;
42     string rfID = data.rfID;
43     string firststart = data.firststart;
44     string RequestPin = data.RequestPin;
45
46
47
48     //Thread.Sleep(5000);
49
50     long nowStamp = ApiFunctions.Now();
51
52     char[] PortInfo = null;
53     if (portString != null)
54     {
55
56         PortInfo = portString.ToCharArray();
57
58     }
59     else
60     {
61         //return new string[] { "value1,value2" };
62         return "";
63     }
64
65     // Query Database with the received serialNumber
66     Controller controller = WebApiApplication.CollectionControllers.FindOneById(
        serialNumber);
67
68
69     if (controller == null)
70     {
71         return "";
72     }
73
74
75
76     controller.LastPing = nowStamp; // Always set last ping
77
78     if (controller.ControllerState == Enums.ControllerState.Undefined)
79     {
80         //Response.Write(ReturnData("config", string.Join("", controller.
81             PortConfig)));
82         //return; // Just return to the client, the controller is new, and
83             needs configuration.
84
85         return ""; //Not Used
86     }
87
88     if (firststart == "true")
89     {
90         if (controller.ControllerState != Enums.ControllerState.Normal)
91         {
92             controller.ControllerState = Enums.ControllerState.Normal;
93             controller.DoorState = Enums.DoorState.Open;
```

```
93         controller.LastMovement = nowStamp;
94     }
95 }
96
97
98
99 if (rfID != "" || rfID != null) //RFID comparison
100 {
101     var query = Query<User>.EQ(lo => lo.rfID, rfID);
102     User user = WebApiApplication.CollectionUsers.Find(query).
        FirstOrDefault();
103
104     if (user != null) // Used for Test
105     {
106         if (string.IsNullOrEmpty(user.Pin))
107         {
108             controller.DoorState = Enums.DoorState.Open;
109             controller.ControlerState = Enums.ControlerState.Normal;
110         }
111         else
112         {
113             if (RequestPin != null)
114             {
115
116
117
118                 if (user.Pin == RequestPin) //if pin is accepted
119                 {
120
121
122
123                     // Check if user is registered with the board
124                     var qr = Query.And(Query<UserVsController>.EQ(us =>
                        us.Username, user.Username.ToString()) ,
                        Query<UserVsController>.EQ(us => us.
                            SerialNumber, serialNumber));
125                     UserVsController usr = WebApiApplication.
                        CollectionUserVsController.Find(qr).
                            FirstOrDefault();
126
127                     if (usr != null) //user and board match found
128                     {
129
130
131                         controller.DoorState = Enums.DoorState.Open; //
                            Open the door
132                         controller.ControlerState = Enums.ControlerState.
                            Normal; // Move the PIR to nomarl state
133                         WebApiApplication.CollectionControllers.Save(
                            controller); //Save the trasction in DB
134                     }
135
136                     else
137                     {
138                         // Pin Ok but not authroize to the controller
                            room board
```

```

139         }
140     }
141     else
142     {
143         // Fail pin is not correct, ToDo 3 fails disable the
            user?
144     }
145
146
147     }
148
149     else
150     {
151         // Ask for the Pin
152         //Not Used AnyMore
153         return "";
154     }
155 }
156 }
157 }
158
159
160
161 // now check what to do.
162 for (var i = 0; i < PortInfo.Count(); i++)
163 {
164
165
166     if (PortInfo[i] != 'N')
167     {
168         switch (controler.PortConfig[i])
169         {
170
171
172             case "P":
173             {
174                 controler.LastMovement = nowStamp;
175                 break;
176             }
177             case "B": //Button Press
178                 //Arm the BOARD
179                 if (controler.ControlerState != Enums.ControlerState.
                    Armed &&
180                     controler.ControlerState != Enums.ControlerState.
                        Warning)
181                 {
182                     //controler.ControlerState = Enums.ControlerState
                        .Warning;
183                     controler.LastMovement = DateTime.Now.AddMinutes
                        (-6).ToFileTimeUtc();
184                     controler.ControlerState = Enums.ControlerState.
                        Armed;
185                     controler.DoorState = Enums.DoorState.Closed;
186                     WebApiApplication.CollectionControlers.Save(
                        controler);

```



```
187         return controler.ControllerState.ToString();
188
189     }
190
191
192
193     break;
194 }
195 case "W":
196 { //Wire cut
197     controler.LastMovement = nowStamp;
198     controler.ControllerState = Enums.ControllerState.Alarm
199         ; //set Alarm
200     break;
201 }
202 case "T":
203 { // Tamper
204     controler.LastMovement = nowStamp;
205     controler.ControllerState = Enums.ControllerState.Alarm
206         ; //set Alarm
207     break;
208 }
209
210
211 }
212 }
213 }
214
215
216
217
218 ApiFunctions.SetControlerState(controler);
219
220 WebApiApplication.CollectionControllers.Save(controler);
221 //controler.DoorState = Enums.DoorState.Closed;
222
223 return ReturnData("config", string.Join("", controler.PortConfig))+ (
224     ReturnData("doorStatus", controler.DoorState.ToString()))+(
225     ReturnData("state", controler.ControllerState.ToString()))+ (
226     ReturnData("lastmovement", new TimeSpan(nowStamp - controler.
227     LastMovement).ToString()))+ (ReturnData("lastPing", new TimeSpan(
228     nowStamp - controler.LastPing).ToString()))+ (ReturnData("
229     lastWarning", new TimeSpan(nowStamp - controler.LastWarning).
230     ToString()));
231
232 }
```

```
233
234
235
236
237
238
239     private string ReturnData(string tag, string data)
240     {
241         return "<" + tag + ">" + data + "</" + tag + ">";
242     }
243
244
245
246
247 }
248
249
250
251
252
253
254 }
```

Listing A.1: BoardsController.cs

```
1
2 using MongoDB.Driver.Builders;
3 using Newtonsoft.Json;
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Net;
8 using System.Net.Http;
9 using System.Web.Http;
10 using WebApplication9.Models;
11
12 namespace WebApplication9.Controllers
13 {
14     public class RClientsController : ApiController
15     {
16
17
18
19         public String Post([FromBody] CData data)
20
21         {
22
23
24             var userId =data.userId;
25             var password = data.password;
26             var action = data.action;
27             var sno = data.sno;
28
29
30             //checking ID
```



```
80         userVsController.SerialNumber);
81         responseGetController.Rooms.Add(controller);
82     }
83     string response = JsonConvert.SerializeObject(
84         responseGetController);
85     return response;
86 }
87
88
89
90
91     case "ARM":                // Arm the Controller Board
92     {
93         var controller = WebApiApplication.
94             CollectionControllers.FindOneById(sno);
95
96         if (controller != null)
97         {
98
99             controller.LastPing = DateTime.UtcNow.
100                 ToFileTime();
101
102             if (controller.ControllerState != Enums.
103                 ControllerState.Armed &&
104                 controller.ControllerState != Enums.
105                 ControllerState.Warning)
106             {
107                 //controller.ControllerState = Enums.
108                 //    ControllerState.Warning;
109                 controller.LastMovement = DateTime.Now.
110                     AddMinutes(-6).ToFileTimeUtc();
111                 controller.ControllerState = Enums.
112                     ControllerState.Armed;
113                 controller.DoorState = Enums.DoorState.
114                     Closed;
115                 WebApiApplication.CollectionControllers.
116                     Save(controller);
117                 return controller.ControllerState.ToString
118                     ();
119             }
120
121         }
122         break;
123     }
124
125     case "DISARM":            //DISARM the Controller BOARD
126     {
127
128         var controller = WebApiApplication.
```

```
123         CollectionControllers.FindOneById(sno);
124     if (controler != null)
125     {
126         controler.LastPing = DateTime.UtcNow.
127             ToFileTime();
128
129         if (controler.ControlerState != Enums.
130             ControlerState.Normal)
131         {
132             controler.ControlerState = Enums.
133                 ControlerState.Normal;
134             controler.DoorState = Enums.DoorState.
135                 Open;
136             controler.LastMovement = Now();
137             WebApiApplication.CollectionControllers.
138                 Save(controler);
139             return controler.ControlerState.ToString
140                 ();
141         }
142     }
143     }
144     }
145     }
146     }
147     }
148     }
149     }
150
151
152
153
154
155     return "";
156 }
157
158
159 private long Now()
160 {
161     return DateTime.Now.ToFileTimeUtc();           //Time Stamp
162 }
163
164
165
166
167 }
168 }
169 }
```

Listing A.2: RClientsController.cs

```
1
2 using MongoDB.Bson.Serialization.Attributes;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Web;
7
8 namespace WebApplication9.Models
9 {
10     public class Controller
11     {
12         [BsonId]
13         public string SerialNumber;
14         public Enums.ControllerType ControllerType;
15         public string Name;
16         public string Location;
17         public string[] PortConfig; // P=Pir, B=Door Button (open door), O=Output,
18             L=Lock (magnetic lock)
19         public long LastMovement; // Timestamp
20         public long LastPing; // Timestamp
21         public long LastWarning; // Timestamp
22         public Enums.ControllerState ControllerState;
23         public Enums.DoorState DoorState;
24         public bool UpdateConfig;
25     }
26 }
27
```

Listing A.3: Controller.cs

```
1
2
3
4 using MongoDB.Bson.Serialization.Attributes;
5 using System;
6 using System.Collections.Generic;
7 using System.Linq;
8 using System.Web;
9
10 namespace WebApplication9.Models
11 {
12     public class User
13     {
14         [BsonId]
15         public string Username;
16         public string Password;
17         public string rfID;
18         public string Pin;
19     }
20 }
```

```
20 }  
21 }
```

Listing A.4: User.cs

```
1  
2  
3 using MongoDB.Bson.Serialization.Attributes;  
4 using System;  
5 using System.Collections.Generic;  
6 using System.Linq;  
7 using System.Web;  
8  
9 namespace WebApplication9.Models  
10 {  
11     public class UserVsController  
12     {  
13         [BsonId]  
14         public string UID;  
15         public string Username;  
16         public string SerialNumber;  
17     }  
18 }
```

Listing A.5: UserVsController.cs

```
1  
2 using System;  
3 using System.Collections.Generic;  
4 using System.Linq;  
5 using System.Web;  
6  
7 namespace WebApplication9.Models  
8 {  
9     public class ApiFunctions  
10    {  
11        //Motion Deceted  
12  
13        public static long Now()  
14        {  
15            return DateTime.Now.ToFileTimeUtc();  
16        }  
17  
18  
19        public static void SetControlerState(Controler controler)  
20        {  
21            // Find state of the controler  
22            // Calculate the time between lastmovement and last Ping  
23  
24  
25            TimeSpan Inactive = new TimeSpan(0, 0, 5, 0);  
26            TimeSpan Warning = new TimeSpan(0, 0, 0, 30);  
27            TimeSpan MovementDiff = new TimeSpan(Now() - controler.LastMovement);  
28
```

```

29
30     switch (controler.ControlerState)
31     {
32         case Enums.ControlerState.Undefined:
33             {
34                 // Write in the log contact from new controler.
35                 break;
36             }
37         case Enums.ControlerState.Normal:
38             {
39                 if (MovementDiff > Inactive)
40                 {
41                     // Goto warning
42                     controler.ControlerState = Enums.ControlerState.
43                         Warning;
44                     controler.LastWarning = Now();
45                 }
46                 controler.DoorState = Enums.DoorState.Open;
47                 break;
48             }
49         case Enums.ControlerState.Warning:
50             {
51                 if (Warning > MovementDiff)
52                     // Some one activated the pir/button in the warning
53                     // period - Retrun to normal.
54                 {
55                     controler.ControlerState = Enums.ControlerState.
56                         Normal;
57                     controler.DoorState = Enums.DoorState.Open;
58                 }
59                 else
60                 {
61                     TimeSpan warningDiff = new TimeSpan(Now() - controler
62                         .LastWarning);
63                     if (warningDiff > Warning)
64                     {
65                         // Goto Armed
66                         controler.ControlerState = Enums.ControlerState.
67                             Armed;
68
69                         //If door is closed reading from sensor then
70                         //close the door
71
72                         controler.DoorState = Enums.DoorState.Closed;
73
74                     }
75                 }
76                 break;
77             }
78         case Enums.ControlerState.Armed:
79             {
80                 if (controler.LastMovement >= controler.LastPing) //

```



```
78         movement activated.
79     {
80         controler.ControllerState = Enums.ControllerState.Alarm
81         ;
82         controler.DoorState = Enums.DoorState.Closed;
83     }
84     break;
85 }
86 case Enums.ControllerState.Alarm:
87 {
88     controler.DoorState = Enums.DoorState.Closed;
89     break;
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
```

Listing A.6: ApiFunctions.cs

```
1
2
3
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Web;
8
9 namespace WebApplication9.Models
10 {
11     public class Enums
12     {
13         public enum ControlerType
14         {
15             Room,
16         }
17         public enum ControlerState
18         {
19             Undefined, // New
20             Normal, // Open
21             Warning, // Beeping 30 sec to arming
22             Armed, // Armed, alarm on
23             Alarm, // The alarm i active (sirene)
24         }
25
26         public enum ControlerEvent
27         {
28             None, // Nothing - the default.
29             PirActivation, // Pir activated
30         }
31     }
32 }
```

```
31
32     public enum DoorState
33     {
34         Open, // Nothing - the default.
35         Closed, // Pir activated
36     }
37
38
39
40     }
41 }
```

Listing A.7: Enums.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 namespace WebApplication9.Models
7 {
8     public class ResponceGetControler
9     {
10         public List<Controler> Rooms;
11     }
12 }
```

Listing A.8: ResponceGetConroler.cs

```
1
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Web;
6
7 namespace WebApplication9.Models
8 {
9     public class Data
10    {
11
12        public string portString { get; set; }
13
14        public string serialNumber { get; set; }
15
16        public string rfID { get; set; }
17
18        public string firststart { get; set; }
19
20        public string RequestPin { get; set; }
21
22
23    }
24 }
25 }
```

Listing A.9: Data.cs

```
1
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Web;
6
7 namespace WebApplication9.Models
8 {
9     public class CData
10    {
11
12        public string userId { get; set; }
13
14        public string password { get; set; }
15
16        public string action { get; set; }
17
18        public string sno { get; set; }
19
20
21    }
22 }
23
```

Listing A.10: CData.cs

A.2 Mobile Application Code

```
1
2
3
4 <!DOCTYPE html>
5 <html>
6 <style>
7
8
9 input[type=text], input[type=password] {
10     width: 100%;
11     padding: 12px 10px;    /* Clear Area around the content */
12     margin: 8px 0;        /* space around */
13     display: inline-block;
14     border: 1px solid #ccc; /* Box border width and color */
15     box-sizing: border-box; /* Sizing properties of box */
16 }
17
18 button {
19     background-color: #4CAF50; /* Color of button */

```

```
20     color: white;                /* Text Color */
21     padding: 14px 20px;         /* space around */
22     margin: 8px 0;
23     border: none;
24     width: 100%;
25 }
26
27 button:hover {
28     opacity: 0.8;
29 }
30
31
32
33 .imgcontainer {
34     text-align: center;
35     margin: 24px 0 12px 0;
36 }
37
38 img.avatar {
39     width: 40%;
40     border-radius: 50%;
41 }
42
43 .container {
44     padding: 16px;
45 }
46
47
48
49 </style>
50
51 <script type="text/javascript">
52
53     //var newWindow;
54     var result;
55     //var thisIsAnObject = {foo:'bar'};
56
57     function CheckLogin() { /*function to check userid & password*/
58
59
60         if( document.getElementById("us").value.length == 0 || document.getElementById("
61             ps").value.length == 0 || document.getElementById("us").value.trim().length
62             == 0 || document.getElementById("ps").value.trim().length == 0 )
63         {
64             alert("Please enter User/Pswd and white spaces not allowed in text field");
65             return false;
66         }
67
68
69         var xhttp = new XMLHttpRequest();
70
71         xhttp.onreadystatechange = function() { //CallBack Function
72             if (this.readyState == 4 && this.status == 200) {
73                 result = this.responseText;
```

```
73
74     result = result.split('').join('');
75
76     sessionStorage.setItem("usent", document.getElementById("us").value);
77     sessionStorage.setItem("pswsent", document.getElementById("ps").value);
78
79     if(result=="UserOk")
80     {
81
82
83         newWindow = window.open('menu.html')/*opens the target page while Id & password
84             matches*/
85     }
86     else {
87         alert(result);/*displays error message*/
88     }
89 }
90 };
91
92
93
94
95
96 xhttp.open("POST", "http://10.0.0.35:8080/api/RClient", true); //Post Method with URI
97   Path
98 xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); //
99   Headers
100 xhttp.send("userID="+document.getElementById("us").value + "&password="+document.
101   getElementById("ps").value+"&action=login"); //Request Body
102 }
103 </script>
104 <body>
105
106
107 <div class="imgcontainer">
108     
109 </div>
110
111 <div class="container">
112     <label><b>Username</b></label>
113     <input id="us" type="text" placeholder="Enter Username" name="uname" required>
114
115     <label><b>Password</b></label>
116     <input id="ps" type="password" placeholder="Enter Password" name="psw" required>
117
118     <button type="button" onClick="return CheckLogin()" >Login</button>
119
120 </div>
121
122
123
```

```
124 <div id="demo">
125
126 </body>
127 </html>
```

Listing A.11: index.html

```
1
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <style>
7
8
9
10 .room{
11
12     height:70px;
13     width:100%;
14     left:0px;
15     margin:5px 0;
16 }
17
18
19 .button {
20     padding: 10px 25px;
21     position:absolute;
22     top: 500px;
23     right:58px;
24     width: 250px;
25     font-size: 24px;
26     cursor: pointer;
27     text-align: center;
28     outline: none;
29     color: #fff;
30     background-color: #4CAF50;
31     border: none;
32     border-radius: 15px;
33 }
34
35 div.groom {
36     position:absolute;
37     left:0px;
38     width: 100%;
39     color: white;
40     bottom : 0;
41 }
42
43
44 p1 {
45
46     font-size: 15px;
47 }
48
```

```
49
50
51
52 .button1 {
53   position: absolute;
54   padding: 10px 0px;
55   width: 100px;
56   right:0%;
57   font-size: 24px;
58   cursor: pointer;
59   text-align: center;
60   outline: none;
61   color: #fff;
62   background-color: #4CAF50;
63   border: none;
64   display:"inline";
65 }
66
67
68
69 .button2 {
70
71
72   width: 100%;
73   font-size: 24px;
74   padding: 10px 25px;
75   cursor: pointer;
76   text-align: center;
77   outline: none;
78   color: #fff;
79   background-color: #4CAF50;
80   border: none;
81
82 }
83
84 .button2:active {
85
86   transform: translateY(2px);
87 }
88
89
90 .button1:active {
91
92   transform: translateY(2px);
93 }
94
95 </style>
96 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></
   script>
97
98 <script type="text/javascript">
99
100   var uid = sessionStorage.getItem("usent");
101   var psw = sessionStorage.getItem("pswsent");
102   var res;
```

```
103     var jsn;
104     var sd;
105     var state;
106     var btn1;
107
108 function ARM(sNumber,t)
109 {
110
111 //alert(t);
112
113
114 var xhttp = new XMLHttpRequest();
115 xhttp.onreadystatechange = function() {
116     if (this.readyState == 4 && this.status == 200) {
117         res = String(this.responseText);
118         res = res.replace(/\\/g, "");
119         res = res.substring(1,res.length - 1);
120         //alert(res);
121         //btn1=res;
122         //var change =document.getElementById("m"+sNumber);
123         //change.value=btn1;
124
125         $("#MAIN").empty();
126         RoomList();
127
128
129
130     }
131
132
133 };
134
135 xhttp.open("POST", "http://10.0.0.35:8080/api/RClient", true);
136 xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
137 xhttp.send("userID="+uid + "&password="+psw+"&action="+t+"&sno="+sNumber);
138
139
140 }
141
142
143
144
145
146
147 function RoomList() { /*function to check userid & password*/
148
149
150     //var myVariable = window.opener.thisIsAnObject;
151     //alert(myVariable.foo);
152
153
154 var xhttp = new XMLHttpRequest();
155 xhttp.onreadystatechange = function() {
156     if (this.readyState == 4 && this.status == 200) {
157         res = String(this.responseText);
```



```

158     res = res.replace(/\\/g, "");
159     res = res.substring(1, res.length - 1);
160
161     //alert(res);
162     //document.getElementById("demo").innerHTML = sd;
163
164     jsn=JSON.parse(res);           //parsing the response
165
166     for (var i = 0; i < jsn.Rooms.length; i++) { //loop based on rooms
167                                           //received
168     //alert(jsn.Rooms[i].SerialNumber);
169     //jsn=JSON.parse(res);
170     //alert(jsn.Rooms[i].SerialNumber);
171                                           //Creating Div into Main Div
172     $('<div/>', {
173         id: jsn.Rooms[i].SerialNumber,
174         class: 'room' }
175     ).prependTo('#MAIN');
176
177     if(jsn.Rooms[i].ControlerState == 1) //Translating numbers to user
178                                           //understandable format
179     {
180         state="Normal";
181         btn1="ARM";
182     }
183
184     else if(jsn.Rooms[i].ControlerState == 3)
185     { state="ARMED";
186       btn1="DISARM";
187     }
188     else if(jsn.Rooms[i].ControlerState == 2)
189     { state="Warning";
190     }
191     else{}
192
193     //Updating the newly created div contents
194     document.getElementById(jsn.Rooms[i].SerialNumber).innerHTML="<p1>Board Number :"+
195     jsn.Rooms[i].SerialNumber+ '<button class="button1" type="button" value='+btn1
196     + ' onclick='+ "ARM(" +jsn.Rooms[i].SerialNumber+', '+ "\""+btn1+"\" " +') id=m'+
197     jsn.Rooms[i].SerialNumber+ '>'+ btn1+'</button>' + "<br>Controller State:"+
198     state+"</p1>";
199
200 }
201 };
202
203 xhttp.open("POST", "http://10.0.0.35:8080/api/RClient", true);
204 xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
205 xhttp.send("userID="+uid + "&password="+psw+"&action=getControlers");
206
207 //alert(newWindow.result);
208 //alert(window.result);

```

```
209 }
210 }
211
212
213 </script>
214
215 </head>
216 <body>
217
218     <div id="MAIN">
219     </div>
220
221
222
223
224 <div class="groom">
225
226 <button class="button2" onClick="RoomList()">Get Room List</button>
227 </div>
228
229 <div id="demo">
230
231 </body>
232 </html>
```

Listing A.12: menu.html

A.3 Firmware

```
1
2
3 #include <SPI.h>
4 #include <Ethernet.h>
5 #include <Wiegand.h>
6 #include "AdeoController.h"
7
8
9 // MAC
10 byte mac[] = {
11     0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
12 };
13
14
15
16 //Wiegand
17 WIEGAND wg;
18
19
20 String postData;
21
22
23 //char server[] = "www.arduino.cc";
```

```
24 //char server[] = "192.168.2.71";
25
26
27
28 //char server[] = "www.arduino.cc";
29 // initialize the library instance:
30 EthernetClient client;
31
32 int beeperState = LOW;
33 unsigned long success = 0;
34 unsigned long failure = 0;
35 //String sn = 1234;
36 long OnTime = 15;           // milliseconds of on-time
37 long OffTime = 985;
38 unsigned long pinEntryTime = 15L * 1000L;;
39
40 int blueledstate = LOW;
41 //int redledstate = LOW;
42
43 unsigned long lastConnectionTime = 0;           // last time you connected to the
         server, in milliseconds
44 unsigned long lastCheck = 0;           // last time you connected to the server, in
         milliseconds
45 unsigned long oneSec = 1L * 1000L;
46 unsigned long halfSec = 0.5L * 1000L;
47 unsigned long beepDuration=0;
48 unsigned long beepInterval = 200L;
49 unsigned long ledDuration = 0;
50 unsigned long pinDuration = 0;
51
52 const unsigned long postingInterval = 1L * 1000L; // delay between updates, in
         milliseconds
53 const unsigned long TimeoutInterval = 60L * 1000L; // delay between updates, in
         milliseconds
54 const unsigned long postingLong = 15L * 1000L; // delay between updates, in
         milliseconds
55
56 // the "L" is needed to use long type numbers
57
58
59 int blueled = 19;
60 int redled = 18;
61 int doorBell = 6;
62 int BellState = 0;
63 char PortConfig[8];
64 char CurrentStatus[8];
65 char PortTransmit[8];
66 char OldPortTransmit[8];
67 char buttonTransmit = 'N';
68 bool Transmitting= false;
69 bool debugtext= false;
70 bool currentBellStatus = false;
71 bool beeper = false;
72 bool firstwarning = false;
73 bool firststart = true;
```

```

74 bool pin = false;
75 //bool timeout = false;
76
77 String readString = String(800); //string for fetching data from address
78 String rfID="";
79 String pinValue;
80 void setup() {
81   //serial port:
82   Serial.begin(57600);
83   while (!Serial) {
84     ; // only
85   }
86   // give the ethernet module time to boot up:
87   delay(1000);
88   // start the Ethernet connection using a fixed IP address and DNS server:
89   //Ethernet.begin(mac, ip, myDns);
90   Ethernet.begin(mac);
91   // print the Ethernet board/shield's IP address:
92   Serial.print("My IP address: ");
93   //Serial.println(Ethernet.localIP());
94   wg.begin();
95   strncpy(PortConfig, "NNNNNNNN",8);
96   strncpy(CurrentStatus, "NNNNNNNN",8);
97   strncpy(PortTransmit, "NNNNNNNN",8);
98   strncpy(OldPortTransmit,"NNNNNNNN",8);
99   // initialize the LED pin as an output:
100  // pinMode(18, OUTPUT);
101  // initialize the pushbutton pin as an input:
102
103  // pinMode(7, OUTPUT); //LED
104  // digitalWrite(7, HIGH);
105
106  pinMode(6, OUTPUT); //BEEPER
107  digitalWrite(6, LOW);
108  Serial.println("Started");
109  setupPorts(PortConfig);
110
111
112
113  // Using analog pins for the status indication
114
115  pinMode(blueled, OUTPUT); //blue led
116  digitalWrite(blueled, LOW);
117
118  pinMode(redled, OUTPUT); //blue led
119  digitalWrite(redled, LOW);
120
121  pinMode(17, OUTPUT); //Warning
122  digitalWrite(17, LOW);
123
124
125  pinMode(5, OUTPUT); // Setting pin number 5 as OUTPUT
126
127  digitalWrite(5,HIGH); // The pin 5 will generate HIGH (3.3V) Signal
128

```



```

184     pinMode(i, INPUT);
185     break;
186 }
187 case 'L':
188 {
189     pinMode(i, OUTPUT);
190     break;
191 }
192 case 'M':
193 {
194     pinMode(i, INPUT);
195     break;
196 }
197 case 'S':
198 {
199     pinMode(i, OUTPUT);
200     break;
201 }
202 default:
203     break;
204 }
205 }
206 }
207 }
208 }
209
210
211 int buttonState = 0;
212
213 void CheckPorts() {
214     // Use config to test ports.
215     for(int i=14; i< 22; i++){
216         int ix = i-14;
217         switch (PortConfig[ix])
218         {
219             case 'B':
220                 buttonState= digitalRead(i);
221                 if(buttonState == HIGH)
222                 {
223                     CurrentStatus[ix] = 'T';
224                     if(PortTransmit[ix] == 'N')
225                     {
226                         PortTransmit[ix] = 'T';
227                         logfile((String)"Port: " + i + " triggered");
228                     }
229                 }
230             else
231             {
232                 CurrentStatus[ix] = 'N';
233             }
234
235             break;
236             case 'P':
237                 //do something when var equals 2
238                 CurrentStatus[ix] = checkSensor(i);

```

```
239
240     if (CurrentStatus[ix] != 'N')
241     {
242         if (PortTransmit[ix] == 'N')
243         {
244             PortTransmit[ix] = CurrentStatus[ix];
245             logfile((String)"Port: " + i + " triggered");
246         }
247     }
248     break;
249     default:
250         // if nothing else matches, do the default
251         // default is optional
252     break;
253 }
254 }
255
256
257
258 }
259
260 char checkSensor(int sensorInput)
261 {
262     //Serial.print(sensorInput);
263
264     char state;
265
266     int sensorReading = analogRead(sensorInput);
267     //logfile(String(sensorReading));
268     if (sensorReading < 400) {
269         state = 'W'; // Wire shorted. Possible tampering.
270     }
271     else if (sensorReading >= 400 && sensorReading < 590) {
272         state = 'N'; // Normal state, sensor not triggered
273     }
274     else if (sensorReading >= 590 && sensorReading < 800) {
275         state = 'T'; // Sensor triggered.
276     }
277     else {
278         state = 'C'; // Open circuit. Cut or tamper triggered.
279     }
280     // Output the current reading to the host via the serial connection
281     //Serial.print(sensorInput, DEC);
282     //Serial.print(": ");
283     //Serial.print(sensorReading, DEC);
284     //Serial.print(" (");
285     switch (state) {
286     case 'W': // Wire shorted. Possible tampering.
287         // Serial.print("Wire short sound alarm");
288         break;
289     case 'N':
290         // Serial.print("Normal state");
291         break;
292     case 'T':
```

```
294     // Serial.print("Sensor triggered");
295     break;
296 default:
297     //Serial.print("Open. Cut or tamper sound alarm");
298     break;
299 }
300 //Serial.println(" ");
301 // Pass the current state back to the calling function
302 //delay(100);
303 return state;
304 }
305
306
307 void CheckReader()
308 {
309
310
311
312 if ( (pin == true) && ((millis() - pinDuration) >= pinEntryTime) )
313 {
314
315     lastConnectionTime = 0;
316 }
317
318
319 if (wg.available())
320 {
321
322 // //Serial.print("Wiegand HEX = ");
323 // //Serial.print(wg.getCode(), HEX);
324 // //Serial.print(", DECIMAL = ");
325 // //Serial.print(wg.getCode());
326 // //Serial.print(", Type W");
327 // //Serial.println(wg.getWiegandType());
328 // //Serial.print(wg.getCode());
329 //
330 //
331
332 if(pin)
333 {
334     pinValue += String(wg.getCode());
335     lastConnectionTime = millis();
336     pinDuration=millis();
337
338     logfile("This is the rfid value");
339     logfile(rfID);
340     logfile(pinValue);
341     Serial.println("Key Presses");
342     //Serial.println(rfID);
343     Serial.println(pinValue);
344
345     if (pinValue.length() > 3)
346     {
347         logfile(pinValue);
348         Serial.println("Final Pin Value");
```



```
349         Serial.println(pinValue);
350         lastConnectionTime = 0;
351         pin = false;
352     }
353
354     }
355     else if(pin!=true)
356     {
357         rfID = String(wg.getCode());
358         logfile((String)"RFID DATA: " + rfID + " triggered");
359         Serial.print((String)"RFID DATA: " + rfID + " triggered");
360         pin = true;
361         lastConnectionTime = 0;
362     }
363 }
364
365 void CheckBell()
366 {
367     BellState = digitalRead(doorBell);
368
369     if (BellState == HIGH)
370     {
371         if(currentBellStatus == false)
372         {
373             //send port
374
375             if (buttonTransmit == 'N')
376             {
377                 buttonTransmit = 'Y';
378                 logfile((String)"Button: " + buttonTransmit + " triggered");
379             }
380
381             if (lastConnectionTime > oneSec)
382             {
383                 lastConnectionTime = 0;
384             }
385
386             currentBellStatus = true;
387         }
388     }
389     else
390     {
391         currentBellStatus = false;
392     }
393 }
394
395 }
396
397 }
398
399 }
400
401 }
402
403 }
```

```
404 }
405 }
406
407
408 void CheckTransmitState() {
409
410 //Is transmitting?
411 if(Transmitting)
412 {
413 //logfile("I am in tranmiting");
414 // If client.available()
415 if(client.available())
416 {
417 logfile("I am in client avaiable lets see");
418 // Read it All,
419 readString = "";
420 while (client.available()) {
421 char c = client.read();
422 //Serial.write(c);
423 //read char by char HTTP request
424 if (readString.length() < 400) {
425
426 //store characters to string
427 readString+= c;
428 }
429 }
430 }
431
432 logfile("EndOf return :");
433
434 EvaluateReturnValues();
435 Serial.println(readString);
436 lastConnectionTime = millis();
437 lastCheck = millis();
438 Transmitting= false;
439 }
440 else
441 {
442 //logfile("Timer Down");
443 //logfile("Timeout Timer "+String(millis() - lastConnectionTime));
444 //logfile("Timer Up");
445 //timeout = true;
446 if (millis() - lastConnectionTime > TimeoutInterval) {
447 Transmitting= false;
448 //timeout = false;
449 lastCheck = millis()+ (30L*1000L);
450 client.stop();
451 logfile("Ending transmission due to timeout");
452 // Update the portstatus with not send data...
453 }
454 }
455 }
456 }
457 else
458 {
```

```
459
460 // Is it time
461 if (millis() - lastCheck > postingInterval) {
462     // Check if
463     logfile("*****I am in check*****");
464     logfile(String(lastCheck));
465     bool sendit=false;
466     logfile(PortTransmit);
467     for(int i=0; i< 7; i++){
468         if(PortTransmit[i] != 'N')
469         {
470             String xxx = (String)"Port: " + i + " triggered"; //This is the reason
471             logfile(xxx);
472             sendit=true;
473         }
474     }
475
476
477     logfile(String(millis() - lastConnectionTime));
478     //ledDuration = millis();
479     //Blink();
480     logfile(String(postingLong));
481
482     if (millis() - lastConnectionTime > postingLong) {
483         sendit = true;
484     }
485
486     if(sendit)
487     {
488         // Diff, Send the data.
489         //BuildHttpRequest();
490     }
491     else
492     {
493         logfile("not sending anything no diff");
494         // Jump interval.
495         lastCheck = millis();
496         logfile("*****This is the last Check*****");
497     }
498 }
499
500 }
501
502 }
503
504 void EvaluateReturnValues()
505 {
506     // Find config
507     char newSetting[8];
508
509     String doorStatus = GetValue("doorStatus"); // get keyword doorStatus from the
510         response
511     String state = GetValue("state");           // get keyword state from the
512         response
```

```
511     String tempnewSetting=GetValue("config");    // get keyword config from the
           response
512 String pinStatus = GetValue("pin");            // get keyword pinStatus from the
           response
513
514 logfile("Check out the Status");
515 logfile(pinStatus);
516
517
518 tempnewSetting.toCharArray(newSetting, 8);
519 setupPorts(newSetting);
520
521 Serial.print(state);
522
523 if (state == "Normal")
524 {
525
526     beeper = false;
527     firstwarning = false;
528 }
529
530 if (state == "Warning")
531 {
532     // Start beeping....
533
534     beeper = true;
535     if (firstwarning == false)
536     {
537         beepInterval = 1000;
538         firstwarning = true;
539     }
540     else
541     {
542         beepInterval = 200;
543         firstwarning = false;
544     }
545
546     logfile("*****I am in Beeper*****");
547
548 }
549
550
551 if (state == "Armed")
552 {
553     // Show red indicator on RFID
554     //(When door is closed led changes to Red)
555     beeper = false;
556 }
557
558 if (state == "Alarm")
559 {
560     // Start beeping....
561     beepInterval = 10;
562     beeper = true;
563 }
```

```
564 }
565
566 if(doorStatus == "Open")
567 {
568     digitalWrite(5, HIGH); // Open the door
569 }
570
571
572 if (doorStatus == "Closed")
573 {
574     digitalWrite(5, LOW); // Close the door
575 }
576
577
578 if (pinStatus == "On")
579 {
580     pin = true;
581     pinDuration = millis();
582 }
583
584
585 if (pinStatus == "")
586 {
587     pin = false;
588     rfID = "";
589     pinValue = "";
590 }
591
592
593
594 }
595
596
597 String GetValue(String tag)
598 {
599
600     int fromIx = readString.indexOf("<" + tag + ">");
601     if (fromIx == -1)
602     {
603         return "";
604     }
605     int ToIx = readString.indexOf(">" + tag + "<");
606     String returnValue = readString.substring(fromIx + tag.length()+2, ToIx);
607     logfile(returnValue);
608     return returnValue;
609 }
610 void logfile(String text)
611 {
612     if(debugtext)
613     {
614         Serial.println(text);
615     }
616
617 }
618
```

```

619 // this method makes a HTTP connection to the server:
620 void BuildHttpRequest() {
621     //HttpRequestTest();
622     httpRequest();
623     Transmitting = true;
624     strncpy(OldPortTransmit,PortTransmit,8);
625     strncpy(PortTransmit,"NNNNNNNN",8);
626     buttonTransmit = 'N';
627     postData = "";
628     //rfID = "";
629     // note the time that the connection was made:
630     lastConnectionTime = millis();
631 }
632 }
633
634
635
636
637 void httpRequest() {
638     //Blink();
639     logfile("before stop");
640     logfile((String)client.status());
641     client.stop();
642     logfile("after");
643     // if there's a successful connection:
644
645
646
647
648     digitalWrite(redled, HIGH);
649     int statuscode = client.connect(server, 80);
650     Serial.println("connection : " + (String)statuscode);
651
652
653
654     if (statuscode>=0){
655
656
657         Serial.println("connecting..");
658
659     postData = "serialNumber=12345&portString=" + String(PortTransmit) + "&rfID=" + rfID +
660         "&RequestPin=";
661
662     if (pin)
663     {
664         //logfile("Pin Send");
665         postData = postData + pinValue;
666     }
667
668     // send the HTTP GET request:
669     client.print("POST api/boards HTTP/1.1");
670     client.print("Host : 10.0.0.42:8080");
671     client.print("Connection: close");
672     client.println("Content-Type: application/x-www-form-urlencoded;");

```

```
673 client.print("Content-Length: ");
674 client.println(postData.length());
675     client.println();
676 client.println(postData);
677
678 } else {
679     // if you couldn't make a connection:
680     failure++;
681     Serial.println("connection failed: "+(String)statusCode);
682 }
683 }
684
685
686 void httpRequestTest() {
687     // close any connection before send a new request.
688     // This will free the socket on the WiFi shield
689     client.stop();
690
691     // if there's a successful connection:
692     if (client.connect(server, 80)) {
693         Serial.println("connecting...Test");
694         // send the HTTP GET request:
695         client.println("GET HTTP/1.1");
696         //client.println("Host: www.arduino.cc");
697         client.println("User-Agent: arduino-ethernet");
698         client.println("Connection: close");
699         client.println();
700     } else {
701         // if you couldn't make a connection:
702         Serial.println("connection failed");
703     }
704 }
705
706 void CheckDoorPosition()
707 {
708     int doorAlignment = digitalRead(0);
709
710     if (doorAlignment == LOW)
711     {
712         Serial.println("YESS IT WORKSSSS!!");
713     }
714
715     if (doorAlignment == HIGH)
716     {
717         Serial.println("HIGH CHARGE WORKSSSS!!");
718     }
719 }
720
721 void Beeper()
722 {
723
724
725
726     if (beeper)
```

```
728 {
729
730     if (millis() - beepDuration >= beepInterval) // Change state after every
        specified time
731     {
732         beepDuration = millis();
733
734         if (beeperState == LOW)
735         {
736             beeperState = HIGH;
737         }
738
739         else {
740
741             if(beepInterval >10 )
742                 beeperState = LOW;
743
744         }
745
746         digitalWrite(6, beeperState); // Fluctuating the output in order to play
        sound at different interval
747
748
749
750         digitalWrite(17, beeperState); // Red LeD
751     }
752 }
753
754 else if(beeper == false)
755 {
756     beeperState = LOW;
757     digitalWrite(6, beeperState);
758     digitalWrite(17, beeperState); //Red Led
759 }
760
761 }
762
763
764 void Blink()
765 {
766
767     if ( (blueledstate== HIGH) && (millis() - ledDuration) >= OnTime)
768     {
769
770         blueledstate = LOW;
771         ledDuration = millis();
772         digitalWrite(blueled, blueledstate);
773
774
775         //if (timeout == true)
776         //{
777         //    redledstate = LOW;
778         //    digitalWrite(redled, redledstate);
779         //}
780
```



```
781     }
782
783
784     else if ((blueledstate == LOW) && (millis() - ledDuration) >= OffTime)
785     {
786         blueledstate = HIGH;
787         ledDuration = millis();
788         digitalWrite(blueled, blueledstate);
789
790         //if (timeout == true)
791         //{
792         //    redledstate = HIGH;
793         //    digitalWrite(redled, redledstate);
794         //}
795
796     }
797
798
799
800 }
```

Listing A.13: Board Firmware

Bibliography

- [1] URL: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.
- [2] URL: <https://www.forbes.com/sites/louiscolombus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#719d3e13292d>.
- [3] URL: <http://www.historyofkeys.com/>.
- [4] URL: https://www.alibaba.com/product-detail/RFID-Access-Control-System-Single-door_519509382.html.
- [5] URL: <https://i.pining.com/736x/61/b7/2d/61b72d59c179e8d13349473f1c59c4e8--security-lock-home-security.jpg>.
- [6] URL: <http://grandingchina.en.made-in-china.com/product/pMzJPXeuCwYS/China-Fingerprint-Access-Control-with-TCP-IP-5000APlus-.html>.
- [7] URL: https://en.wikipedia.org/wiki/Security_alarm.
- [8] URL: https://www.matrixssl.com/wiki/index.php?title=What_Is_a_Keypad.
- [9] URL: <http://www.home-security-systems-answers.com/alarm-system-keypads.html>.
- [10] URL: http://www.blunet.net.cn/uploads/Siren-Alarm/Siren-Alarm/_1A25G24.jpg.
- [11] URL: <http://www.smartliving.com.au/insteon-window-door-open-close-sensor.html>.
- [12] URL: <http://www.alarm.org/HomeSafety/BurglarsSpillAboutSecuritySystems.aspx>.
- [13] URL: https://en.wikipedia.org/wiki/Distributed_computing.
- [14] URL: https://en.wikipedia.org/wiki/Distributed_computing.
- [15] URL: <http://wla.berkeley.edu/~cs61a/fall/lectures/communication.html#id3>.
- [16] URL: <http://cse.csusb.edu/tongyu/courses/cs660/notes/distarch.php>.

-
- [17] URL: http://www.berkes.ca/archive/berkes_gnutella_freenet.pdf.
- [18] URL: <https://www.infosec.gov.hk/english/technical/files/peer.pdf>.
- [19] URL: <https://www.kullabs.com/classes/subjects/units/lessons/notes/note-detail/3042>.
- [20] URL: <http://www.ubiq.com/hypertext/weiser/UbiHome.html>.
- [21] URL: http://newmedia.wikia.com/wiki/Ubiquitous_Computing.
- [22] URL: <http://www.thbs.com/blog/ubiquitous-computing-living-in-a-smart-world>.
- [23] URL: <http://www.ubiq.com/hypertext/weiser/UbiHome.html>.
- [24] URL: http://newmedia.wikia.com/wiki/Ubiquitous_Computing.
- [25] URL: <http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>.
- [26] URL: <https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/connecting-legacy-devices-brief.pdf>.
- [27] URL: <https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#70f16ec61d09>.
- [28] URL: <http://www.technologyguide.com/feature/internet-of-things/>.
- [29] URL: <http://www.nxp.com/applications/solutions/internet-of-things/smart-things/healthcare/wireless-insulin-pump:WIRELESS-INSULIN-PUMP>.
- [30] URL: <https://iot-analytics.com/10-internet-of-things-applications/>.
- [31] URL: <https://nest.com/thermostats/nest-learning-thermostat/overview/>.
- [32] URL: <http://www.scoot-utc.com/>.
- [33] URL: <http://www.csoonline.com/article/3075023/privacy/the-difference-between-privacy-and-security.html>.
- [34] URL: <http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>.
- [35] URL: <https://en.wikipedia.org/wiki/Turnstile>.
- [36] URL: <https://security.stackexchange.com/questions/35351/what-is-the-difference-between-types-of-motion-sensors>.
- [37] URL: <http://www.infodev.ca/about/read-our-articles/comparative-analysis-of-counting-technologies.html>.
- [38] URL: http://cdn2.hubspot.net/hubfs/325181/The_Truth_on_In-Store_Analytics_WiFi_iBeacon_and_Video_in_Retail.pdf.

- [39] URL: https://developer.mbed.org/users/4180_1/notebook/using-the-hc-sr04-sonar-sensor/.
- [40] URL: <http://lwww.ecmweb.com/lighting-amp-control/occupancy-sensors-101>.
- [41] URL: <http://www.facilitiesnet.com/lighting/article/Occupancy-Sensors-Passive-Infrared-Ultrasonic-and-Dual-Technology-Facility-Management-Lighting-Feature--9608>.
- [42] URL: <http://www.briticent.co.uk/assets/Uploads/ispot-sensor-PIR-vs-Microwave.pdf>.
- [43] URL: <https://www.online-sciences.com/technology/microwave-motion-detector-microwave-sensor-uses-features-cons-and-pros/>.
- [44] URL: <https://www.ibm.com/blogs/internet-of-things/sensors-smart-home/>.
- [45] URL: <http://securisat.co.za/difference-active-infrared-ir-beam-passive-infrared-pir/>.
- [46] URL: <http://www.securityandselfdefensestore.com/resources/47-security-cameras/151-pir-detection.html>.
- [47] URL: <https://www.pololu.com/product/2731>.
- [48] URL: <https://circuitdigest.com/electronic-circuits/pir-sensor-based-motion-detector-sensor-circuit>.
- [49] URL: <http://howtomechatronics.com/tutorials/arduino/how-pir-sensor-works-and-how-to-use-it-with-arduino/>.
- [50] URL: <https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor?view=all>.
- [51] URL: <https://www.hkvstar.com/technology-news/difference-between-pir-motion-sensor-and-infrared-beam-motion-sensor.html>.
- [52] URL: <https://www.ecosensor.co.za/pages/motion-sensor-selection>.
- [53] URL: <https://www.alarmgrid.com/faq/how-does-a-motion-sensor-work>.
- [54] URL: <https://www.aliexpress.com/popular/pir-microwave.html>.
- [55] URL: <http://www.homesecurityguru.com/microwave-motion-sensors>.
- [56] URL: <http://www.yli.cn/en/customer/FAQ/2013-02-01/474.html>.
- [57] URL: https://en.wikipedia.org/wiki/Electromagnetic_lock.
- [58] URL: <http://www.yli.cn/en/customer/FAQ/2013-02-01/453.html>.
- [59] URL: https://www.cctv-information.co.uk/i/Access_Control.
- [60] URL: <http://www.stampsco.com/security-access/learn-access-control/types-of-readers/>.

- [61] URL: <http://www.nationwidesecuritycorp.com/wp-content/uploads/What-Type-of-Access-Control-Reader-Should-I-be-Using.pdf>.
- [62] URL: <http://www.maglocks.com/access-guide>.
- [63] URL: <http://www.tcom-security.co.uk/card-pin-access-control.html>.
- [64] URL: <https://www.raspberrypi.org/>.
- [65] URL: <http://www.explainthatstuff.com/howrelayswork.html>.
- [66] URL: <http://blog.opensecurityresearch.com/2012/12/hacking-wiegand-serial-protocol.html>.
- [67] URL: <https://www.supremainc.com/en/node/754>.
- [68] URL: http://kb.supremainc.com/knowledge/doku.php?id=en:1xfaq_understanding_wiegand.
- [69] URL: <https://www.honeywellaccess.com/documents/Td2058.pdf>.
- [70] URL: <https://www.itead.cc/iboard.html>.
- [71] URL: <https://github.com/monkeyboard/Wiegand-Protocol-Library-for-Arduino>.
- [72] T. M. Alfaqih and Al Muhtadi. "Internet of Things Security based on Devices Architecture". In: *International Journal OF Engineering Sciences and Management Research* (2016). URL: <http://www.ijcaonline.org/research/volume133/number15/alfaqih-2016-ijca-908191.pdf>.
- [73] Kevin Ashton. "That "internet of things" thing". In: (). URL: www.rfidjournal.com/articles/pdf?4986.
- [74] F. Bu et al. "Estimating pedestrian accident exposure: automated pedestrian counting devices report". In: (2007).
- [75] G. Cessford et al. "Developing new visitor counters and their applications for management". In: (2002).
- [76] C. Y. Chan and F. Bu. "Literature review of pedestrian detection technologies and sensor survey. technical report". In: (2005).
- [77] G. F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems: principles and paradigms*. Pearson-Education, 2005.
- [78] C. Harrison, J. Wiese, and A. K. Dey. "Achieving ubiquity: The new third wave". In: (2010).
- [79] Qi Jing et al. "Security of the Internet of Things: Perspectives and challenges. Wireless Networks." In: (2014). URL: http://csi.dgist.ac.kr/uploads/Seminar/1407_IoT_SSH.pdf.
- [80] J. John Livingston and A. Umamakeswari. "Internet of Things Application using IP-enabled Sensor Node and Web Server." In: *International Journal of Computer Networks* (2015). URL: <http://www.indjst.org/index.php/indjst/article/view/65577>.

- [81] R. Khan et al. "Future internet: the internet of things architecture, possible applications and key challenges." In: (2012). URL: <http://ieeexplore.ieee.org.proxy.findit.dtu.dk/stamp/stamp.jsp?arnumber=6424332>.
- [82] P. Marwedel. *Embedded system design (Vol. 1)*. Springer, 2006.
- [83] G. S. Matharu, P. Upadhyay, and L. Chaudhary. "The Internet of Things: challenges security issues". In: *International Conference* (2014). URL: <http://www.ijesmr.com/doc/Archive-2016/November-2016/5.pdf>.
- [84] Sethi Pallavi and R. Sarangi Smruti. "That "internet of things" thing". In: *Journal of Electrical and Computer Engineering* (). URL: <https://www.hindawi.com/journals/jece/2017/9324035/>.
- [85] T. Pering and R. Want. "System challenges for ubiquitous and pervasive computing." In: *27th International Conference on Software Engineering* (2005).
- [86] O. Said and M. Masud. "Towards internet of things: Survey and Future Vision." In: *International Journal of Computer Networks* (2013). URL: http://cs.brown.edu/courses/cs227/papers/Towards_Internet_of_Things_Survey_and_Fu.pdf.
- [87] R. Sampson. "False burglar alarms". In: (2001).
- [88] H. Suo et al. "Security in the internet of things: a review". In: *In Computer Science and Electronics Engineering (ICCSEE)* (2012). URL: <http://ieeexplore.ieee.org.proxy.findit.dtu.dk/stamp/stamp.jsp?arnumber=6188257>.
- [89] A. S. Tanenbaum and M. Van Steen. *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.
- [90] D. Uckelmann, M. Harrison, and F. Michahelles. *Architecting the internet of things*. Springer, 2011.
- [91] A. V. Vijayalakshmi and L. Arockiam. "A Study on Security Issues and Challenges IN IOT". In: *International Journal OF Engineering Sciences Management Research* (2016). URL: http://www.repository.up.ac.za/dspace/bitstream/handle/2263/39762/Ye_Efficient_2014.pdf?sequence=1.
- [92] M. Weiser. "The future of ubiquitous computing on campus." In: (1998).
- [93] M. Wu et al. "Research on the architecture of Internet of things". In: *In Advanced Computer Theory and Engineering (ICACTE)* (2010).
- [94] Ning YE et al. "An Efficient Authentication and Access Control Scheme for Perception Layer of Internet of Things". In: *International Journal of Applied Mathematics Information Sciences* (2014). URL: http://www.repository.up.ac.za/dspace/bitstream/handle/2263/39762/Ye_Efficient_2014.pdf?sequence=1.

