Zagazig University
Faculty of Engineering
Computer and Systems Engineering Dept.

B. Eng. Final Year Project

REMOTE CONTROL SYSTEM FOR SMART HOMES

In Partial Fulfillment
for The Award Of The degree of
Bachelor of Engineering
In
Computers and Systems Engineering Dept.

Supervisor:
Dr. Ahmed Helmi

By:
Ahmed Mahmoud Ali
Anas Ahmed Fouad
Mahmoud El-Sayed Hussein
Menna-Allah Ashraf Zaher
Mustafa Kamel El-Sayed
Sultan Ibrahim Ibrahim

JULY, 2017

DEDICATION

To our professor and supervisor Dr. Ahmed Helmi for his guidance and instructions.

To our professor Dr. Haitham Abu Bakr for.his help and being patient to respond our questions.

To our families for their continuous support.

We dedicate this report.

DECLARATION

 We certify that this project, the entire design and construction of the system was carried out and submitted as true work of our team members under the supervision of **Dr.Ahmed Helmi** in computers and systems engineering department, Zagazig University, in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer and Systems Engineering.

Dr. Ahmed Helmi (Project Supervisor)	Dr. Nesreen I.Zidan (Head of Department)

ABSTRACT

This project involves the design and construction of remote control system for home automation using Raspberry pi and Arduino kits via internet. Home automation is the automatic or semi-automatic control and monitoring of household appliances and residential house features like doors and even windows. In this project, we design and build a multipurpose remotely controlled system that can provide these features: safety and security (Fire detection, Motion detection, and Water leakage detection, Gas leakage detection), appliances (Light control, Temperature control, Doors control, television, washer and fridge control). The system is not limited to these features as we can add as many as we need.

We made a model of the system on a maquette to illustrate how the connection and the behavior of the system works, and how these features are applied by implementing it using sensors, actuators and microcontrollers. In addition, we made different user interfaces to allow the user to manage the system. We have three-user interfaces android mobile application, raspberry pi GUI and website, these interfaces help user to monitor the house and control it.

We consider the whole project as a human body, the Raspberry pi is its brain, the server is its neural system as it connects everything together, the sensors are its eyes, noses and ears and the house is the environment.

The sensors sense the environment and send their output to the arduino kits, which in turn send Wi-Fi signals to the raspberry pi. The raspberry pi controls the whole system, collects the sensors' data then sends it to the server, which updates the database and sends notifications to the users. In case the user wants to modify sensor value it goes the same way back to Arduino.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter One

1 INTRODUCTION

Imagine how helpful it will be if you are able to switch on/off air conditioning system when you are ten minutes away from home on a hot afternoon in July. How about having a security system that gives you a 24 hour watch out to your home when you aren't there and your kids are home alone ,can detect smoke, fire, any unusual movements, water leakage and give you a warning notification on your mobile, this is what home automation is all about and there is no end to its applications, it can help disabled people to do what they want from their chair without needing any effort, in fact home automation systems are now being developed to maintain inventory of household items, record their usage through an RFID (radio frequency identification) tag and prepare shopping list or automatically order replacement!

Home automation made it possible for you to have what is often referred to as 'Smart Home', the home that waters your flowers, open the light in the way you prefer, open the door for you when you approach it, heat water for bath and tea, stream to you anywhere in the world via the internet a live video to what's going into and around your house. It makes it possible to link lightning, telecommunication, entertainment, security, heating and air conditioning into one centrally controlled system. This allows you to make your house to be an active partner in your busy life.

Nowadays, you can hardly find a house without home automation system which can range from remote for television, burglar alarm and hi-tech security gates to an automated air conditioning system that maintain temperature in a predefined value.

1.1 AUTOMATION

Automation is the use of control systems and information technology to control equipment, industrial machinery and processes, reduce the need for human intervention. In the scope of industrialization, automation is a step behind mechanization provided human operators with machinery to assist them with the physical requirements of work while automation greatly reduce the need of human sensory and mental requirements as well.

Automation plays an important role in global economy and in daily experience. Engineers strive to combine automated devices with mathematical and organizational tools to create complex systems for rapidly expanding range of applications and human activities. Many roles for human in industrial processes presently lie beyond the scope of automation. Human-level pattern recognition and language recognition are well beyond capabilities of modern mechanical and computer systems. Tasks require subjective assessment such as scent and sound as well as high level tasks such as strategic plans, currently require human expertise.

Automation has had a notable impact in a wide range of highly visible industries beyond manufacturing. Once ubiquitous telephone operators have been replaced largely by automated telephone switchboards and answering machines, Medical processes such as primary screening in electrocardiograph or radiography and laboratory analysis of human genes, blood plasmas, cells and tissues are carried out at much greater speed and accuracy by automated systems.

1.1.1 Office Automation

Office automation refers to the varied computer machinery and software used to digitally create, collect, store, manipulate and relay office information needed for accomplishing basic tasks and goals. Raw data storage, electronic transfer and management of electronic business information comprise the basic activities of an office automation system.

1.1.2 Building Automation

Building automation describes the functionality provided by the control of a building. The control system is computerized, intelligent network of electronic devices, designed to monitor and control the mechanical and lighting system of a building. A building automation system is an example of a distributed control system. The building automation system (BAS) core functionality keeps the building climate within a specific range, provides lighting based on an occupancy schedule and monitors system performance and devices failure and provide email and/or text notifications to building engineering staff. The BAS functionality reduces building energy and maintenance costs when compared to non-controlled building.

1.1.3 Power Automation

Power automation is the automated control and monitoring of power plants, substation and transformers for effectiveness, efficiency and fault detection. It has made it possible to have a reliable municipal or national electricity system, which often comprises control remote and hard to reach transformers and power sub systems units. It makes it possible to different power units, relay their status and health information, and even carry out fault detection and correction without human interference.

1.1.4 Home Automation

Home automation may be designate an emerging practice of increased automation of household appliances and features in residential dwellings, particularly through electronic means. home automation includes all that a building automation provides like climate controls, doors and windows control, in addition control of multimedia home theaters, pet feeding, plant watering and so on, But there exists a difference in that home automation emphasis more on comforts through economics and ease of operation.

1.2 PROJECT AIM

The aim of this project is to design and construct a home automation system that will remotely switch on/off any household appliances connected to it, detect human motion, fire, smoke, water leakage and send alarm notification to user on his mobile instantaneously using raspberry pi and Arduino kits, also user can monitor and control the house using website.

1.3 PROJECT OBJECTIVE

The objective of this project is to implement a low cost, reliable and scalable home automation system that can optimize inhabitant productivity, minimize operating costs, improve comfort, simplify use of technologies, ensure security, enhance accessibility.

According to this we will help people to monitor and control their homes remotely, keep track of what is happening now in their homes, keep eyes on their children if they are out the home, Safe the homes from fires, leakage and theft, make it easy for old people to control everything without others' help and it's useful for the persons who are travel a lot to reassure their homes. Finally, it facilities living alone for the physical challenges.

Chapter Two

2 LITRETURE REVIEW

2.1 HISTORY OF HOME AUTOMATION

Home automation has been around since the World War 1(1914), in fact the television remote (a simple home automation system) was patented in1893 (Wikipedia), since then different home automation systems has evolved with a sharp rise after the second World War. Its growth has been through various informal research and designs by technology enthusiasts who want a better way of getting things done at home without effort. The system evolved from one that can automatically switch on or off security lights to more sophisticated ones that can adjust intensity of lighting, and control doors.

2.2 HOME AUTOMATION SYSTEMS

Home automation systems may designate electronic systems at homes and residential buildings that make possible the automation of household appliances. The new stream of home automation systems has developed into a vast one, and the current market is flooded with a flurry of home automation systems and device manufactures.

The types of home automation system based on their control system are:

2.2.1 Individual control systems

These types where the first to hit the market in early years, here each device like the heater or air conditioner will have its own control system.

2.2.2 Distributed control systems

The main feature of these types is emergency shut down; with this system, you can change the control parameters of several similar devices, for example the thermostat of air conditioner and their on/off timing.

2.2.3 Central control systems

These are computerized systems programmed to handle all functions of multiple utilities like air conditioning system, home entertainments, doors, windows, refrigerators and cooking systems, all at same time regardless of whether you are at home or away, you can connect to the control system through internet from anywhere in the world.

The types of home automation systems based on their carrier mode:

2.2.4 Powerline carrier systems

The most affordable type of home automation systems operates over the home's existing wiring or powerline carrier; these can range from X-10 based lamp timers, to more sophisticated systems that require installation by a trained professional.

2.2.5 Hardwired systems

wired or hardwired home control systems are the most reliable and expensive, these systems can operate over high-grade communication cables such as category 5 or 5e, or their own proprietary bus cable. That's why it's better to plan for them during house construction hardwired systems can perform more than one task at a time and make them quickly and reliably, which make them ideal for large homes. They can also integrate more than one system in a home, effectively tying together indoor and outdoor lightning, audio and video equipment, security system, even the heating and cooling system in one control package that will be easy to operate.

2.2.6 Wireless systems

wireless home automation systems are advancements of wired systems, these are great for existing homes as there is no need to run cabling behind the walls, wireless systems are using technologies like IR, ZigBee, Wi-Fi, GSM, Bluetooth, etc. means that its compatible with any existing home network, so it's used to turn on/off lights and devices without needing for wiring.

2.2.7 Internet control protocol systems

Internet protocol (IP) control automation systems use the internet, give each device under its control an internet protocol address and create a local area network (LAN) in the home. Hence, the home can be interacted with over the internet with possibility of live video streaming and real time control.

2.3 HOME AUTOMATION PROTOCOLS

There are wide variety technology platforms or protocols on which a smart home can be built each one is essentially has its own language, and speak to various devices connected to it to make them perform functions. The major and popular protocols are: INISTEON, ZigBee, Z-wave, KNX and universal powerline bus.

2.3.1 INSEON Protocols

INSTEON is a dual-band mesh topology employing ac-power lines and radio frequency protocol to communicate with and automate home electronic devices which normally work independently INSTEON was developed based on the X10 model, for control and sensing applications in home, INSTEON devices communicate with both powerlines and wirelessly.

There are almost 200 different INSTEON-enabled home automation devices available on the market many sold at national retail chains. INSTEON devices don't have to be enrolled into the home automation network; they join the network as soon as they powered up, there is no practical limit to the size of INSTEON network, so it's not unusual to have 400 devices in single installation, all INSTEON devices are peers means that each one can transmit, receive and repeat any message of the INSTEON protocol, without requiring a master controller or complex routing software.

The possible applications of INSTEON: scene and remote control lighting, security alarm interfaces and sensors, home sensors (e.g. water humidity, temperature), access control, audio and video control and appliance management.

2.3.2 ZigBee Protocols

ZigBee is exclusively a wireless home automation protocol. While it claims many home automation enthusiasts, its full acceptance is limited by the lack of interoperability between ZigBee devices, which often have difficulty communicating whit those from different manufacturers, as a result ZigBee is not the perfect choice for everyone unless they use device from just one manufacturer.

ZigBee is a low-cost, low-power, wireless mesh networking standard, the low cost allow the technology to be widely deployed in wireless control and monitoring applications, the power usage allows longer life with smaller batteries and the mesh networking provide high reliability and wide range.

The applications of ZigBee are home entertainment and control (smart lighting control, temperature control, safety and security).
Home awareness (water sensors, power sensors, smoke and fire detectors).
Mobile services, commercial buildings, industrial plants.

2.3.3 Z-wave Protocols

One of the most popular of the wireless home automation protocols, z-wave runs on 908.42 MHZ frequency band, which is a lower frequency band than most household wireless products so it is not affected by their interference and traffic jams, z-wave can talk to all other z-wave devices regardless of type, version or brand.

Z-wave protocol is wireless communication propriety designed for home automation systems, especially to remote control applications in residential and light commercial places, z-wave is a mesh network topology where each node or device in network can send and receive control commands through floors, walls and around household obstacles or radio dead spots that might occur in home, z-wave devices can work singly or in groups and can be programmed into scenes or events that trigger multiple devices, either automatically or via remote control.

Z-wave's low cost and power consumption make it easy to be embedded in consumer electronics products, including battery operated devices such as remote control, security alarms and sensors. Z-wave is currently supported by over 200 manufacturers worldwide. Some common applications of z-wave: Remote home control and management, energy conversion, home safety and security systems and home entertainment.

2.3.4 KNX Protocols

KNX is OSI-based network communication protocol for intelligent buildings; this protocol is based on the communication stack of EIB but enlarged with the physical layers, configuration modes and application experience.

KNX define several physical communication media: twisted pair wiring, powerline networking, radio, infrared and Ethernet, KNX is designed to be independent of any particular hardware platform; a KNX device network can be controlled by anything from a 8-bit microcontroller to a pc according to the needs of a particular implementation. The most common way of installation is over twisted pair medium.

2.3.5 Universal power bus

The universal power bus (UPB) is an industry emerging standard for communication among devices used for home automation. It uses powerline wiring for signal and control. Household electrical wiring is used to send digital data between UPB devices, the UPB communication method consists of series of precisely timed electrical pulses called UPB pulses.

UPB controllers range from extremely simple plug-in modules to very sophisticated whole house home automation controllers, the simple controllers are plug-in controllers that are recommended for a moderate amount of switches and devices as it become cumbersome to control wide range of devices.

2.4 HOME AUTOMATION IMPLEMENTATION PLATFORMS

Home automation can be implemented over plenty of platforms as powerline, Ethernet, Bluetooth, infrared and microcontrollers, each has its own peculiarity and area of application.

2.4.1 Powerline Communication

Powerline communication is a system for carrying data on a conductor also used for electrical power is transmitted over high voltage transmission lines, distributed over medium voltage and used in buildings at lower voltages, powerline communication can be applied at each stage.

All powerline communication systems operate by impressing a modulated carrier signal on the writing system. Different types of powerline communication are different frequency band, depending on the signal transmission characteristics of the power wiring used.

2.4.2 Ethernet

Ethernet defines a number of wiring and signaling standards for the physical connection of two or more devices together. Ethernet was originally based on idea of computer communication over coaxial cable acting as a broadcast transmission medium. The methods used show some similarities to radio systems, although there are fundamental differences, such as the fact that it's much easier to detect collision in a cable broadcast system than a radio broadcast. The common cable providing the communication channel was likened to the ether and it was from the reference that the name "Ethernet" was derived.

Despite the significant changes in Ethernet from a thick coaxial cable bus running at 10 Mbit/s to point to point links running at 1 Gbit/s and above. All the version of Ethernet shares the same frame format and can be readily interconnected, that leads to decrease of cost of the hardware need to support it and reduce the panel space needed by twisted pair Ethernet.

2.4.3 Bluetooth

Bluetooth is an open wireless protocol of exchanging data over short distances from fixed and mobile devices, creating personal area network (PANs). It was originally conceived as a wireless alternative to RS323 data cables; it can connect several devices overcoming the problem of synchronization. It is a slandered and its primary designed for low power consumption, with a short range based on low cost transceiver microchips in each device, so devices can connect to each other and exchange data when they are in the range.

2.4.4 Infrared

Infrared (IR) radiations are electromagnetic radiations whose wavelength is longer than visible light (400-700 mm) but shorter than microwave radiations. Its wavelength spans between 750 nm and 100 microm and is used in short range communications.

Remote control and IrDA devices use infrared light emitting diodes to emit infrared radiations which are focused by a plastic lens into a narrow beam. The beam is modulated e.g. switch on or off, to encode the data. The receiver used a silicon photodiode to convert the infrared radiations to an electric current.

2.4.5 Microcontrollers

Microcontroller is an expensive single-chip computer. Single chip means the entire computer system lie within the confines of the integrated circuit chip. The microcontroller on the encapsulated silver of silicon has features similar to those of our computers. It has the ability to run and stop and perform logic operations and mathematical ones also, so microcontrollers used in automatically controlled products and devices, Microcontrollers accept inputs from one or more device and provide output to other devices in the system, in fact they are responsible for the intelligent in most devices in the market now.

Microcontrollers have two general architecture types that define its mode of operation and design, which are: Von-Neumann architecture, Harvard architecture.

Chapter Three

3 CONNECTVITY AND DATABASE

INTRODUCTION

This chapter explains the communication between mobile interface and web interface to the server, how the data flows from/to raspberry pi and interconnection between the server and the database. The database contains the data of users, homes and all sensor data and variables, so why the database is important.

3.1 OVERALL BEHAVIOR

We have two scenarios, the first is when the user make update from the web or mobile interface, and the other is when the raspberry pi sends and update (i.e. according to alarm detection or user made update through raspberry pi user interface). For the first, we have a mobile interface (i.e. android mobile application) and web interface or website (i.e. the user can access the system using any internet browser), both of them are connected to the webserver which is connected to the database server and the raspberry pi. The raspberry pi is connected to multiple arduino kits via Wi-Fi and the arduino kits are hardwired to all the sensors. When the user updates the status of any object, from the mobile app or from the website, or change the preferred value for any of them, the server will update this data in the database and send it to the raspberry pi instantly via REDIS. Then the server will send a notification to all mobile clients via firebase cloud messaging. The second scenario is when a sensor detects an alarm (i.e. fire or gas leakage), or the user edit a value from raspberry pi GUI then the raspberry pi sends it to the server using http request, and the server will take the data and update the database and notify mobile users for an update occurrence.

To keep our system secure from unauthorized access or intrusion we must authenticate the connection each time a user make request to the server so each user should have an account to access the system securely. For mobile interface each time a user make a request, he must send his token, which he got after login if this token was right the server would give him access and accept his request. For web interface the system checks the user session and data each time he open a page or try to send an update if the system found that this access is not authorized he will break the connection and redirect the user to the login page without giving him access to the system until he login to it again.

Users have accounts so the system should enable them to create new accounts, is it secure? Yes because after they create their accounts the system will not give them access to the resources until an admin approves them, in this case they will be normal users that can have authorized access to the system. The system admins are normal users that have access to the admin panel and can approve, delete, set or unset other users as admins. Therefore, when the user subscribe to the service first time register its home he will get an admin account then he can approve the other users of to the same home.

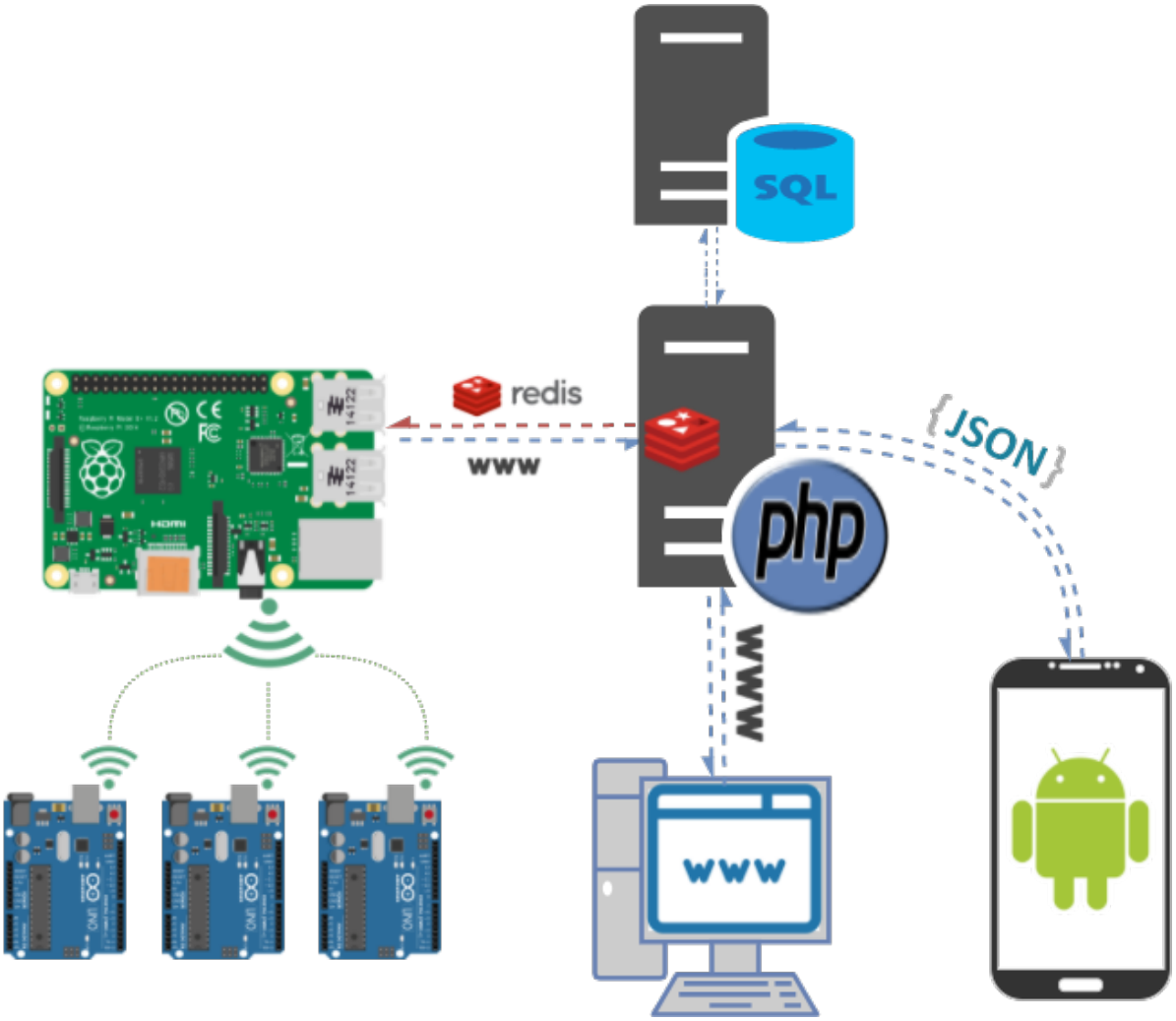


Figure 3-1: System Architecture

3.2 CONNECTION

The webserver is the main core and the backbone of the system that is why nothing can connect to any other part without connecting through the webserver, so any dataflow between the connected parts streams through the webserver. The webserver is a set of scripts written in PHP, which do all the backend, processing in the background, updates the data in the database and keep all parts in synchronization.

We have four routes that the data can flow through them:

a. The webserver can access the database server directly.
b. The website is a web interface to the webserver, which uses HTTP requests to communicate with the webserver. It uses its classes and properties to connect to the other interfaces and update or retrieve data from the database.
c. The mobile interface can communicate with the webserver via internet by the means of REST API (i.e. the mobile application sends HTTP requests to the API of the webserver and receives responses), the mobile application sends the data to the server and receives responses in JSON format.
d. The raspberry pi have two different scenarios for send and receive. For sending, it can establish connections with the webserver using HTTP requests to send data to it. On the other hand, when the server intends to send the data to the raspberry pi it uses REDIS.

3.3 DATABASE

The system database is a relational MySQL database that store all homes, users and sensors data in tables categorized depending on the type. We have two versions of the database one designed for a generic system that can serve many homes, we created the other one to serve one home for simplicity and this is the one applied on the system model. The difference between the two systems is that the first have an extra table for homes that have a relation with the users table and the sensors table, as we will discuss.

Business rules

Users table:

- User can register a new account.
- User can login using username or email and password.
- User can update his data (i.e. username, email, password and phone).
- User can be admin.
- Admin can approve new users.
- User can get access if and only if he is approved.
- Admin can assign new admins.
- Admin can remove assigned admins.
- Admin can remove users.

Sensors table:

- Each sensor does specific function.
- Sensor stands for a sensor or a switch (i.e. actuator).
- Sensors types are four categories (i.e. light, fire, temp., alarm and appliances).
- Each sensor have a state that indicates its status.
- Light sensors have “maxVal” in addition to the state.
- Temp sensors have “preVal” and “curVal” in addition to the state.
- Users can update sensor state, light maxVal and temp preVal.
- Raspberry pi can update alarm state, curVal.
- Raspberry pi and only raspberry pi can update alarm if fire, motion or leakage detected.
- Raspberry pi can update values if the user updated it from the raspberry pi GUI.
- UI can read sensors’ data.
- If value of fire or safety sensor equals one then throw alarm.
- Users cannot reset an alarm.

3.3.1 Database Importance

- Saving users data and contact.
- Saving homes, floors, rooms, sensors names and data.
- Saving the most recent states and values of all sensors.
- If any part of the system breaks down and restarts at any time, it can return the most recent values on startup automatically without need to reconfigure the system, so no need for maintenance.
- The database allows us to achieve synchronization between all parts of the system (i.e. no data inconsistence).
- We can see log of events and requests to monitor the system or maintain it.
- We can get the data of any part of the system independently.

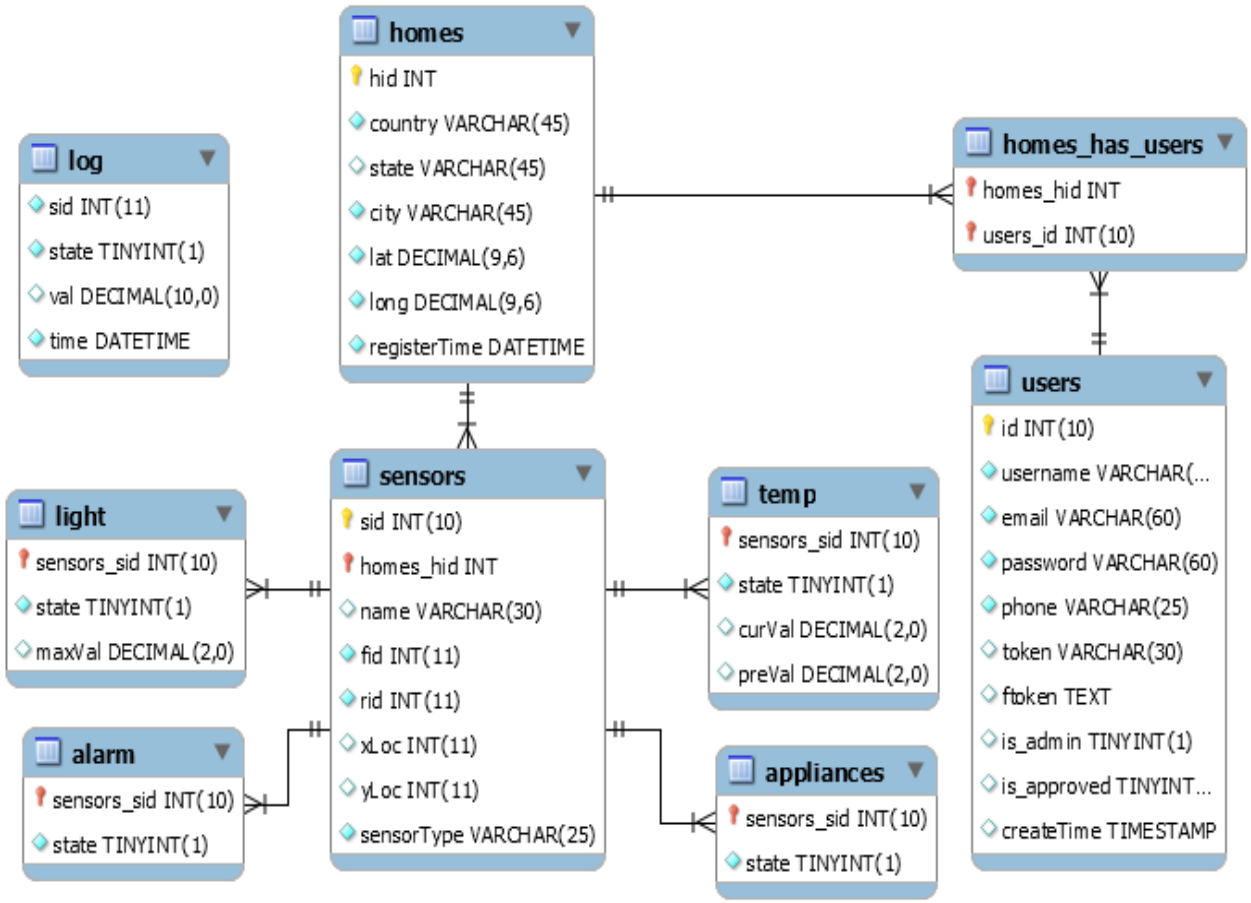


Figure 3-2: General Database ERD

3.3.2 General Database

Tables

As we can see in *Figure 3-2,* the database consists of eight tables:

1. Homes: it has seven attributes that describe each home address and geographical location.
2. Users: it has ten attributes to store the data of each user or homeowner, his contact, and required data for authentication and notification and the time of registration.
3. Sensors: it has eight attributes that describe the name, home, type, and the location where each sensor exists. Note that “fid” is the floor number or floor id, “rid” is the room id, “xLoc” is the horizontal coordinate and “yLoc” is the vertical coordinate.
4. Log: used to store update logs (i.e. when a user updates the state or value of any sensor an entity will be added to the “log” table with the update time. it has four attributes that describe the update and its time, one of them (i.e. Val) may be null if the sensor doesn’t have a value like alarm sensors. This table is helpful to monitor the update history of the sensors. The entries in the “log” table that is older than 7 days are deleted automatically on the occurrence of the any update to the “temp.” table thanks to the “temp_AFTER_UPDATE” trigger.

Then we have four tables as children for the parent table “homes”, each one represents a category of a sensor type (i.e. light, temperature or alarm) and its attributes are the values and states of the sensors at the current time. For all of them the attribute “sensors_sid” is the id of the parent table “homes”:

5. Light: it has three attributes that describe an LED state (i.e. on or off) and value, the “maxVal” attribute is the preferred value to turn off light when the sunlight becomes equal or greater than it, and the state is the current preferred status of the light at a specific place.
6. Temp.: it has four attributes that describe the preferred status (i.e. on or off) of the air conditioning system in a specific room, the preferred value of the temperature and the current value measured by the temperature sensor.
7. Appliances: it has two attributes, id and state that describe the preferred status (i.e. on or off) of the home appliances like (TV, washer, refrigerator).
8. Alarm: it has two attributes like “appliances” table, id and state, which is the real time status of an alarm sensor, which detects if it is safe or dangerous, alarm sensors like (fire, gas or water leakage and intrusion detection).

Relationships

We have six relationships in the database between the tables; “homes” table is the parent of two children “sensors” table and “users” table:

1. Relationship between “homes” table and “sensors” table: is a mandatory one to mandatory many, “homes” table is the parent of the child “sensors” table, this means that every home must have one or more sensors and each sensor belongs to one and only one home.
2. Relationship between “homes” table and “users” table: is mandatory many to mandatory many, “homes” table is the parent of the child “users” table, this means that every home must have one or more users or owners and each user belongs to one or more homes.

The “sensors” table is the parent of the four children (light, alarm, temp., appliances) so it has mandatory one to mandatory many relationship with each one of them. This means that each sensor must belongs to one and only one category, and each category must have many sensors.

3. Relationship between “sensors” table and “light” table.
4. Relationship between “sensors” table and “alarm” table.
5. Relationship between “sensors” table and “temp” table.
6. Relationship between “sensors” table and “light” table.

Because it is many to many we will represent it as a separated table.

Triggers

The database have five triggers in five different tables:

1. “sensors_AFTER_INSERT”: it runs after inserting a new entry in the “sensors” table, it inserts a new entry in the table corresponding type to this sensor.
2. “light_AFTER_UPDATE”: it runs after updating a row in the light table.
3. “alarm_AFTER_UPDATE”: it runs after updating a row in the light table.
4. “temp_AFTER_UPDATE”: it runs after updating a row in the light table.
5. “appliances_AFTER_UPDATE”: it runs after updating a row in the light table.

Each one of the four previous triggers runs after updating sensor type tables and inserts the update in the “log” table. That is how the “log” table stores the updates of the tables. The “temp.” table trigger have an extra function that is not found in the other triggers as it deletes the entries older than seven days from the “log” table.

3.3.3 Applied Database

It is like the general database but it can work for one home only (i.e. it does not have “homes” table) so all the sensors listed in the “sensors” table belongs to the same home. Similarly, all the users listed in the “users” table belongs to the same home. Nevertheless, all the remaining tables are similar to the previous version.

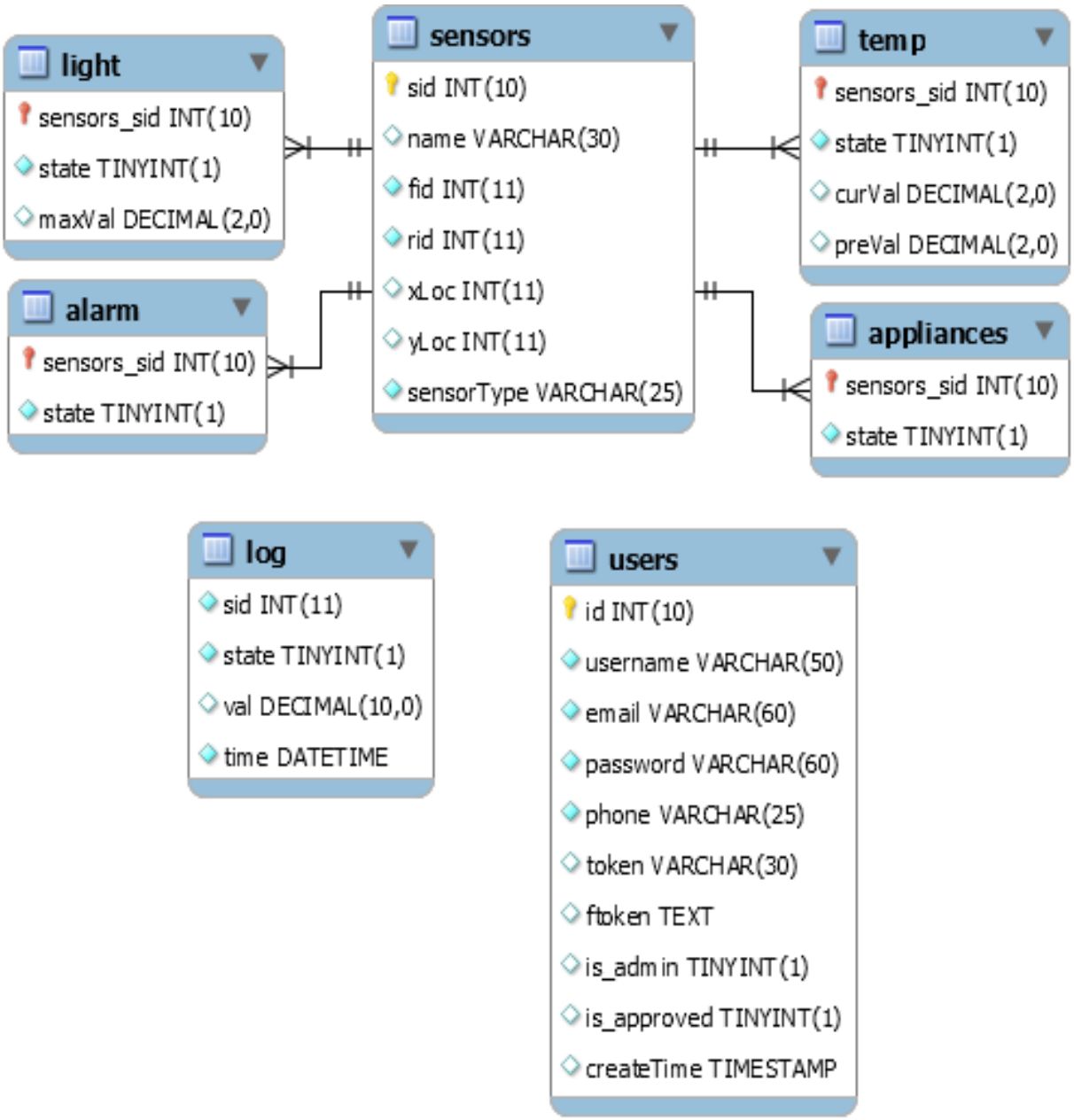


Figure 3-3: Applied Database ERD

There will be some differences in the development between the two versions of the database because of the difference in the ERD but most of these differences will be in the lowest level of the code so we do not need many edits.

3.4 SERVER IMPLEMENTATION

INTRODUCTION

This part explains the implementation of the webserver, how it process the data in the backend and it communicates with the raspberry pi and the other interfaces to deliver the data. In addition, we will explain how it authenticates all the connections, filters the data, blocks the unauthorized access to it and how the users can get authorized access to the server.

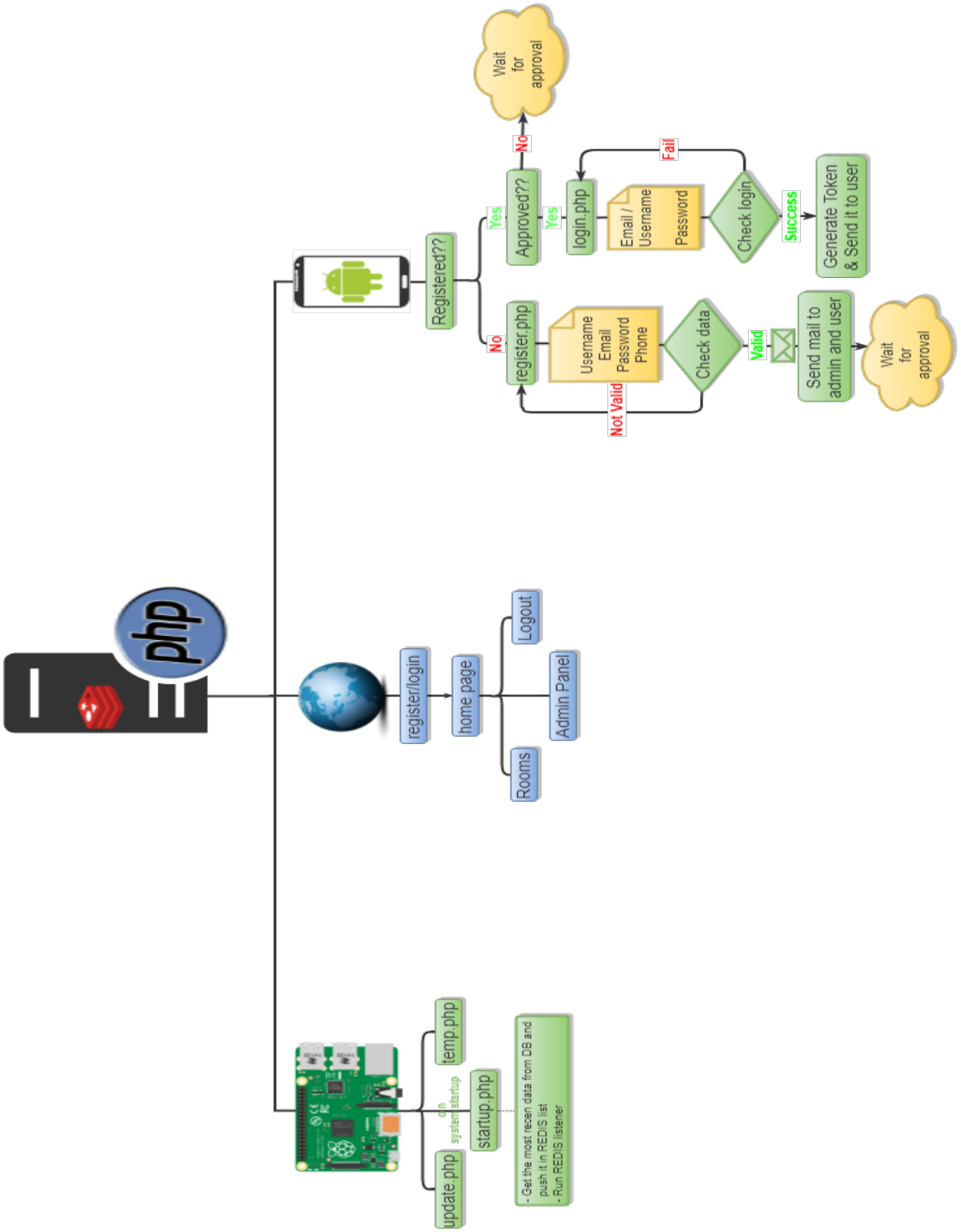


Figure 3-4: Server Flowchart

The server backend side development uses PHP as the main programming language; PHP is one of the best or even the best language for web development as it provide us with many helpful libraries and have many extensions that we can use. As well, it can connect to MySQL databases directly with high-speed connection and that will be helpful.

The webserver consists of a set of pages organized like *Figure 3-4*. As we could see the project directory, (i.e. SmartHome) contains five directories:

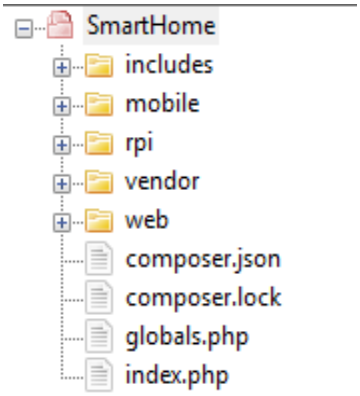


Figure 3-5: Pages Tree

- "includes": contains the core files of the system that is required for running it and no page can load without them like "config.php" which contains parameters required for the database configuration and the models that contains the used classes to interact with the database for storing, updating and retrieving data.
- "web": contains the pages for the web application (i.e. web interface) that we will discuss in the next chapter.
- "mobile": contains the pages that the mobile interface can interact with them, and they represents the mobile API.
- "rpi": contains the pages that the raspberry pi can interact with them for sending and retrieving data. As we mentioned before that the raspberry pi can retrieve data from the server using REDIS.
- "vendor": contains the "predis" extension that allows us to talk with REDIS server from PHP as it cannot talk to it directly.

3.5 INCLUDES

It contains another two directories (i.e. core and models).

Core Directory Contains:

- "config.php": the database configuration.
- "firebase.php": the used class to connect to firebase server API using cURL to send mobile notifications.
- "mysql.class.php": the used class to interact with the database by calling its methods.
- "raintpl.class.php": the used class to interact and view html templates.
- "system.php": contains global functions that we need in almost every page in the project.

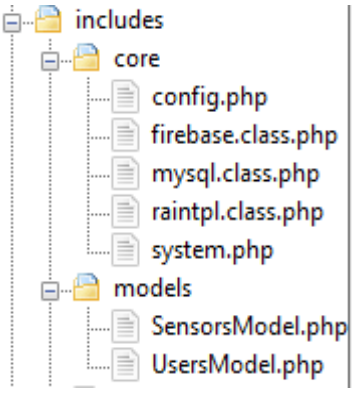


Figure 3-6: Includes directory

Models Directory Contains:

- “SensorsModel.php”: this model is responsible for the database part of the sensors as we can see from its name.
- “UsersModel.php”: this model is responsible for the database part of the users.

All the files inside includes directory are available everywhere on the project. The functions of sensors model and users model functions are allowed everywhere we just need to create an object instance from them and then call their functions.

3.5.1 Sensors Model functions

This model contains ten functions each one does a specific job.

1. get ($extra=’’): get all rows from the parent “sensors” table as array of rows. Each row is an array of single sensor data. It takes one string parameter called “extra” if we want to add a condition.
2. getBySid ($sid): get one sensor data (i.e. single row) from “sensors” table. It takes one integer parameter called “sid”, which is the id of the sensor to get its data.
3. getByRid ($rid): get all sensors data in specific room from “sensors” table. It takes one integer parameter called “rid”, which is the id of the room to get the data of all the sensors found in it.
4. getByFid ($fid): get all sensors data in specific floor from “sensors” table. It takes one integer parameter called “rid”, which is the id of the room to get the data of all the sensors found in it.
5. getSensor ($sid): get state (i.e. and value if exists) of one sensor (i.e. one row) using its id from one of the children tables of the “sensors” table.
6. getRoom ($rid): get states (i.e. and values if exist) of all the sensors in specific room (i.e. array of rows) from the children tables of the “sensors” table.
7. getFloor ($fid): get states (i.e. and values if exist) of all the sensors in specific floor (i.e. array of rows) from the children tables of the “sensors” table.
8. getCat ($sensorType): get states (i.e. and values if exist) of sensors of specific type (i.e. array of rows) from one of the children tables of the “sensors” table.
9. updateSensor ($data): Updates the statue (and the value if exists) (i.e. “light”.”maxVal” or “temp”.”preVal”). It takes one array parameter called “data” which contains the data of the sensor to be updated.
10. piTempUpdate ($data): Update the “curVal” for the sensors in the “temp” table. It takes one array parameter called “data” which contains the data of the sensor to be updated.

The difference between the functions like “getBySid” and “getSensor” is that the first get the data from the “sensors” table, then we get its sensor type, using it, we will get the absolute data of the sensor (i.e. state and value). In other words, we get the data of the sensor in two steps the first when we get its type and the second when we get its data.

3.5.2 Users Model functions

This model contains twenty functions each one does a specific job. These functions' jobs differs between data retrieval, inserting, updating, deleting user, checking approval and admin properties and updating them, checking existence, authorization and login.

In the web, mobile and raspberry pi parts, we will use the implemented functions of the two previous models to achieve the required response, updating the database, sending notifications or sending data to the raspberry pi using these prepared functions.

3.6 WEB APPLICATION

It implements the functions mentioned before in includes and creates views for the users to interact with it, and allows the users to update the sensors and if they are admins they can get access to the admin panel and edit it. It uses JavaScript to give the page the functionality of sending updates and executing scripts without refreshing the page. The next chapter will explain this part in detailed view.

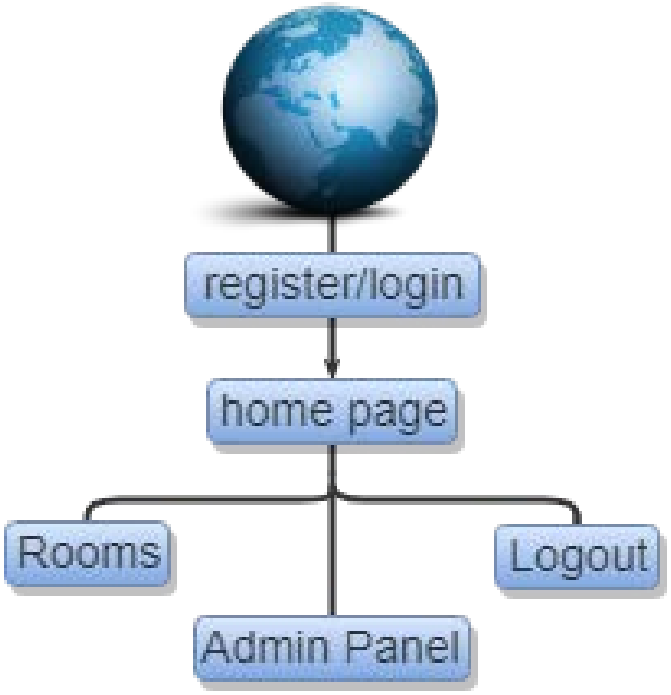


Figure 3-7: Web Application FlowChart

3.7 RASPBERRY PI INTERFACE

The raspberry pi part implements two different scenarios for sending and receiving data. First scenario is when we want to send data from the raspberry pi to the server; in this case, we use HTTP requests (i.e. the program on the raspberry pi that collects the data from the arduino kits will call a PHP script and send it the data for each sensor). Nevertheless, when we want to send the data from the server to the raspberry pi we cannot use the same method (i.e. HTTP request) in this simple way. As the raspberry pi is just a host in a local network, and when it connects to internet it will get a random public IP, which is a dynamic IP. Therefore, if we want to use the same method we must give the raspberry pi a static public IP address, which will require from us to ask ISP for one and pay extra fees, so this method is not practical. For this reason, we looked for another solution that does not require a static IP address and gives us high data streaming speed, and we selected REDIS.

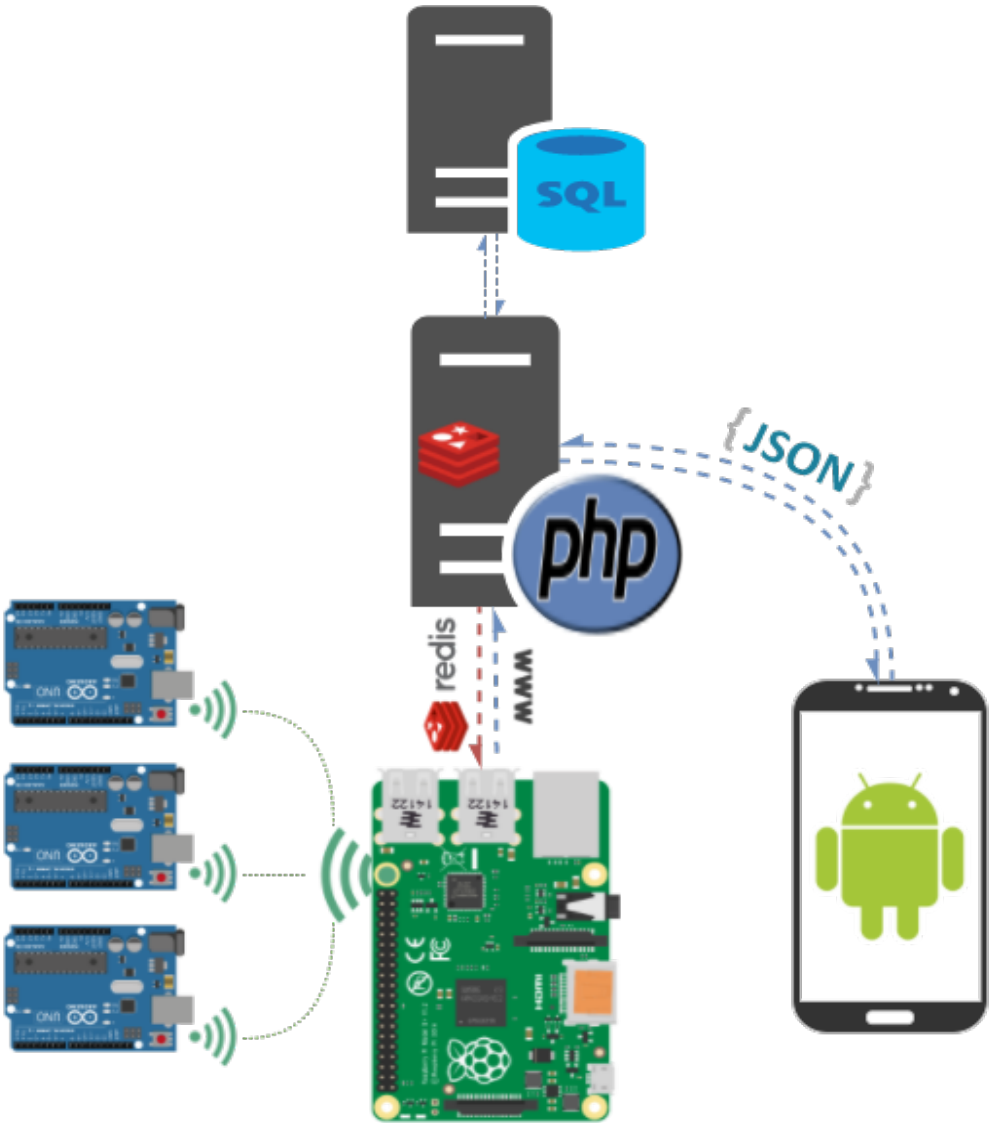


Figure 3-8: Raspberry Pi Connection Diagram

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets and more. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. Redis provide a very high speed and supporst most of the popular programming languages.

In this project we will use REDIS as message queue (i.e. FIFO), the first come entries will be poped first from the queue. When a user make an update the interface sends it to the srever, the server will process this update after data validation and filtering, if all the data are valid it will update these values in the database then it will push it in a REDIS list, on the other side of the raspberry pi there is a listener that listens on the same list and pops any new entries instantly then sends it to a program on the raspberry pi that will send the data to the arduino kits to be applied on its pins.

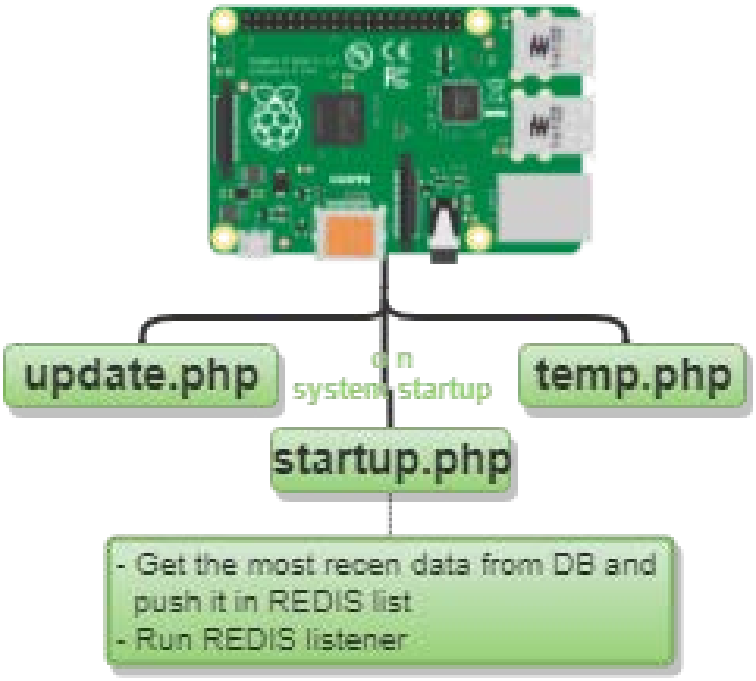


Figure 3-9: Raspberry Pi Flowchart

As we can see in the two figures, we have four PHP scripts on the raspberry pi and three on the server what happens is that the scripts on the two parts can talk with each other.

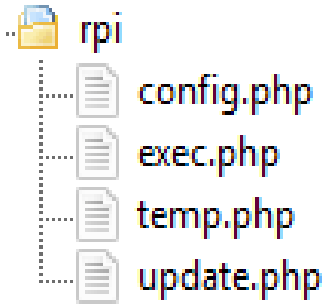


Figure 3-10: Raspberry Pi Scripts

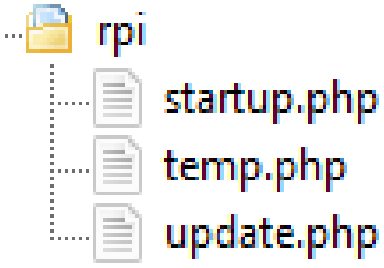


Figure 3-11: Online Raspberry Pi Scripts

On the raspberry pi, we have four files:

3.7.1 **Config** “config.php”:

Configuration file, which contains the URL of the webserver and the REDIS client.

3.7.2 **Update** “update.php”:

The file that the raspberry pi calls to pass sensors’ data to it to be updated.

Algorithm:

1. The raspberry pi calls the script using a shell command and passes the data of sensor to it as arguments (i.e. id, state and value if exists).
2. The script will get the parameters passed to it via shell.
3. The script will call the corresponding script on the online server (i.e. update.php) using curl with GET method.
4. The online script “update.php” will get the data from the URL and push it in an array after checking the required parameters and filtering the data.
5. Then it will use the sensors model “updateSensor” function to update the data in the database.
6. If the previous step executed successfully, it will create a fire base object and use the send mobile notification to all android users after retrieving their fire base tokens from the database.

3.7.3 **Temp** “temp.php”:

this script is like the previous one except it is responsible for updating only the temperature current values by calling the online corresponding script “temp.php” and passing the parameters, which it got, from the shell. The online script will do the same function like the previous one by calling the function “piTempUpdate” and update the database and sends a mobile notification to all the users.

3.7.4 **Exec** “exec.php”:

this script is the listener that listens on the same port at which the server pushes the data as well as it will get the most recent values from the database when it runs at the system startup.

Algorithm:

1. The script runs automatically at the system startup.
2. Initializes the connection with the REDIS server and creates a REDIS client.
3. Calls the online script “startup.php” using curl session.
4. The “startup.php” script will get the data of all sensors from the database and push the data of each sensor in the used REDIS list, which called ‘project' after serializing it, then returns the execution to the “exec.php” script to resume execution.
5. Then the “exec.php” executes a cpp program on the raspberry pi using shell, this script will get the current values of alarm sensors and temperature current values and passes them to “update.php” script to detect if there is an alarm occurred when the system was at sleep.
6. The “exec.php” then runs its listener, which will execute continuously until system break or shut down.
7. When a new update happens the webserver will push it in the ‘project’ list using REDIS client and the “exec.php” will pop it instantly and passes it to the cpp program, which is responsible for broadcasting this value to the corresponding arduino kit.

3.8 MOBILE API

The mobile API provides a set of pages that represent the interface between the webserver and the mobile application. Thanks to this API, the mobile application can communicate with the other parts of the system, update the database, send updates to the raspberry pi and even send mobile notifications to the other mobile clients.

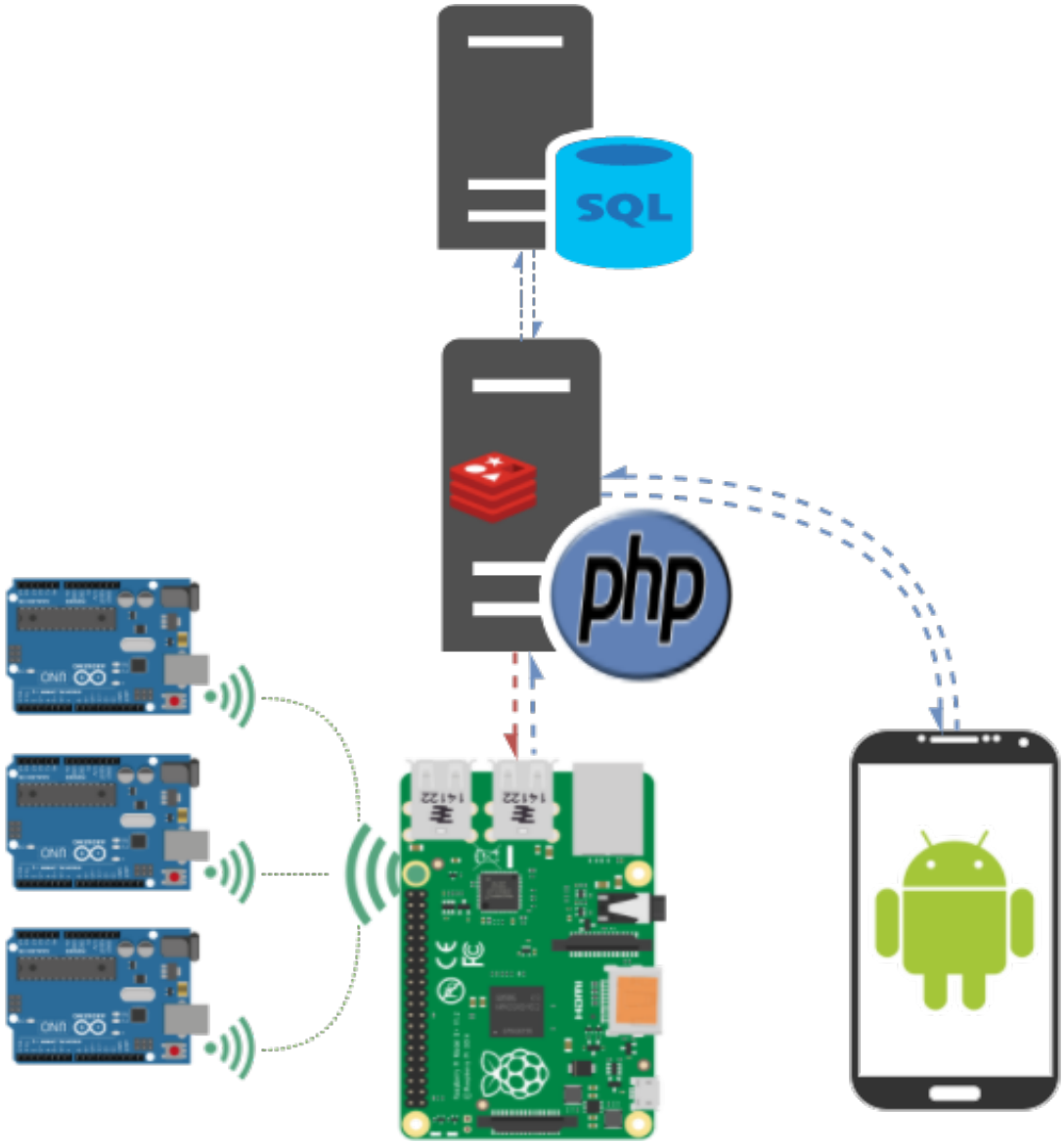


Figure 3-12: API Architecture

The API consists of six pages that provide registering new users, access authorization, retrieving sensors data, updating sensors data, and admin panel. All the mobile pages sends or receives data as JSON objects, other formats are not allowed.

3.8.1 Pages

Register Page “register.php”:

This page is used to register a new user. It receives the data of a new user (i.e. username, email, phone and password). Then it filters the data and validate it using Regular Expression. If the data is proper and valid, it will insert the user data in the database; send a mail to the user to confirm that, send another mail to the admin to let him know of the registration of a new user. After the registration, the user will not get access to the system unless the admin approves him. When the admin approves him, he will receive a mail telling him that he now got access to the system and can use it. It always view success or fail messages to the user telling him what happens now.

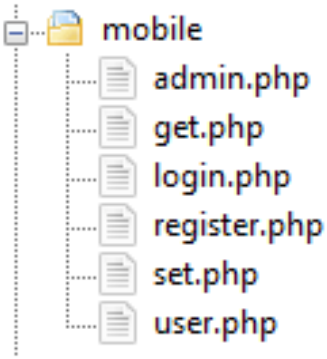


Figure 3-13: API Pages Tree

Login Page “login.php”:

This page is used to start a new user session to give him authorized access to the system. It receives the user data (i.e. username/email, password and his firebase token). Then it will filter and validate the data and checks if it is proper, valid and found in the database then it generates an access token, update it in the database and pass it to the user. Otherwise it will not give him access and breaks the connection. It always view success or fail messages to the user telling him what happens now.

Both of the register and login pages uses the POST method and do not accept other methods. If it was a different method, they will break the connection and display an approciate message.

```
{
    "username": "user123",
    "password": "Password1",
    "email": "user@host.com",
    "phone": "0123456789"
}
```

Figure 3-17: Register Request Example

```
{
    "message": "Error, password can't be empty"
}
```

Figure 3-16: Login Response Example

```
{
    "message": "Your data has been saved successfully, Please wait until the admin approves you!"
}
```

Figure 3-14: Register Response Example

```
{
    "email": "user@host.com",
    "password": "password",
    "key": "fassasddascafsffa8fafasdaf9afaafdsfadfad1fadfheiuhnjnf0uhv2jqnewjh5fndk5da5d41faf"
}
```

Figure 3-15: Login Request Example

Figure 3-18: Register/Login Flowchart

Get Page “get.php”:

This page is used to view the data of one sensor, sensors in the same room, sensors in the same floor or sensors of the same category. It uses the GET method, so if the user used another method it will return an appropriate error message. If the user used GET without no data sent, it will concern him with a message that no data sent, then it checks his token to authorize the connection. If token is right continue otherwise break with a “401 Unauthorized” header message, then checks if all required data are found, then returns the data to the user as JSON object if all the data are authenticated and found in the database. An example request to it is like:

http://localhost/smarthome/mobile/get.php?type=room&id=1&token=8449dd39fb055b3420170703173230

Response:


```
{
    "sensors_sid": "1",
    "state": "1",
    "maxVal": "100"
}
```


Figure 3-19: Get Response

Set Page “set.php”:

This page is used to update the data (i.e. status and val) of one sensor. It uses the GET method, so if the user used another method it will return an appropriate error message. If the user used GET without no data sent, it will concern him with a message that no data sent. Then, it checks the token to authorize the connection, if token is valid continue otherwise break with a “401 Unauthorized” header message. Then checks if all required data are found then send the data to the raspberry pi, update the data of this sensor in the database, and returns the result of the operation as TRUE/FALSE to the user as JSON if everything goes right. At any time if an error occurred send a message to the user tell him about it.

An example request to it is like:
http://localhost/smarthome/mobile/set.php?token=0bd081ba20170422203235&id=3&state=0&val=24

Response:

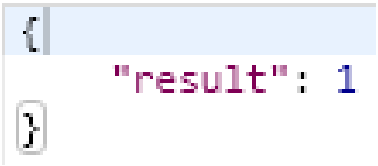


Figure 3-20: Set Response

The next two pages “user.php” and “admin.php” are the backend processing of the admin panel they are like get and set pages but they act on users and only admins can got access to them.

User Page “user.php”:

This page is used to view the data of user(s). It uses the GET method, so if the user used another method it will return an appropriate error message. If the user used GET without no data sent, it will concern him with a message that no data sent. Then it checks the token and admin state to authorize the connection. If the user is admin and token is right continue, otherwise break the connection with a “401 Unauthorized” header message. Then checks if all required data are found (i.e. id and type), then get the data of the user(s) from the database and returns the data to the user as JSON if everything goes right. If an error occurred send a message to the user tell him about it.
An example request to it is like:
http://localhost/smarthome/mobile/user.php?token=8449dd39fb055b3420170703173230&type=new

Response:

Figure 3-21: User Response

Admin Page “admin.php”:

This page is used to update the data (i.e. approve/admin) of one user. It uses the GET method, so if the user used another method it will return an appropriate error message. If the user used GET without no data sent, it will concern him with a message that no data sent. Then it checks the token and admin to authorize the connection. If admin and token is right continue otherwise break with a “401 Unauthorized” header message. Then checks if all required data are found (i.e. id and op). Then update the data of the user in the database, send a mail to the user if the op was equals “approve” or “set_admin” and returns the result of the operation as TRUE/FALSE to the user as JSON if everything goes right. At any time if an error occurred send a message to the user tell him about it.

An example request to it is like:

http://connectvia.netii.net/mobile/admin.php?op=approve&id=51&token=0bd081ba20170422203235

Response:

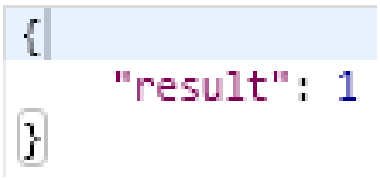


Figure 3-22: Admin Response

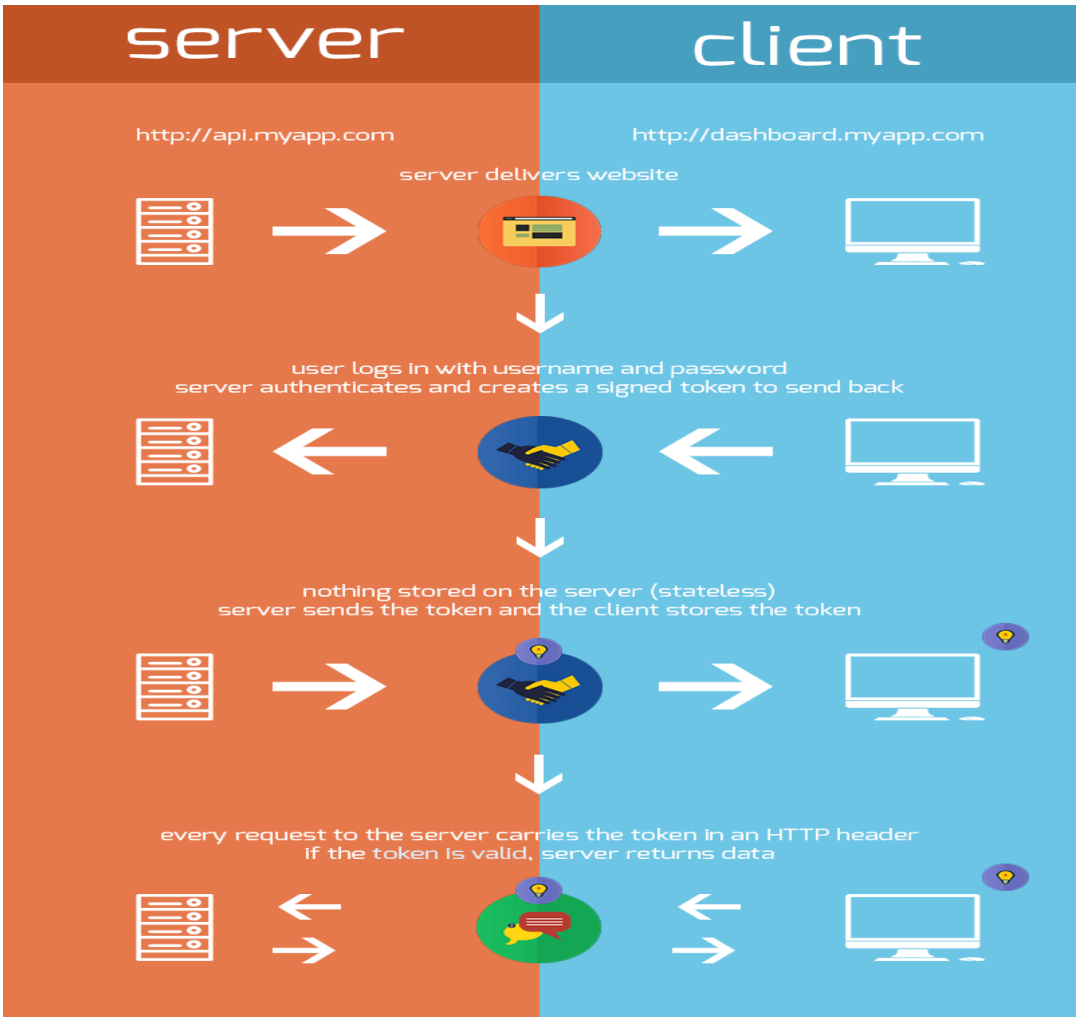


Figure 3-23: Token Authentication

3.9 The system behaviour after login

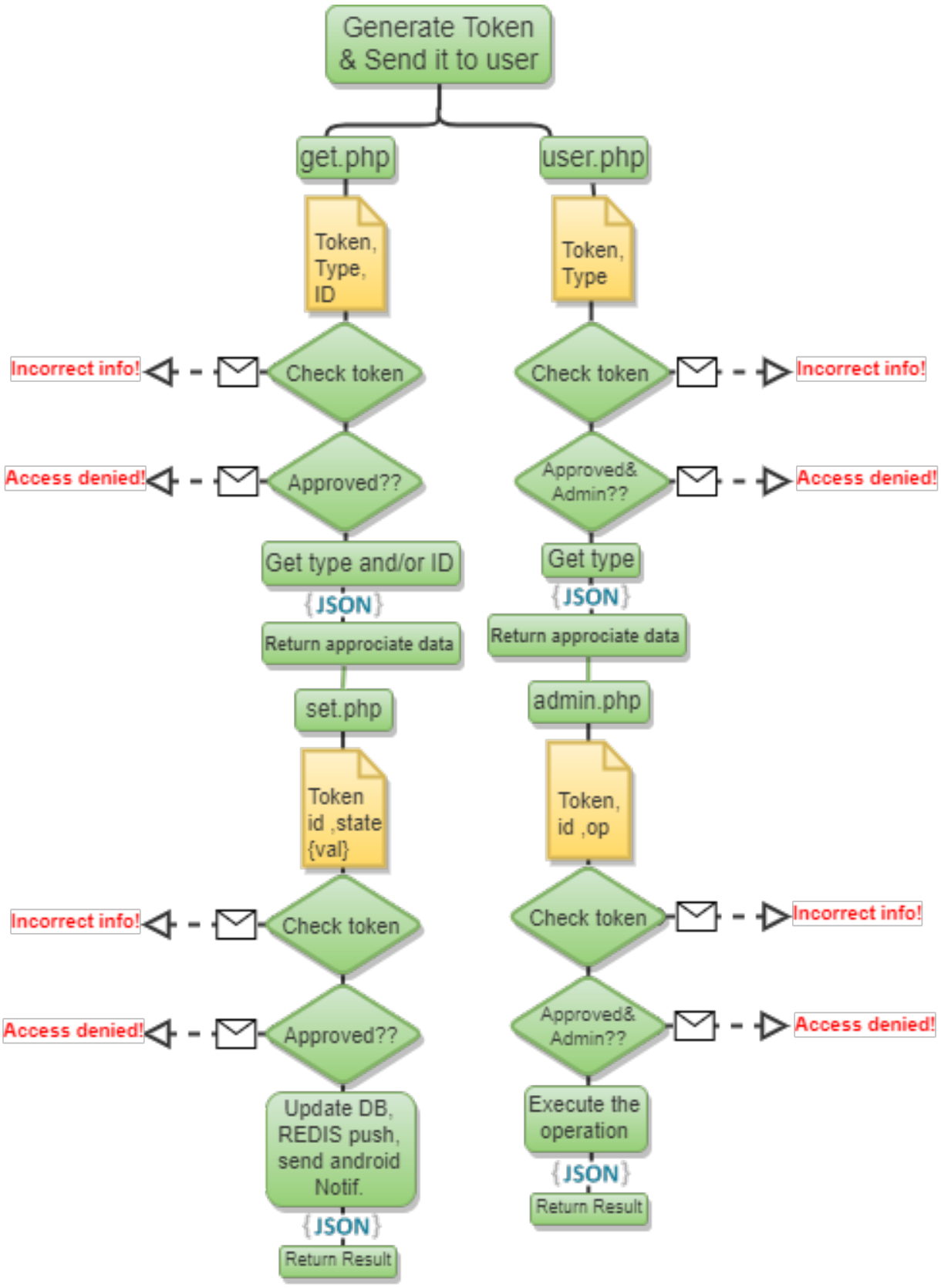


Figure 3-24: Mobile after Login Flowchart

3.10 The overall flowchart of the mobile API

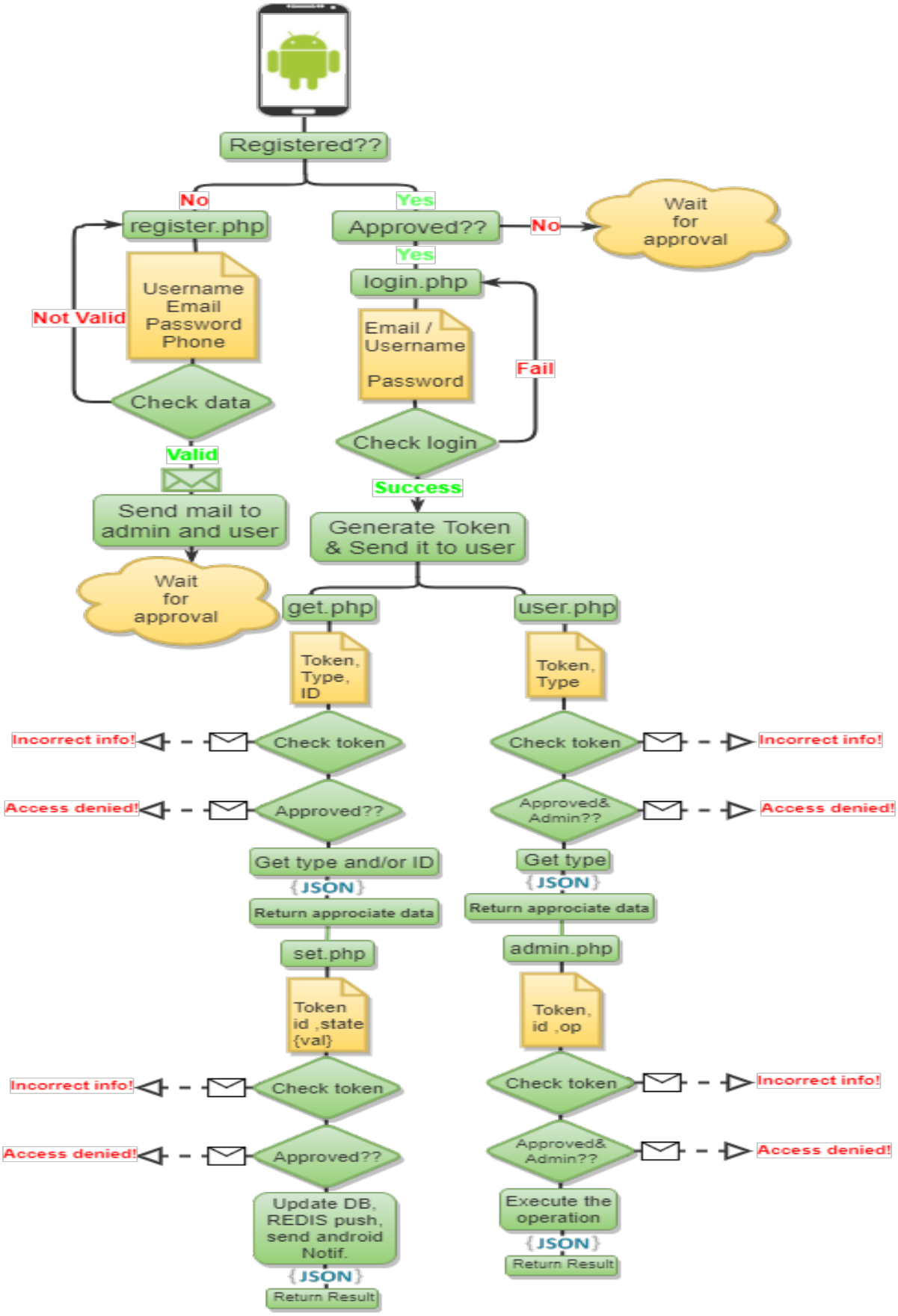


Figure 3-25: Full Mobile API Flowchart

3.11 The overall flowchart of the server

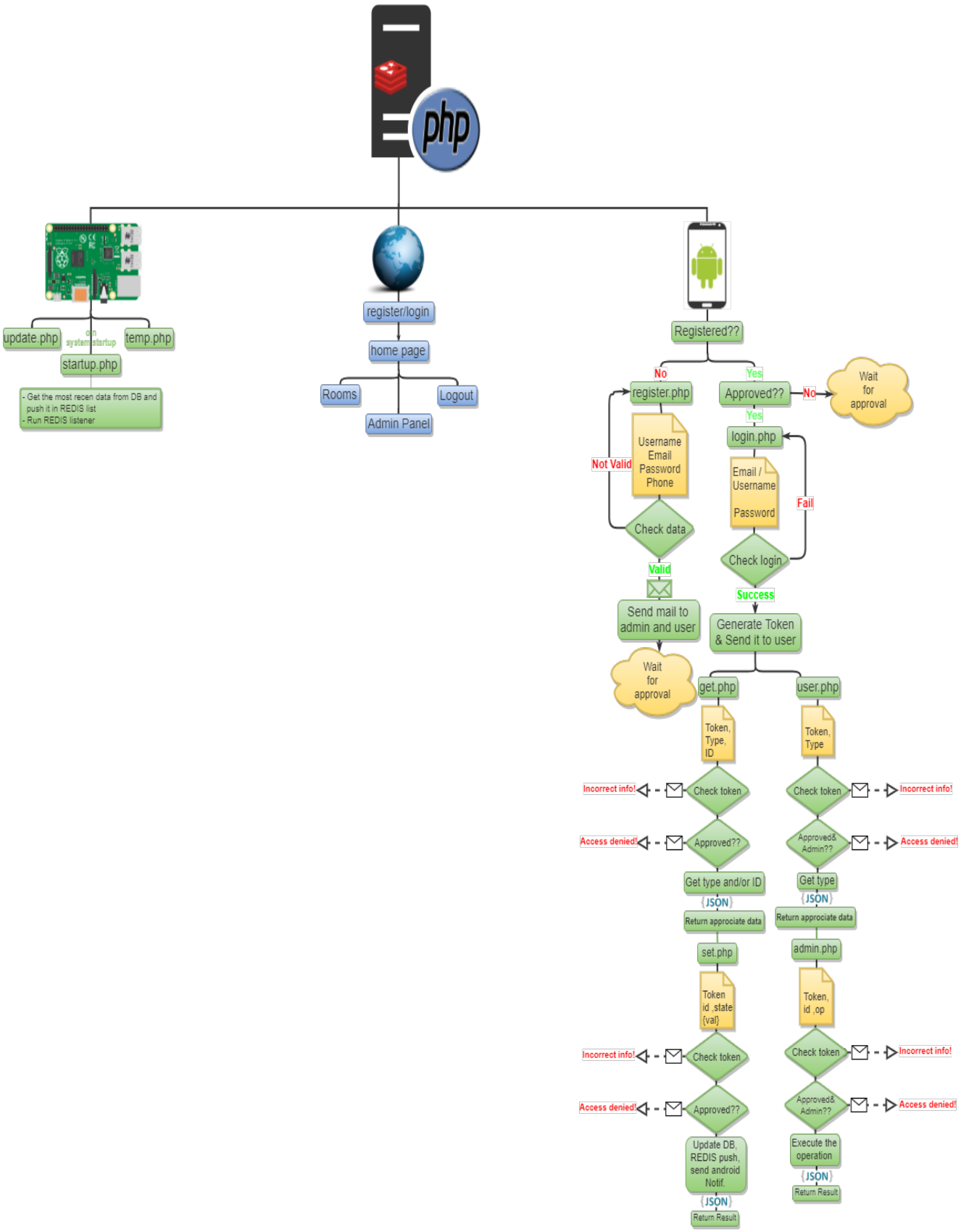


Figure 3-26: Overall Server Flowchart

Chapter Four

4 CONTROL VIA WEBSITE

INTRODUCTION

Controlling home has never been easier! With a website, you have the ability to control and monitor all the sensors inside your house. Be aware of what's happening inside your house even when you're not there! So even if you're out there and forgot to turn off the lights, you simply can log into the website via any internet browser to monitor the status of Light and control the lighting in each room with one click (ON-OFF Switch). You can also adjust the temperature of the air conditioning system in your house in a range from 15°C to 40°C. Forgot to close the doors? - Don't worry, you can do that too! On the website, you can check the status of the home door as well, and set it with one single click to ON to open the door or OFF to close it. You can also check the safety of your house on the website through a fire detection system, water leakage and gas leakage detection systems. More beautiful thing is any change happens in that website, it takes effect on the system's android application that is to be explained further in chapter six. Both the website and the android application work in synchronization; if the user changes any value in the website this value will be changed automatically in the android application and vice versa. The mobile application will receive notification for this change. The user interface of the website is simple and powerful.

4.1 LOGIN PAGE

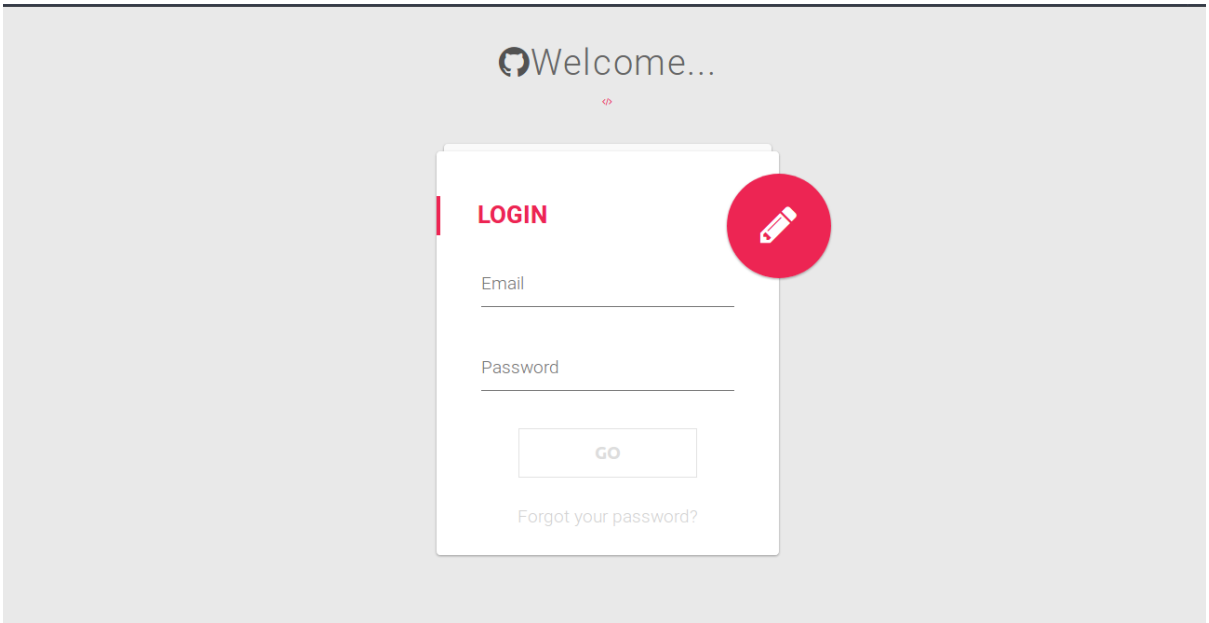


Figure 4-1: Login page

4.1.1 Overview

This page consists of:

- **Heading title:** For “Welcome…” text and “Login” text.
- **Input field type email:** where the user can enter his email.
- **Input field type Password:** where the user can enter his password.
- **Button:** to log in into the administration page of the website.
- **Another Button:** To register if one hasn’t already.

4.1.2 Handling user interaction

- When the user clicks on the Go button, PHP checks first if input fields are not empty, if the user left any of the two fields blank, it will inform him that both fields are required to login.
- After filling both fields and hitting enter or Go button, the PHP will convert that password to SHA1 encryption and go for the user table in the database on the row holding that email and see if the entered password (after the encryption) matches the encrypted password in the database.
- If the corresponding password doesn't match, an error message will appear telling that user has entered invalid data, and therefore can't have the access to the administration page.
- Otherwise, user will successfully login to the administration page of the website.
- Session is stored in the browser and users have a ranking system.
- An admin will have slightly more control than the regular user.
- Regular user can view only the Rooms page and take control from there.
- However, an admin has more permission. An admin can approve new users or remove any user if needed, and add another user.

4.2 REGISTER PAGE

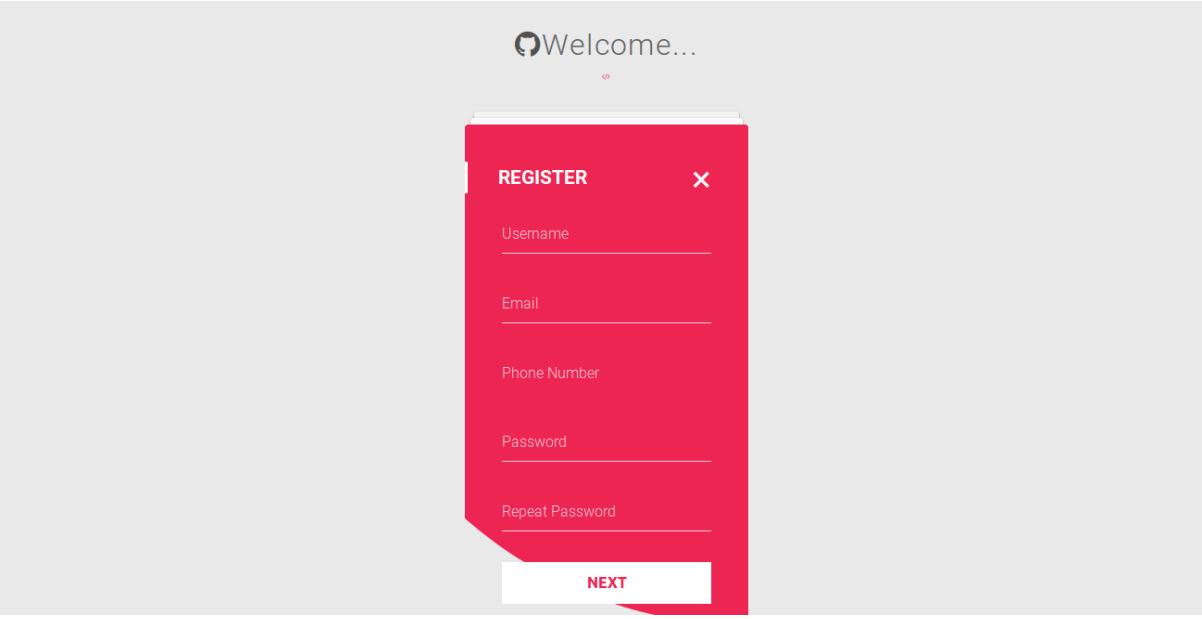


Figure 4-2: Register Page

4.2.1 Overview

The page consists of:

- **Input field type text:** where the user can enter his username.
- **Input field type email:** where the user can enter his email.
- **Input field type tel:** where the user can enter his phone.
- **Input field type password:** where the user can enter his password.
- **Another input field type password:** where the user is required to repeat his password to make sure he doesn't forget it.
- **A button:** to create the account.

4.2.2 Handling user interaction

- When the user clicks on sign up button, PHP will check if the password length is at least 8 characters and at least has 1 uppercase, 1 lowercase, 1 number with minimum 1 special character to make sure that user has entered a strong password.
- Otherwise, the user will see a message “Invalid Password”, the user has to enter the valid password.
- Then it will check if the password entered in the password field is identical to the password entered in the repeat password field. If not, the user will see message “Please confirm your password”.
- Then username, email, phone and password will be put in an array on the server side and inserted as a new row in the user table of the database.
- If the user is successfully registered, he will receive message to notify him “Successful registration, wait until the admin approves you”.

4.3 ROOMS PAGE

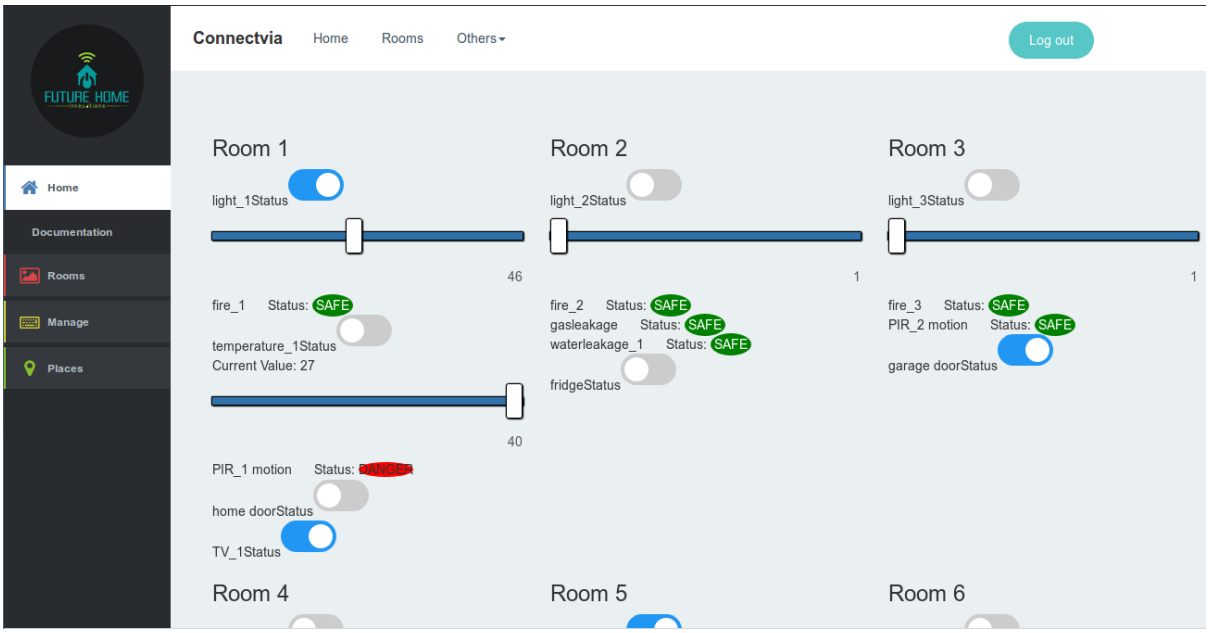


Figure 4-3: Rooms Page

4.3.1 Overview

All pages include 3 constant files for page styling:

- **Header:** to view the navbar on the top of the page with quick navigation links.
- **Sidebar:** to view all the navigation links on the left side of the page.
- **Footer:** to contain copyrights.

The content of this page is:

- **All rooms:** shown as a table of 3 columns and n rows.

  Each room has:

- **Toggle switch for light:** to turn the lights on or off inside a specific room.
- **Range slider:** to control the intensity of the light.
- **Status of fire:** to detect whether the home is safe or on fire.

  Some rooms have:

- **Toggle switch for temperature:** to turn the air conditioner on or off inside a specific room.
- **Range slider:** to control the degree of temperature.
- **Status of gas leakage:** to detect whether the home is safe or some gas leakage detected.

- **Status of water leakage:** to detect whether the home is safe or some water leakage detected.
- **Fridge status:** To close or open the door of the fridge.
- **TV status:** to turn the television on or off.
- **Home door status:** to turn the home door on or off.
- **PIR motion status:** to detect if there is unusual motion in the house.

4.3.2 Handling user interaction

- First off, the page checks if user is logged in or not. If logged in, user will see the page. If not logged in, user will be redirected to the login page.
- Each room takes an id and then PHP loop through each room by the given id, incrementing the id of the room every loop to fetch the data of the next room.
- Inside each loop, data of the sensors are fetched from the database via an associative array using a key and a value.
- Some sensors are meant to display the current state only (Safe or Danger), other sensors allow the user to adjust the value (switch on/off and adjust values)
- When the user clicks on the toggle switch of any sensor, it gets the state of the switch.
- If it is checked the state will be 1 (ON).
- If it is not the checked, state will be 0 (OFF).
- The value of the seek bar of the light is stored and sent to the database.
- Every sensor is assigned to a specific ID that is to be sent with its state (from the switch) and value (of the seek bar) to the database.

4.4 ADMINISTRATION PAGE

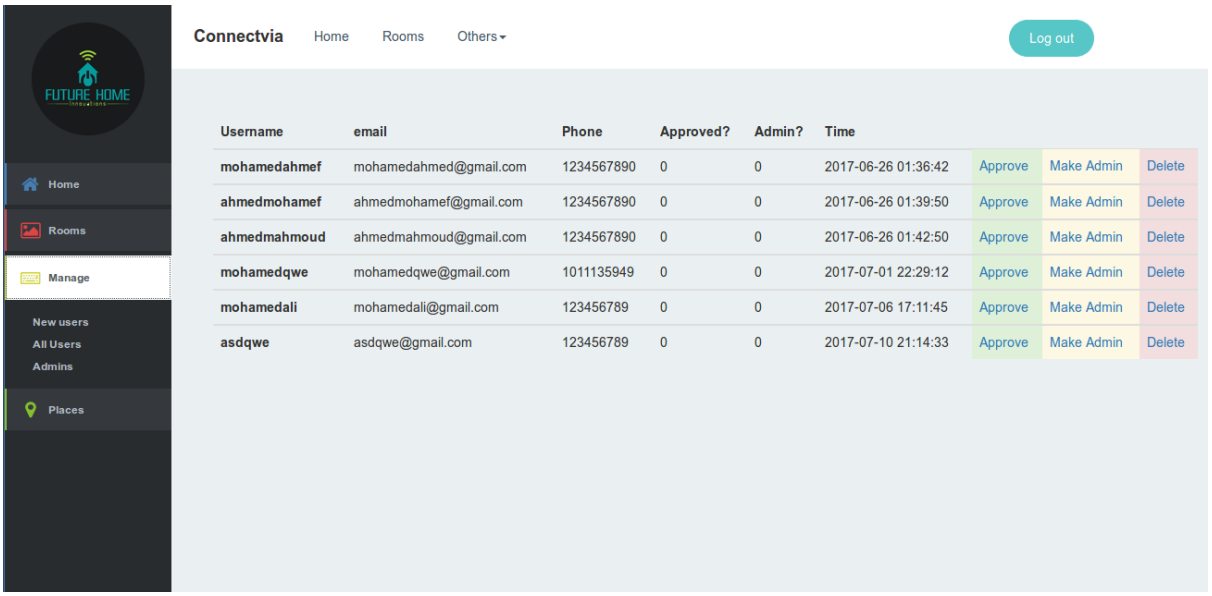


Figure 4-4: New User Page

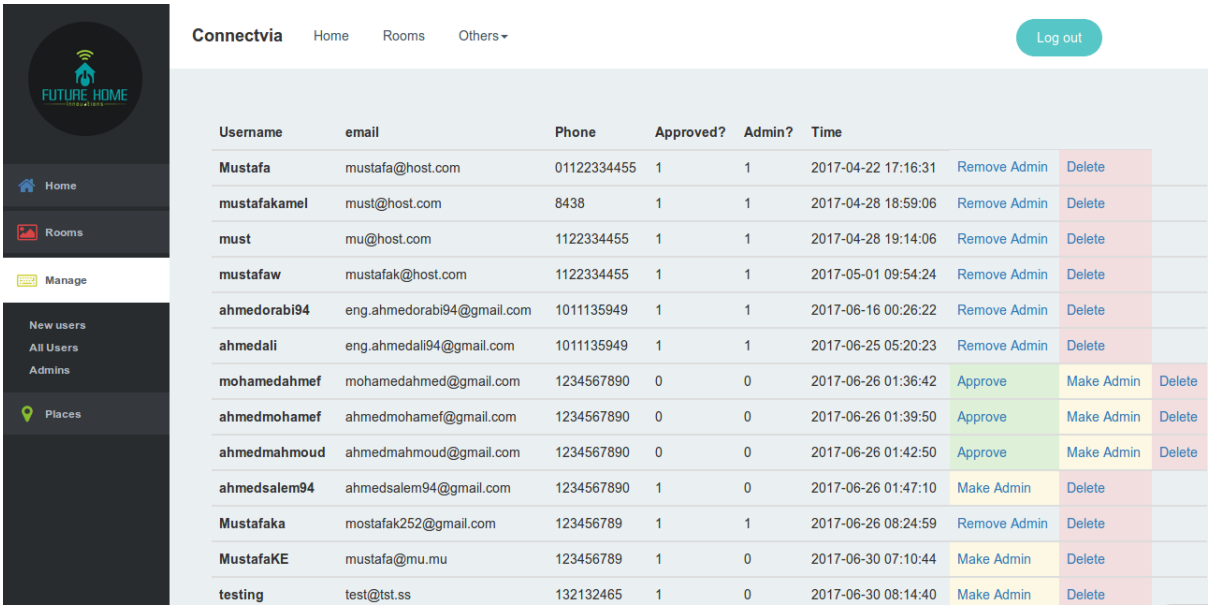


Figure 4-5: All Users Page

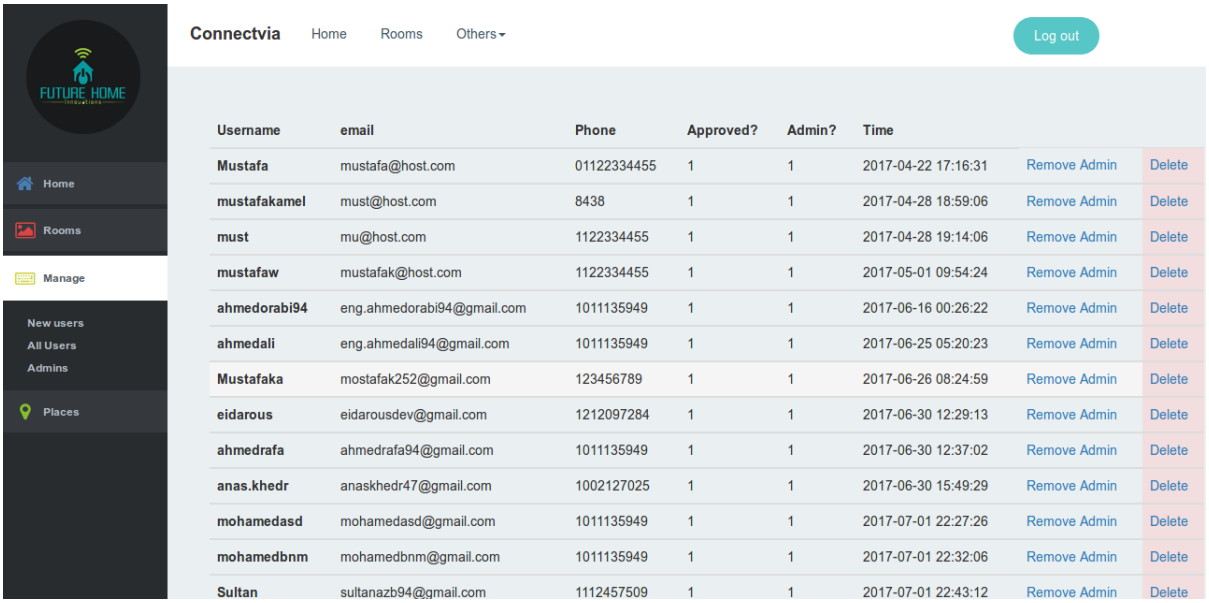


Figure 4-6: Admins Page

4.4.1 Overview

This page displays:

- **New users:** display the new registered awaiting approval users with the ability to approve these users, make them admins, or even delete them.
- **All users:** display all the registered users (The approved and awaiting approvals).
- **Admins:** display all admins with the ability to remove admins and make them as regular users or even delete the entire account.

4.4.2 Handling user interaction

- The page has a function that checks if the logged in user is a regular user or an admin.
- If the logged in user is a regular user, he will be redirected to the home page, and the menu “Manage” on the sidebar will be hidden.
- Menu “Manage” only appears for Admins.
- New users are the users who are not approved yet.
- Admins are the users who can approve new users, make user an admin and have the permission to delete any user.
- When the admin clicks on the New Users page, PHP sends a query to the database in the users table where column is_approved = 0 to fetch and display awaiting approval users.
- It gets the username, e-mail, phone registration time, is_admin? and is_approved? and display them in a table.
- If the admin clicks on the delete button, the data of this user will be deleted from the database.
- The same thing happens when the admin hits the admins and all users item.

4.5 FLOWCHART

Figure 4-7: Server-side Flow Chart

Chapter Five

5 MOBILE APPLICATION

INTRODUCTION

This android application will be used to monitor and control the house. It can help elderly or handicapped people live a more independent life as long as possible. This application will allow the user to control a device that is connected to any home appliance. The users are able to switch on/off the lights and switch on/off air condition system when you are ten away from home. The users could check the status of TV and door. Suppose you aren`t at home you can monitor the status of fire, water leakage and gas leakage. This application can give the users notifications on their devices if another user changes the status of any object like TV or door. The application and the website work in synchronization if the user changed any value in the website this value will be changed automatically in the application and vice versa and the mobile application will receive notification for this change.

The user can also check the status of the outside light and turn on and off the light without the need to get out of the bed. This application would also benefit users with limited mobility that may have a difficult time getting to or even reaching their light switch. The user interface of the application is simple and powerful.

5.1 LOGIN ACTIVITY

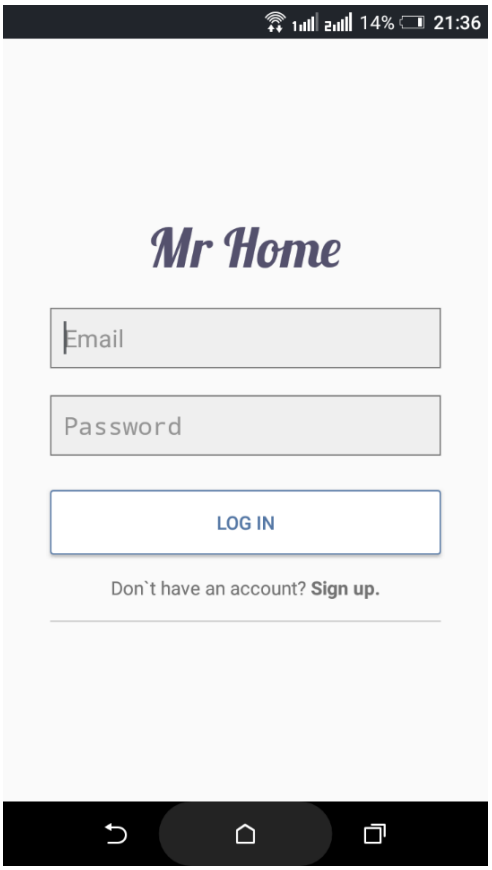


Figure 5-1:Login Screen Activity

5.1.1 Overview

This activity consists of:

- **Edit Text for email:** where the user can enter his email.
- **Edit Text for password:** where the user can enter his password.
- **Button:** to log in into the application.
- **Text view:** Sign up to create new account.
- **Text view:** for “Mr. Home” text.

5.1.2 Handling user interaction

- When the user clicks on log in button, it checks first if there is network available in the mobile, if there is not network available it will display message to the user “please check your internet connection”.
- If there is network available, it gets the email and password from Edit Text and we have the generated Firebase token which we will use to send notifications to the devices.
- It puts email, password and firebase token in JSON object, JSON data has key-value pairs.
- It sends the JSON data to the server, the server will save Firebase token in the database.
- The server will check of the email and password of the user if they exist in the database or not.
- If they exist the server will send token to the mobile application and the user will login successfully and he will go to next activity, if they doesn`t exist the user will receive message “email and password are wrong”.
- The token that the mobile application receives, the application will use it to get data from the server and set data.
- It will store the token and email of the user in shared preferences because it will need to get this data from another activity or fragment.
- It stores the email of the user who logins in because it will need it later to check if this user is an admin or not.
- It will check the email of the user with admins data in the database, to decide if the user can view the page of the admin or not.

5.2 REGISTER ACTIVITY

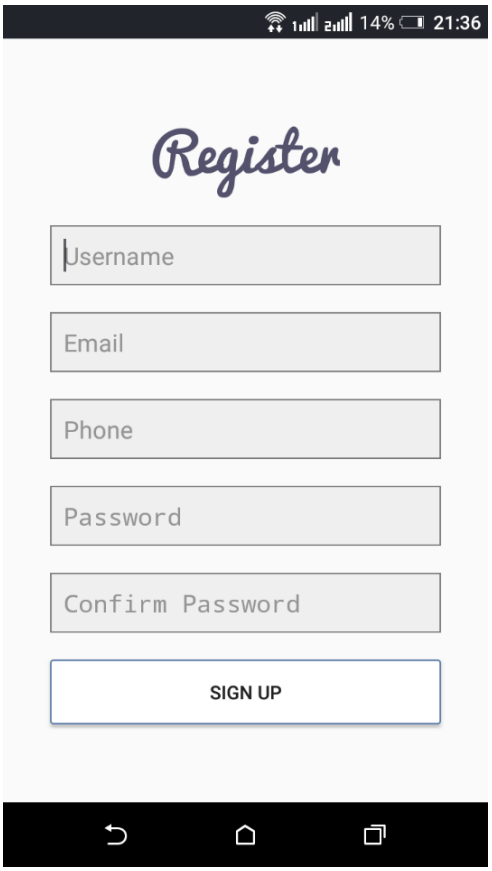


Figure 5-2: Register Screen Activity

5.2.1 Overview

The activity consists of:

- **Edit Text for username:** where the user can enter his username.
- **Edit Text for email:** where the user can enter his email.
- **Edit Text for phone:** where the user can enter his phone.
- **Edit Text for password:** where the user can enter his password.
- **Edit Text for confirm password:** where the user can enter his confirmed password.
- **Button:** to create new account.
- **Text view:** for “Register” text.

5.2.2 Handing user interaction

- When the user clicks on sign up button, it will check if there is network available in the mobile, if there is not available network it will display message to the user “please check your internet connection”.
- If there is network available, it will get the username, email, phone and password from Edit text.
- It will check if the password length is at least 8 characters and at least has 1 uppercase, 1 lowercase, 1 number and at least 1 has special character.
- If the password is wrong, the user will see a message “Invalid Password”, the user has to enter the valid password.
- Then it will check if the password is equal to the confirm password, if not the user will see message “Please confirm your password”.
- Then it will put username, email, phone and password in JSON object, then it will send the data to the server.
- If the user registers successfully he will receive message to notify him “Successful Register, wait until the admin approves you”.
- If the user enters invalid username or email he will receive message to notify him what the error is.

5.3 DRAWER ACTIVITY

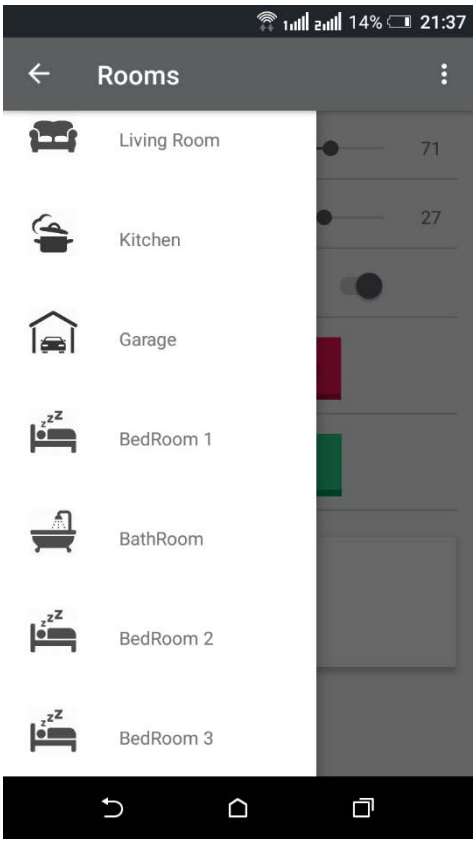


Figure 5-3:Rooms Activity

5.3.1 Overview

This activity consists of:

- **Frame Layout:** to view the room with its sensors and alarms.
- **List View:** consists of the rooms of the house, each item contains text and image for the room.

5.3.2 Handling user interaction

- When the user clicks on an item from the list it opens new fragment related to this item.

  If the user clicks on living room item, list view will disappear and open fragment called living room fragment, its layout contains switches, seek bar and toggle button for each sensor.

5.4 ROOMS

5.4.1 Living room fragment

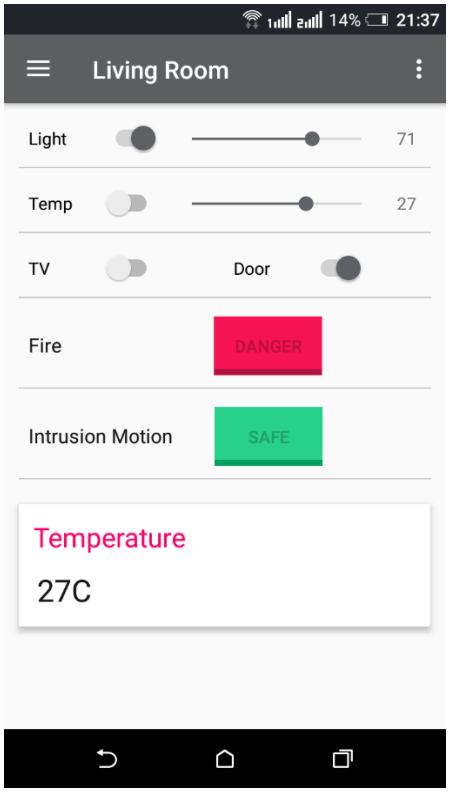


Figure 5-4:Living Room Fragment

5.4.1.1 Overview

This fragment consists of:

- **Switch for light:** to turn on/off the light.
- **Switch for temperature:** to turn on/off the air conditioning system.
- **Switch for TV:** to turn on/off the television.
- **Switch for door:** to open/close the door.
- **Seek bar for light:** to determine the intensity value of the light.
- **Seek bat for temperature:** to determine the degree of the temperature.
- **Toggle button for fire:** to detect if there is fire in the house or not.
- **Toggle button for motion:** to detect if there is unusual motion in the house.
- **Text view:** to display the value of seek bar of the light.
- **Text view:** to display the value of seek bar of the temperature.
- **Text view:** to display the current temperature.

5.4.1.2 Handling user interaction

- When the user opens the living room fragment, it connects to the server gets the states and the values of the sensors and we receive this data in JSON syntax.
- JSON data has key-value pairs, we give it the key “State” or “seek value” and we get the value of the state 1 or 0 and get the value of the seek bar.
- First we check the state if 1 the switch will be checked if 0 the switch will be unchecked, we check the state of the toggle button, if 1 the toggle button will display “Danger” if 0 the toggle button will display “Safe”.
- When the user clicks on the switch of the light, it gets the state of the switch, if it is checked the state will be 1 if not the state will be 0 and gets the value of the seek bar of the light. Every sensor has an id special to it, then it will send the id of the sensor, state of the switch and value of the seek bar to the server.
- When the user clicks on the switch of the TV, it gets the state of the switch, if it is checked the state will be 1 if not the state will be 0 and it has no seek bar so the value will be 0. Every sensor has an id special to it, then it will send the id of the sensor, state of the switch and value 0 to the server.
- The two toggle button the user cannot change its state, it will show the state only. If the state 1 it will display “Danger”, if 0 it will display “Safe”.
- When the user clicks on or changes the Seek bar progress, it checks first the state of the switch related to this seek bar and gets the state, it sends the value of the state and the value of seek bar progress to the server.

5.4.2 Kitchen fragment

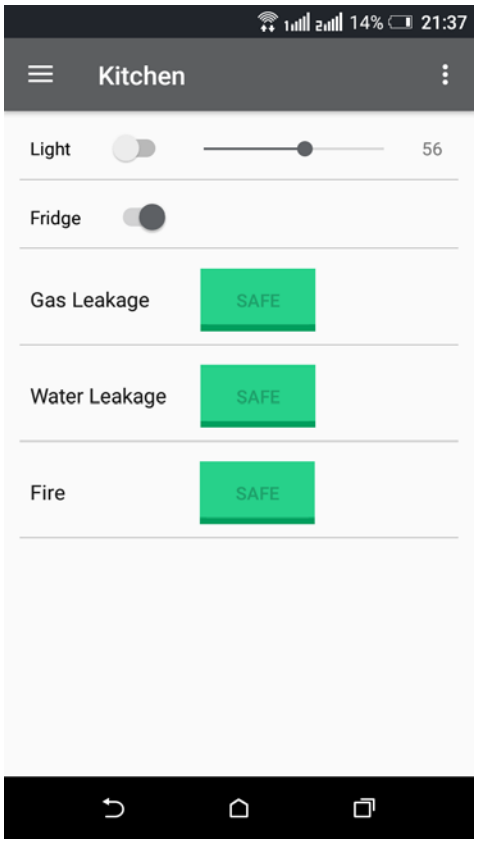


Figure 5-5:Kitchen Fragment

5.4.2.1 Overview

This fragment consists of:

- **Switch for light:** light to turn on/off the light.
- **Switch for fridge:** to open/close the fridge.
- **Seek bar for light:** to determine the intensity value of the light.
- **Text view:** to display the value of the seek bar of the light.
- **Toggle button:** to detect the fire state “Danger” or “Safe”.
- **Toggle button:** to detect the water leakage state “Danger” or “Safe”.
- **Toggle button:** to detect the gas leakage state “Danger” or “Safe”.

5.4.2.2 Handling user interaction

The interaction the same as in the living room fragment.

5.4.3 Garage fragment

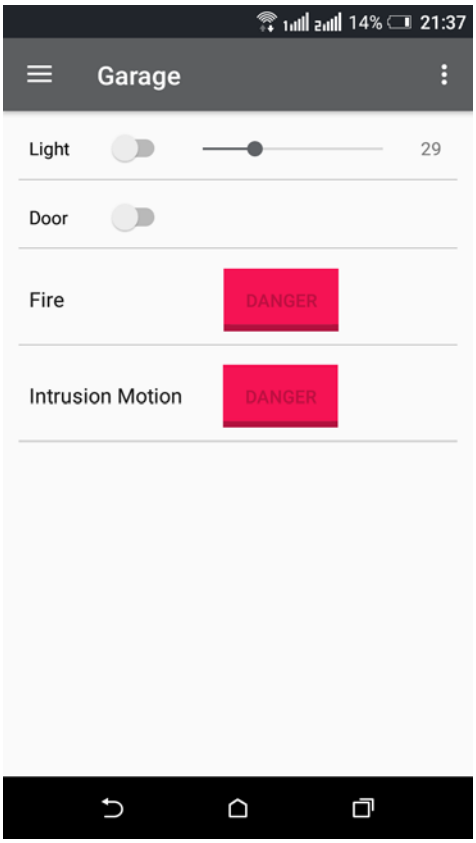


Figure 5-6:Garage Fragment

5.4.3.1 Overview

This fragment consists of:

- **Switch for light:** light to turn on/off the light.
- **Switch for door:** to open/close the door.
- **Seek bar for light:** to determine the intensity value of the light.
- **Text view:** to display the value of the seek bar of the light.
- **Toggle button:** to detect the fire state “Danger” or “Safe”.
- **Toggle button:** to detect unusual motion state in the house “Danger” or “Safe”.

5.4.3.2 Handling user interaction

The interaction the same as in the living room fragment.

5.4.4 Bedroom fragment

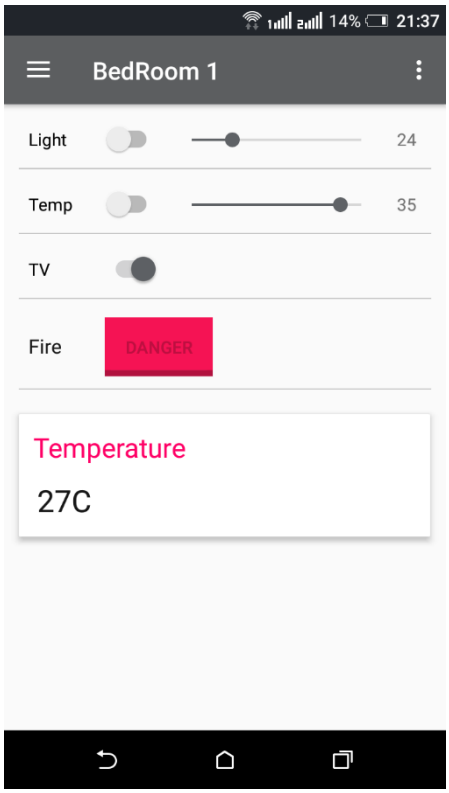


Figure 5-7:Bedroom Fragment

5.4.4.1 Overview

This fragment consists of:

- **Switch for light:** to turn on/off the light.
- **Switch for temperature:** to turn on/off the air conditioning system.
- **Switch for TV:** to turn on/off the television.
- **Seek bar for light:** determine the intensity value of the light.
- **Seek bar for temperature:** to determine the degree of the temperature.
- **Text view for light:** to display the value of seek bar of light.
- **Text view for temperature:** to display the value of seek bar of temperature.
- **Text view:** to display the current temperature.
- **Toggle button:** to detect if there fire in the house or not by displaying “Safe” or “Danger”.

5.4.4.2 Handling user interaction

The interaction the same as in the living room fragment.

5.4.5 Bathroom fragment

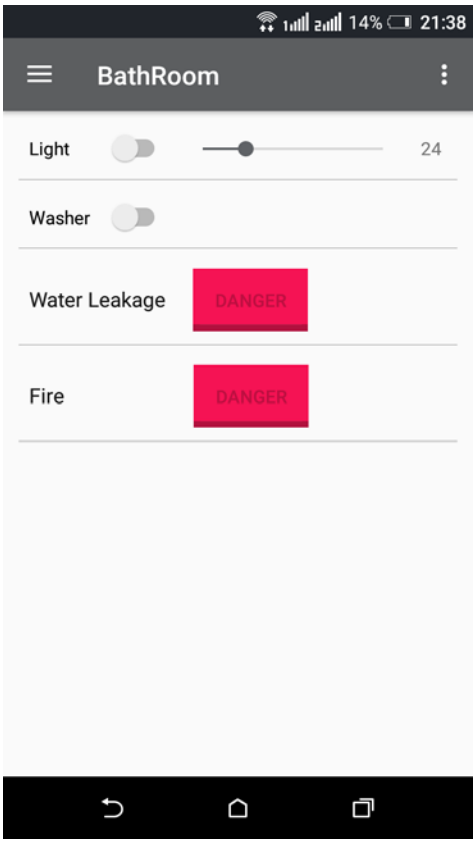


Figure 5-8: Bathroom Fragment

5.4.5.1 Overview

This fragment consists of:

- **Switch for light:** light to turn on/off the light.
- **Switch for washer:** to turn on/off the washer.
- **Seek bar for light:** to determine the intensity value of the light.
- **Text view:** to display the value of the seek bar of the light.
- **Toggle button:** to detect the fire state “Danger” or “Safe”.
- **Toggle button:** to detect water leakage state “Danger” or “Safe”.

5.4.5.2 Handling user interaction

The interaction the same as in the living room fragment.

5.5 MAIN MENU

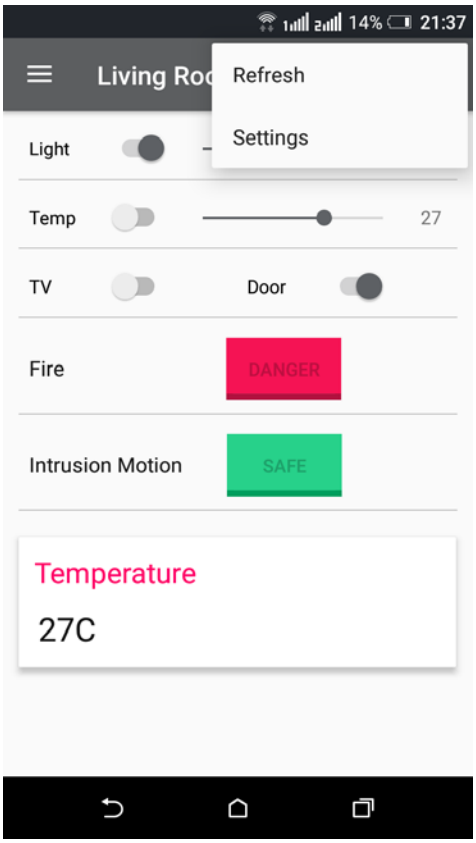


Figure 5-9: Main Menu

5.5.1 Refresh

- When the user hits the menu button the option menu items will be displayed.
- The option menu items are refresh and settings.
- Refresh item is used to refresh the current fragment, when the user hits the refresh item, it connects to the server and gets the updated states and values of the sensors and updates the current fragment`s switches, seek bar and toggle button. The same thing when the user opens any fragment.

5.5.2 Settings

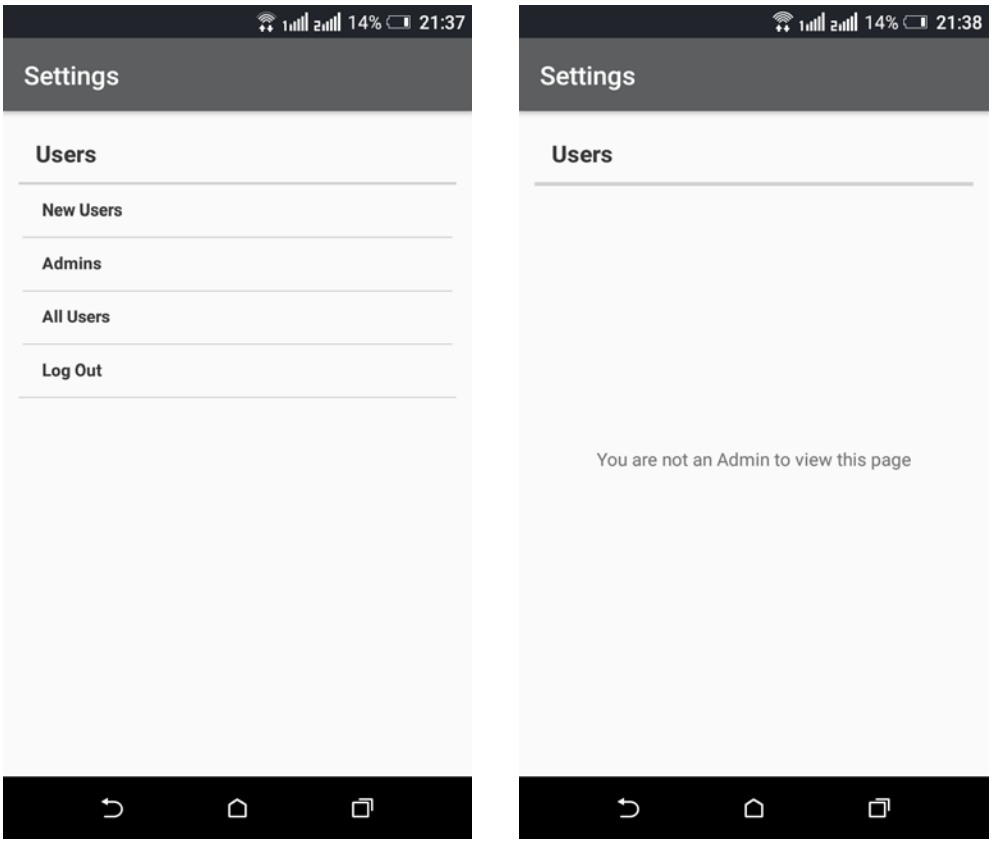


Figure 5-10: Settings Activity

- When the user hits the settings item, new activity will be opened, this activity is designed for the admin.
- First it checks the email of the user who logins in the application with admins data in the database.
- If the email of the user is not equal to the email of any admin in the database, then the user is not an admin, the new activity will display message “You are not an admin to view this page”.
- If the email of the user is equal to the email of any admin in the database, the new activity will display list view has 4 items new users, admins, all users and log out.
- New users are the users who are not approved yet, Admins are the users who can approve new users, make user an admin and can delete any user, all users are new users and admins, log out to log out from the application and go to login screen.

5.5.2.1 List view

- When the admin hits the new Users item, it connects to the server and gets new users data in JSON syntax.
- It gets the id, username, email, is admin and is approved of the user from JSON data and saves it in array list.
- If the user is an admin, check box of admin will be checked if not check box will be unchecked.
- If the user is approved, check box of approve will be checked if not check box will be unchecked.
- When the admin clicks on the check box of admin, first checks the state of the check box, if it is checked the admin will make this user an admin, if it is not checked this user will be not an admin.
- When the admin clicks on the check box of approve, first checks the state of the check box, if it is checked the admin will make this user approved, if it is not checked this user data may be will be deleted.
- When the admin clicks on delete button, the data of this user will be deleted from the database.
- The same thing happens when the admin hits the admins and all users item.

5.5.2.2 Activities of new user, admins and all users

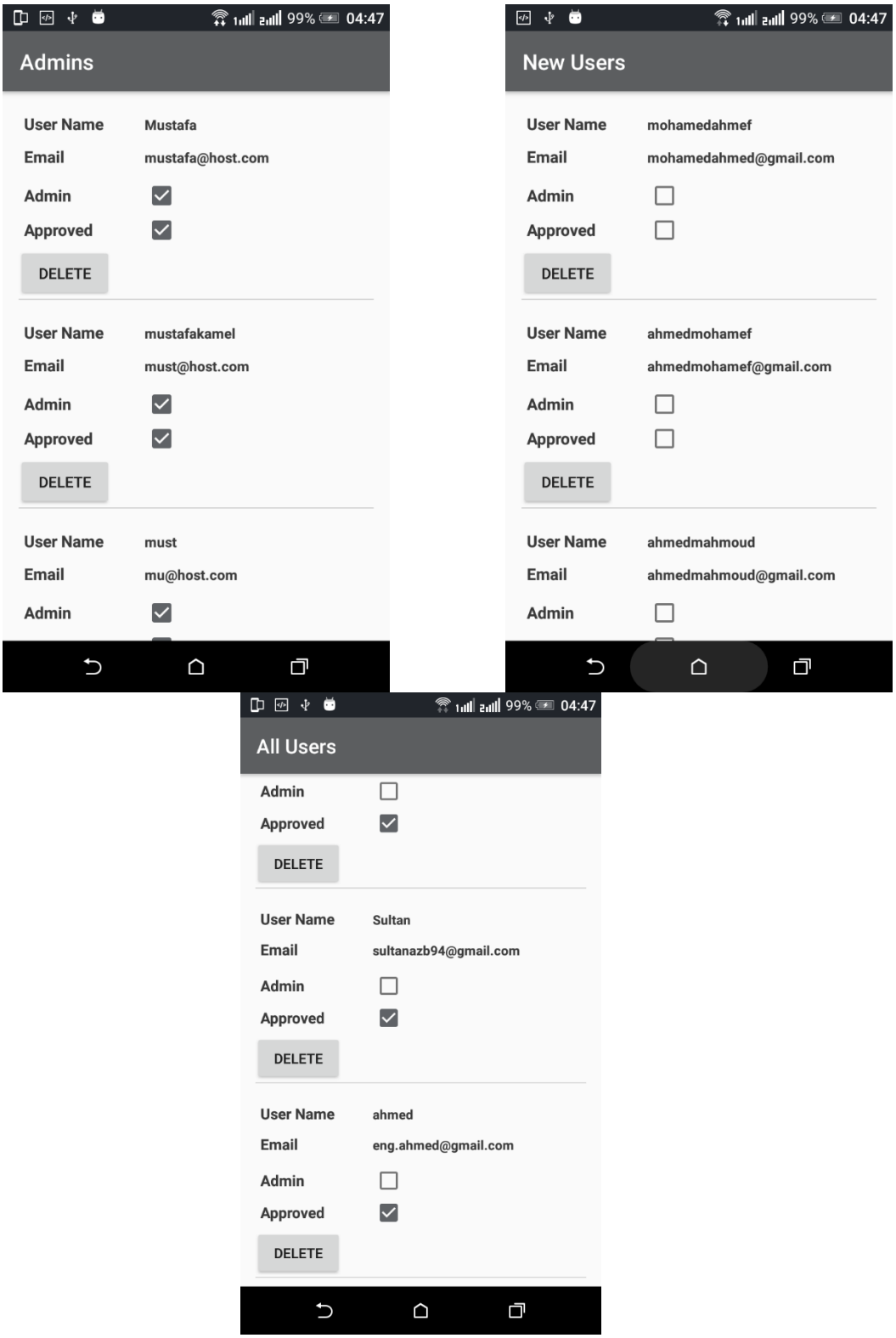


Figure 5-11: All Users Activity

5.6 NOTIFICATIONS

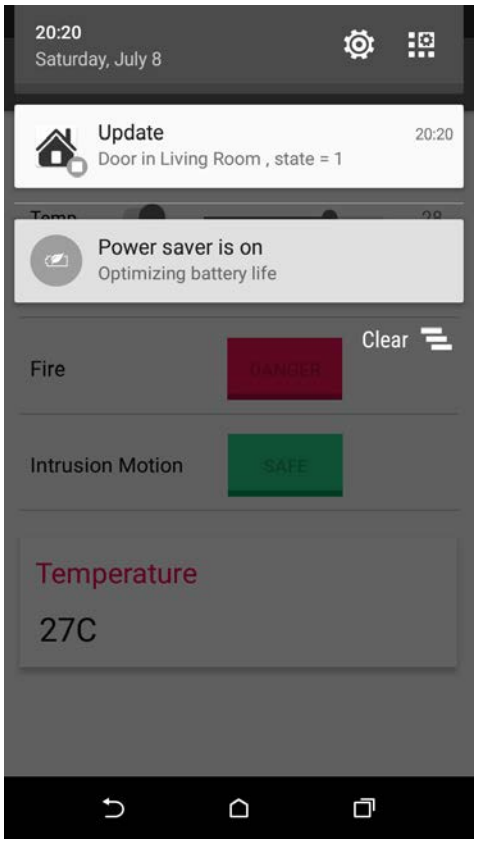
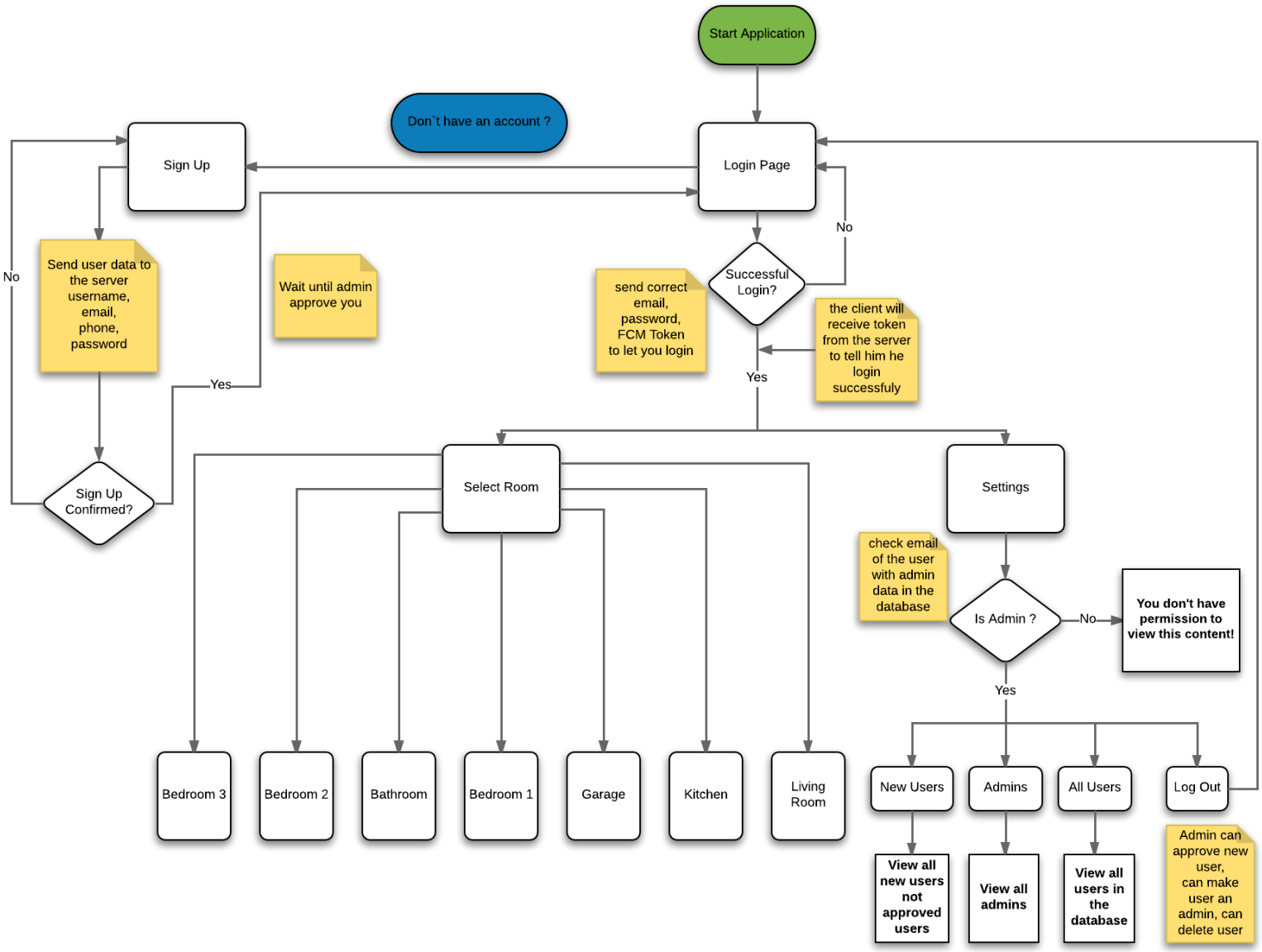


Figure 5-12: Notifications

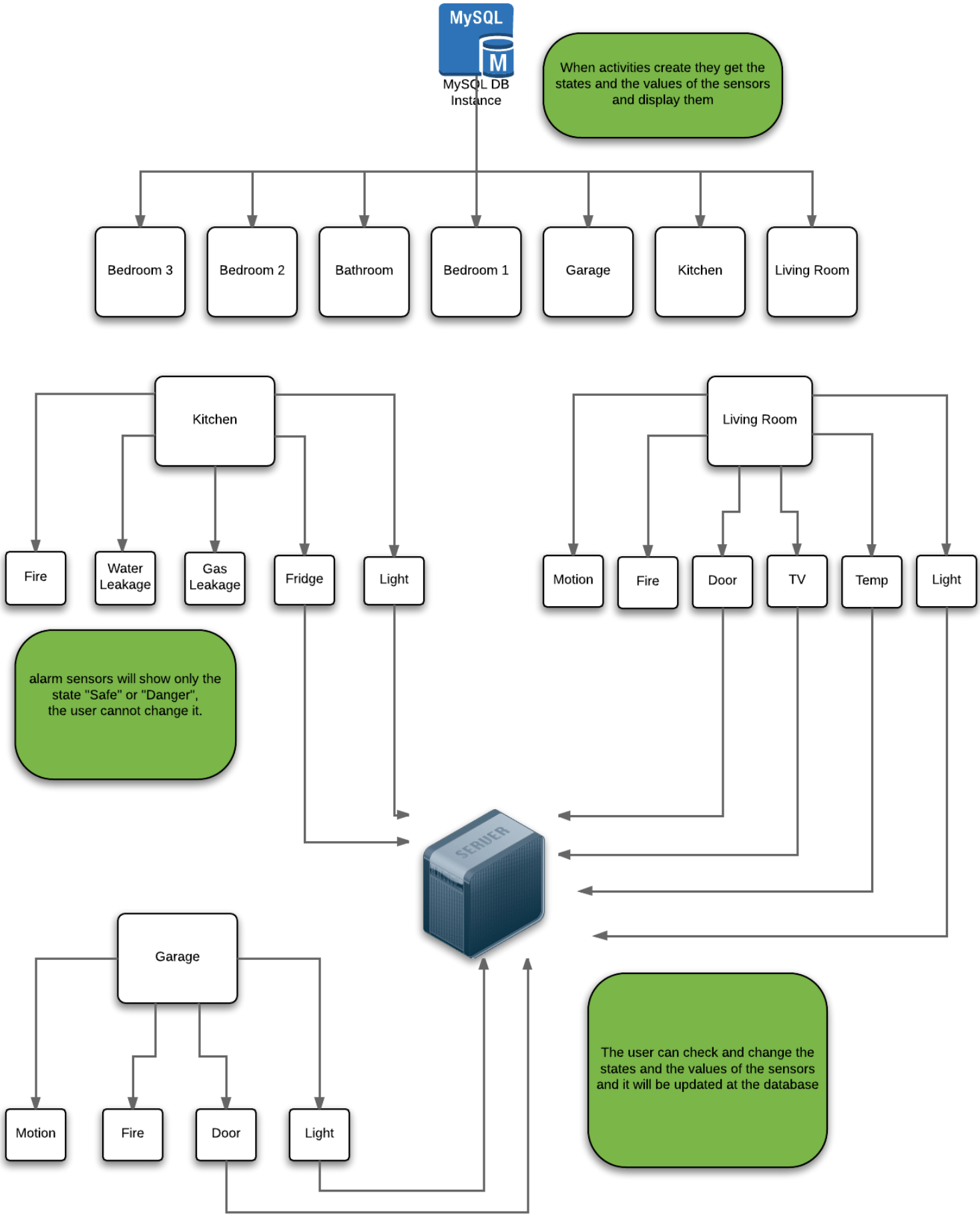
- In this application we used Firebase Cloud Messaging (FCM) to implement notifications.
- By using FCM, we can notify a client app that new email or other data is available to sync.
- When the user turns on or off the light in the house, the other users in the house will receive notification to notify them there is change happens in the house.
- The data in the notification will be the id of the sensor and its state and value.
- When the application receives the notification, it takes the data in this notification to update the fragment by changing the state of switches and the value of seek bar.
- There is also synchronization between the website and the application, when the user changes the switch of any sensor in the website, the mobile application will receive notification for this change, and the value will be updated automatically in the fragment.

5.7 FLOWCHART

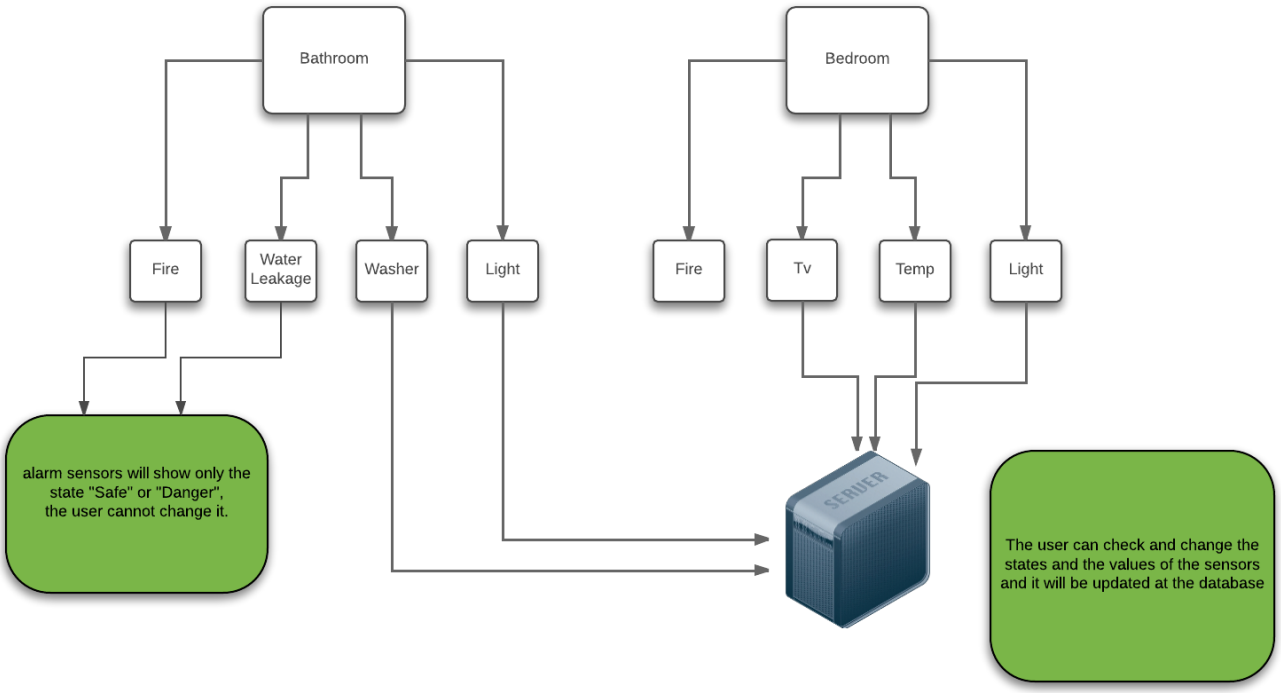
5.7.1 Part 1



5.7.2 Part 2


MySQL
MySQL DB Instance
When activities create they get the states and the values of the sensors and display them
Bedroom 3
Bedroom 2
Bathroom
Bedroom 1
Garage
Kitchen
Living Room
Kitchen
Fire
Water Leakage
Gas Leakage
Fridge
Light
Living Room
Motion
Fire
Door
TV
Temp
Light
alarm sensors will show only the state "Safe" or "Danger", the user cannot change it.
SERVER
Garage
Motion
Fire
Door
Light
The user can check and change the states and the values of the sensors and it will be updated at the database

5.7.3 Part 3


Bathroom
Bedroom
Fire
Water Leakage
Washer
Light
Fire
Tv
Temp
Light
alarm sensors will show only the state "Safe" or "Danger", the user cannot change it.
SERVER
The user can check and change the states and the values of the sensors and it will be updated at the database

Chapter Six

6 Raspberry Pi and Hardware

INTRODUCTION

This chapter is dedicated to the microcontroller (Arduino), the computer (Raspberry Pi) and the connection between them.

This chapter will discuss in details protocol used between Arduino and Raspberry Pi, Arduino's software and hardware connection including all the sensors, their theory of operation and finally, Raspberry Pi with its hardware connection to the wireless module, the code driving this module and the external library used.

The microcontroller used in our project is Arduino, our vision for the project was to use Arduino as a prototype then migrate all the Arduino code to AVR atmega328p which is the same used on Arduino board with different command set which is more focused towards low level languages. It wasn't long until we realized that creating a communication protocol between Arduino and Raspberry Pi able to handle errors and recover from it if needed was not as easy as we anticipated. The Arduino code kept changing (alongside Raspberry Pi code) on a daily bases to update the changes in the protocol. No to mention that the library used for AVR is nothing like the one we used for Arduino so this would have talking more time that we simply don't have.

The computer used in our project was Raspberry Pi zero which is the newest addition to the Raspberry Pi family. The main reason we used Raspberry pi zero was the low funding and flow of money at the beginning of the project, we had to cut our expenses because the university fund too a lot of time to reach us.

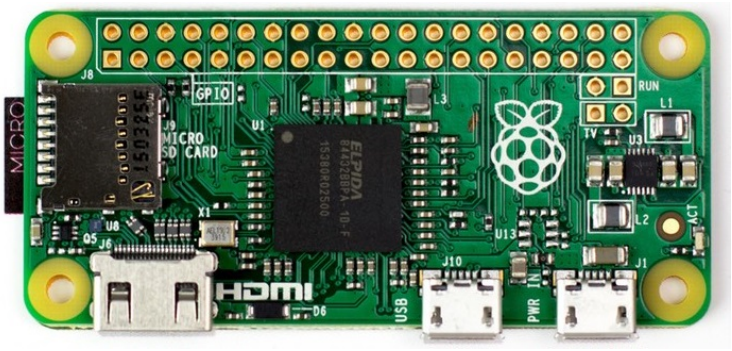


Figure 6-1: Raspberry Pi Zero

Raspberry pi zero is no different to any other raspberry pi in terms of the operating system and the GPIO pins. All of them have 40 identical GPIO pins and runs the official Raspbian OS which is a Linux distribution made by Raspberry Pi company for their Raspberry Pi.

The main difference between all Raspberry Pi boards is the processor and the RAM.

Raspberry Pi Zero specs:

- A Broadcom BCM2835 application processor
- 1GHz ARM11 core (40% faster than Raspberry Pi 1)
- 512MB of LPDDR2 SDRAM
- A micro-SD card slot
- A mini-HDMI socket for 1080p60 video output
- Micro-USB sockets for data and power
- An unpopulated 40-pin GPIO header
- Identical pinout to Model A+/B+/2B
- An unpopulated composite video header
- Our smallest ever form factor, at 65mm x 30mm x 5mm

Another reason we choose raspberry pi zero was its small size, it might be slow compared to other versions, but it can still provide a decent processing power for CL based program, luckily the majority of our programs are CL program except for one GUI program.

The main reason we used NRF24L01+ was the low funding as we mentioned before Also, NRF24L01+ is a very popular transceiver module with a lot of support. Or at least that what we thought at the very beginning, later on, we discovered that all the support was for Python language because it was a very easy programing language suitable for amateurs and beginners or someone who wants to make a simple program. We used C++, the extent of its support was to turn on and off an LED and the flow of data was in one direction only (from Rpi to Arduino).

Initially we wanted to use C but we upgraded to C++ since C++ is just C with extra stuff. The variations that draws the line between the two languages are no concern to us in this specific project.

Writing code for Raspberry Pi in C++ was an uncharted territory for all of us, as we said there was almost no support from the community to C++ and we had to read the library headers and .cpp file just to get a sense of how they all fit together before moving on to the codding part. The reason we crossed of Python in the first place was that of the time it takes to execute one command compared to other languages like C++ which took almost 1/9 of the time

taken by Python to execute a simple blink program sent to Arduino. We were using Raspberry Pi zeros so we had to be conservative when it comes to processing.

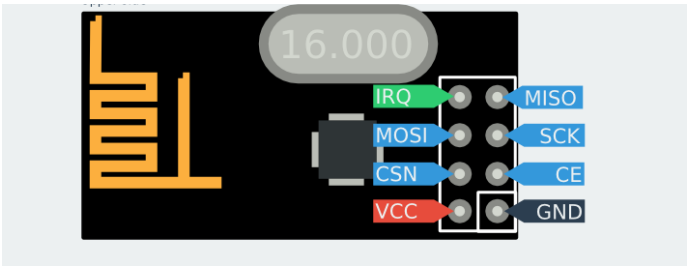
6.1 WIRELESS MODULE (NRF24L01+)

The 2.4G wireless transceivers module used to achieve the communication between Raspberry Pi and Arduino NRF24L01+.

Transceivers NRF24L01+ is one of the most used wireless communication modules due to its low price, the module also provides a very decent range up to 100m.

The transceivers module works in the 2.4GH band which explains the large range for this small power conservative device.

Although the 2.4GH band has a higher range, it also has higher interference chance since we live in a 2.4GH age. This interference (although it's not frequent) can cause a ripple effect that leads up to devastating results bringing down the entire system in rare cases.



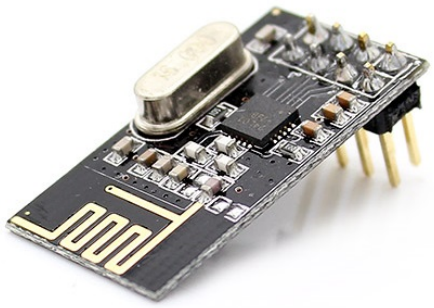


Figure 6-2: NRF24L01+

6.1.1 Advantages:

Transceiver modules can connect via SPI and have a lot of features like low power modes, multiple channels, channel hopping, frequency calibration and CRC.

Very low power concussion comparing to other wireless module. This power consumption is in normal operation mode. Activate low power mode will reduce the power consumption even more.

It has a max of 2Mbps which to very high compared to the price and other modules with the same functionality. This allows for file transfer over the module using low-level byte transfer.

Has a wide variety of libraries that we choose from, we choose tmrh20 library which has a lot of helpful featured that allowed us to find and solve some problems in addition to a well-organized documentation.

6.1.2 Disadvantages

One of the disadvantages of this module is sensitivity to noise and power fluctuation. At most cases when the module faces distortion in data due to the noise or power fluctuation, it will cause Arduino or Raspberry Pi to misinterpret the data received. But on rare occasions, the module will enter a state off hardware error and will not exist this state unless its process was killed manually. We tried to overcome the power fluctuation by adding a small capacitor between GND and VCC of the transceiver.

No communication protocol so we have to build it from the ground up which was a very time-consuming task, not no mention that it was very faulty in the early stages with too many problems which can pose as normal behavior when some of these problems cancel on another.

Although some consider SPI to be a feature, in our case it was not. It consumed 6 pins on Arduino which was too many pins for a small microcontroller with a limited number of pins. To fix this problem we considered moving to a different microcontroller (Atmel) like atmega16 but it was not possible due to the different library used, the tight time schedule and having too many moving parts in the project that would get heavily affected if we migrated.

The transceiver module has only one writing channel, although it has many listening channels. The problem with one writing channel it that more than one program might need to send a command to Raspberry Pi at the same time. If this happens at any instance of the execution then the SPI but will halt causing all other programs using SPI bus to halt as well. A guard code was added to all programs to prevent this from happening.

6.1.3 Why NRF24L01+

Although it might appear that we used the nrf24l01+ module because of the lack of funding early stages of the project, however, this is not the case, we used nrf24l01 because we wanted to go wireless. The I2C protocol was a very strong candidate that both Raspberry Pi and Arduino supported. In face all the microcontroller available today support I2C protocol.

But we took a vote and decided that using I2C would mean too many wires in the maquette and we will have to ground all the electric devices to a common ground for I2C to work.

6.2 THE MICROCONTROLLER (ARDUINO)

The microcontroller are to act as slaves to the main computer (Raspberry Pi), not to talk unless talked to.

We have only two type of programs run by Arduino, the Bed Room program and the Living Room (Hall) program. The room program contains the sensors that indicate there is a danger to the human resident or provide a control to certain devices in the room. On the other hand, the living room program is designed to provide main functions to the house and reading on the environment surrounding the house.

Both programs are continuously listening for incoming data while maintaining control of the sensors connected to them, whether it's controlling a certain device or activating the corresponding alarm to a certain safety sensor.

When a data is available in the NRF24L02+ buffer, Arduino calls a receiver function to read the data from buffer then calls another function reply() to match the data to a certain predefined command that is agreed upon between Arduino and Raspberry Pi. If the command was not defined or the data was courted due to interference the Arduino will not understand it and will send back to Raspberry Pi "No such command".

6.2.1 The Living Room Program

There can only be one Arduino running this program in the sense that any house has one living room.

Raspberry Pi can do two main functions or calls to Arduino, Read and Write. Read as in send me the reading of the sensors you have and Write as in store (write in your memory) the desired state/reference of one of the functions you provide.

6.2.1.1 Read (Raspberry Pi request to read from Arduino):

There is only one read “readDataSLV2” command that Raspberry Pi send to Arduino, this command tells Arduino to send all sensors reading connected to it.

When raspberry pi send a “readDataSLV2” request, Arduino understand this as a request to read sensors to Arduino dose the following:

- Set NRF24L01+ state to write.
- Read the State/value of all connected sensors.
- Send all the reading/value to Raspberry Pi.
- Print those reading/values to serial (if connected)
- Check if send was successful or not.
- Print to serial (if connected) the send return, whether if failed to succeeded.
- Set NRF24L01+ state back to listen and resume normal functions (main loop).

6.2.1.2 Write (Raspberry Pi writes to Arduino):

There are 6 different commands in write section:

6.2.1.2.1 “lightReference”

“lightReference” command is used to set the desired light reference ± margin (read from LDR), this reference determines at which point the light will turn on or off.

When lightReference command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Send ACK to Raspberry Pi to let it know Arduino is expecting the reference.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.
- If sending succeeded then read the reference and update it in the corresponding variable.
- Set NRF24L01+ state to listen and go back to the main loop.

After Arduino sets the light reference (voltage) in its memory then it applies the control on this newly set reference at the next iteration.

6.2.1.2.2 “tempReference”

“tempReference” command is used to set/update the desired temperature reference ± margin (read from a temperature sensor), if the temperature drops below the desired reference,

then a heater will turn on and if it rises above the desired temperature then a fan will start working to reduce the temperature.

"tempReference" command is not much different than "lightReference" command, in fact, they both use the same technique to update their references.

When "tempReference" command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Send ACK to Raspberry Pi to let it know Arduino is expecting the reference.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.
- If sending succeeded then read the reference and update it in the corresponding variable.
- Set NRF24L01+ state to listen and go back to the main loop.

6.2.1.2.3 “tvOn” or “tvOff”

Both of those command are made to control the TV state (turn TV on or off).

When “tvOn” or “tvOff” command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Call TV control function with a parameter of 0 (off) or 1 (on).
- The function then outputs a LOW or HIGH on the corresponding pin of the TV to enable or disable electricity flow to the TV.
- Send ACK to Raspberry Pi to let it know Arduino executed the sent command. There is no chance of failed since it’s a simple digital write to a local register.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.
- If sending succeeded set NRF24L01+ state to listen and go back to main the loop.

6.2.1.2.4 “closeGarageDoor”

“closeGarageDoor” command is used to close the garage door if open.

When “closeGarageDoor” command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Call garage door control function with a parameter of 1 (close).
- The function then checks whether the garage door is open or closed, there is a contact pin in the garage that lets you know when the door is closed.
- If the door is already closed then Arduino immediately send ACK to Arduino, sets NRF24L01+ to listen and resume normal function (main loop).
- If the garage door is open then the Arduino starts to rotate the stepper motor anti-clockwise for a 20 thousand steps (very large) until contact is closed.
- If motor moved 20 thousand steps and contact is still open then exit the function with failed flag. Then send NACK.
- If contact is closed, Arduino exits the function with success flag. Then send ACK.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK/NACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.

6.2.1.2.5 “openGarageDoor”

“openGarageDoor” command is used to open the garage door if closed.

When “openGarageDoor” command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Call garage door control function with a parameter of 0 (open).
- The function then checks whether the garage door is open or closed, there is a contact pin in the garage that lets you know when the door is closed.
- If the door is already opened then Arduino immediately send ACK to Arduino, sets NRF24L01+ to listen and resume normal function (main loop).
- If the garage door is closed then the Arduino starts to rotate the stepper motor clockwise for a 4 thousand steps (measured and predefined from testing) that will successfully open the door.
- If motor moved the 4 thousand steps and contact is still closed then exit the function with failed flag. Then send NACK.
- If contact is opened after finishing, Arduino exits the function with success flag. Then send ACK.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK/NACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.

6.2.1.2.6 Any other command

This was set to handle undefined or corrupted command to not let it fill the RX buffer of NRF24L01+ or cause Arduino to misbehave.

When an undefined command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Print an error message with the command received as a string and floating point number.
- Send “unknown command” to Raspberry Pi and set NRF24L01+ to listen.
- Go back to resuming normal functions (main loop).

6.2.1.3 Sensors

In this sub-section, we will be covering the sensors used with the living room Arduino program and their theory of operation.

6.2.1.3.1 DHT11

DHT11 is a low cost calibrated low power with a single bus digital temperature and humidity sensor. That we documented.

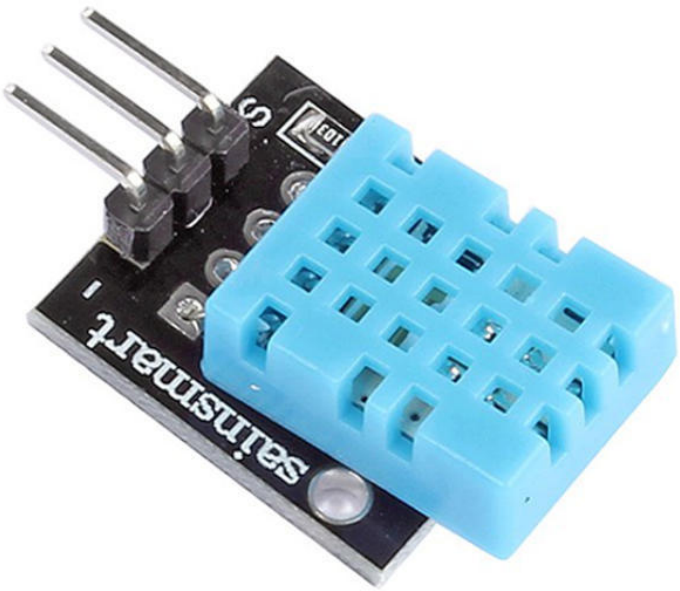


Figure 6-3: DHT11

As we mentioned that the DHT11 sensor is digital sensor the uses a single but to read data from. The only means of communication using a single but with another digital device it to use a master-slave technique (similar to I2C without a CLK). In this case, the microcontroller (Arduino) is the master and DHT11 sensor is the slave.

Initially, the sensor is in ideal state until we pull the signal bus to low (we write) when the sensor reads low on the bus, it knows that this is the master requesting to read, so it starts to send the temperature and humidity readings to the master using its own clock. This creates a timing problem due to the different clock used by the microcontroller.

Timing problem could be easily avoided by examining the datasheet and learning to interoperate the signal and the clock speed it uses. Or by simply using a predefined library. We initially wrote our own code based on the datasheet description of the signal but then moved to the library to avoid unnecessary code.

6.2.1.3.2 LDR (Light Dependent Resistors)

Light Dependent Resistors sensor consist of a photo resistor that changes its resistance with the amount of incident visible light and a film resistor to create a voltage divider that changes the output voltage when the LDR changes its value with incident light.

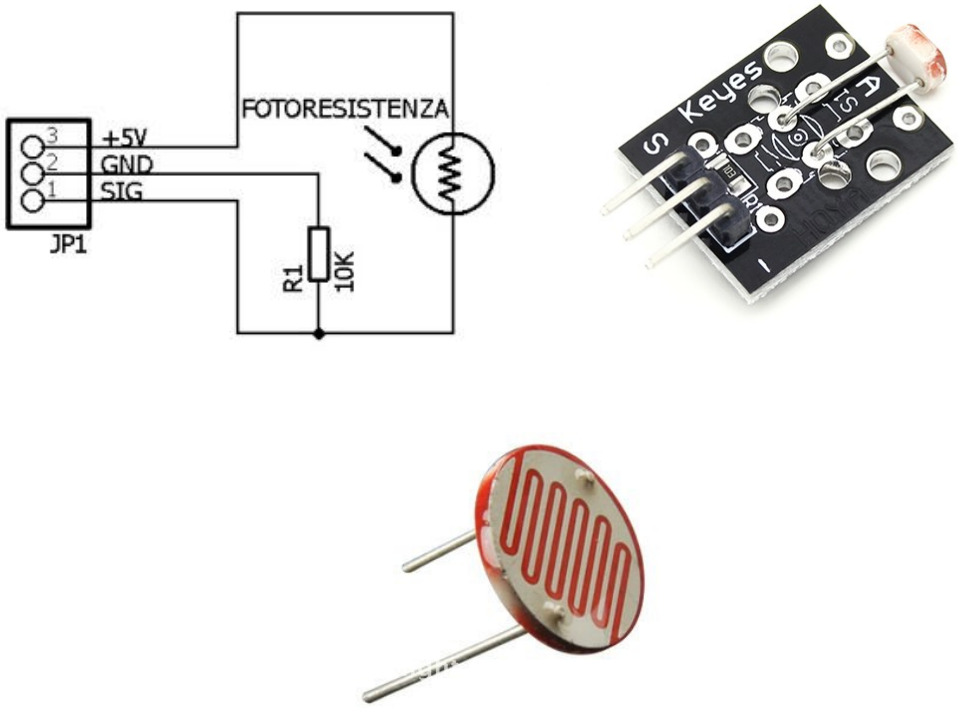


Figure 6-4: LDR (KY - 018)

The principle of operation is quite simple. It has nothing more to it than a simple voltage divider just like the one showed above **Error! Reference source not found.**.

This sensor is common for all rooms and the hall as it indicate the amount of light outside the house.

6.2.1.3.3 Temperature sensor KY-013

The KY-013 sensor consist of a thermistor (resistance increases with the ambient temperature changes) with a film resistor to create a voltage divider just like LDR (Light Dependent Resistors) sensor.

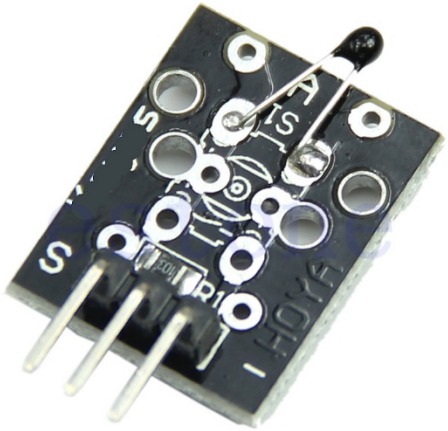


Figure 6-5: KY-013

The resistance of the thermistor changes with the temperature of the room. The range of the sensor is -55°C / +125°C with an error margin of ±0.5°C which is very good yet unnecessary.

Unfortunately we couldn't find its reference on the internet, however, we did find a function that convert the readings from volt to any temperature unite desired.

```
double Thermistor(int RawADC) {
  double Temp;
  Temp = log(10000.0*((1024.0/RawADC-1)));
  Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );
  Temp = Temp - 273.15;            // Convert Kelvin to Celcius
   //Temp = (Temp * 9.0)/ 5.0 + 32.0; // Convert Celcius to Fahrenheit
   return Temp;
}
```

By knowing the max and min volt in addition to the max and min temp, the Thermistor function was possible to create assuming linear behavior of the sensor. Of course the sensor probably doesn't have a linear behavior but it's safe to assume that for the small range we work in (from 16°C to 35°C) which is the typical min and max room temperature that might happen. Rooms with higher or lower temperature than this range are highly unlikely.

6.2.1.3.4 Stepper motor with driver

To open and close the garage door we used a 5 wire unipolar high torque stepper motor 28BYJ-48 with 64:1 gear ration.

We used the driver ULN2003APG to provide an external power source if needed and provide protection to the microcontroller. We mounted the driver on an external board for easier connection and visualization of the motor activated coils.

Figure 6-6: Stepper Motor with Driver

We used a Full Step mode with the stepper to give maximum torque possible. The full step mode gave us 11.25 degree step angle with 32 step for a complete revolution.

The code we used to drive this motor was shown in

“openGarageDoor” and “closeGarageDoor”.

The initial library we used was AccelStepper but then we changed to Arduino standard library “Stepper” because we didn’t need the functions AccelStepper provided since Arduino’s standard library could achieve the same results with an extra few lines of code.

To understand more on stepper motor you could search for “28BYJ-48 Stepper Motor and ULN2003 Driver Intro” on YouTube. The video is provided by Microsoft and has very comprehensive details on stepper motors. We would give you a link to a presentation made by our university but we didn’t ask for permission first plus, it’s in Arabic. The video is just as good.

6.2.1.4 Control Functions

Control functions are all the functions that execute the user’s commands and maintain the desired values that were set/updated by the user..

6.2.1.4.1 lightControl()

This function has the sole purpose of turning on and off the light of the room based on the light reference set by the user.

6.2.1.4.2 tempControl()

This function compares the current temperature of the room with the desired temperature set by the user and then decide to either turn on the fan or the heater.

If the temperature is within the desired range then the function will turn off both the fan and the heater.

6.2.1.4.3 PIRControl()

This function has a simple task, to turn on an alarm (buzzer or turn on an LED) when the PIR sensor detects a movement of a living being to alert the user.

6.2.1.4.4 tvControl()

This function is used to either turn the TV on or off when the user send “tvOn” or “tvOff” commands.

6.2.1.4.5 garageDoorControl()

This function is like tvControl() function, it’s used to open or close the garage door when user send “openGarageDoor” or “closeGarageDoor” commands.

6.2.1.5 Flow Chart of Bed Room Microcontroller

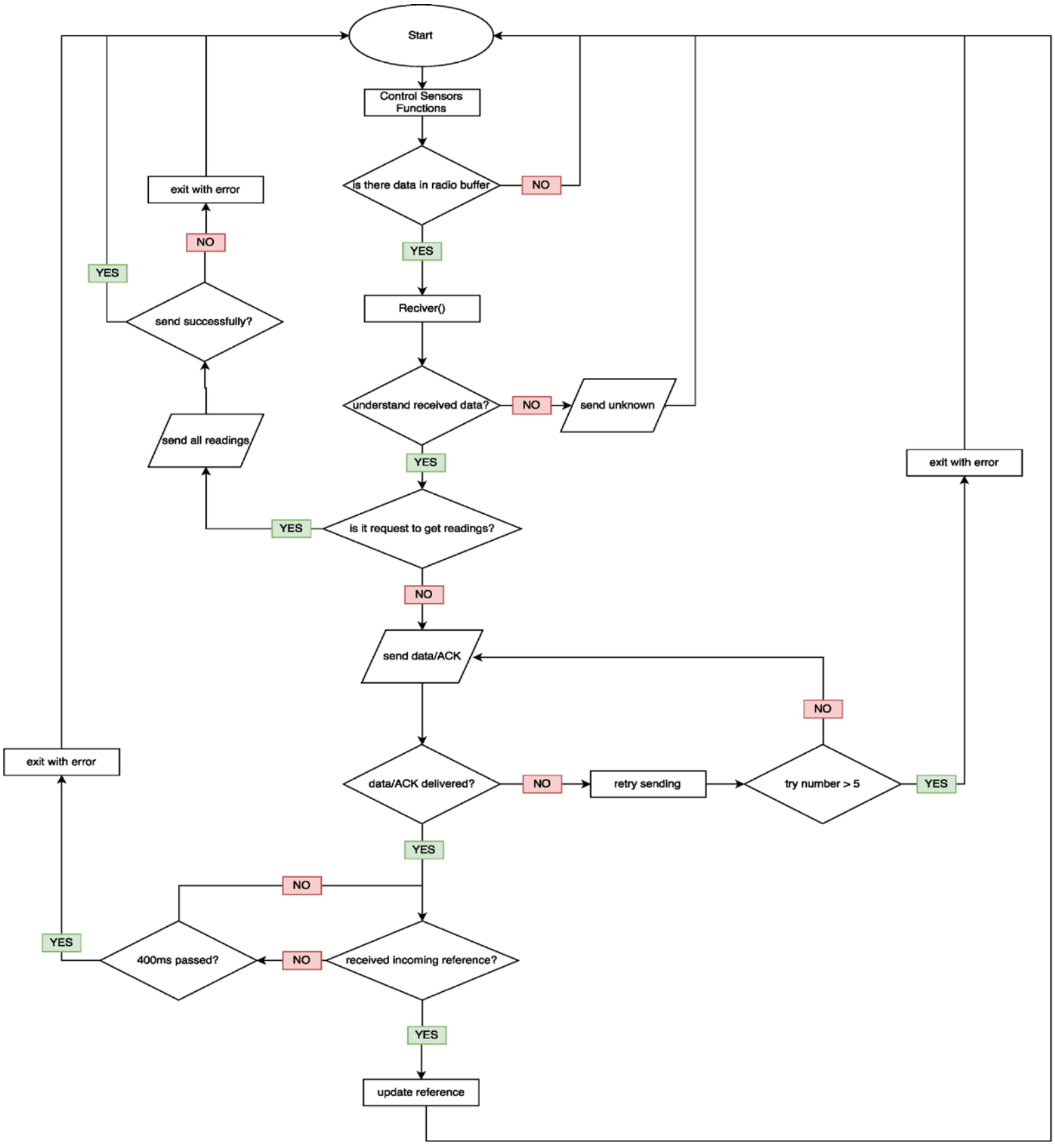


Figure 6-7 - Flow Chart of Bed Room Microcontroller

6.2.2 Bed Room Program

All the rooms in the house run this program including kitchen, bathroom etc...

Raspberry Pi can do two main functions or calls to Arduino which are Read and Write just like living room program with the difference in the controlled functions. Read as in send me the reading of the sensors you have and Write as in store (write in your memory) the desired state/reference of one of the functions you provide.

Bed room program has more safety sensors and less functionality, typically rooms are smaller than the living room and its Arduino is concerned only with this specific room unlike the living room program.

6.2.2.1 Read (Raspberry Pi request to read from Arduino):

There is only one read "readDataSLV1" command that Raspberry Pi send to Arduino, this command tells Arduino to send all sensors reading connected to it.

When raspberry pi send a "readDataSLV1" request, Arduino understand this as a request to read sensors to Arduino dose the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Read the State/value of all connected sensors.
- Send all the reading/value to Raspberry Pi.
- Print those reading/values to serial.
- Check if send was successful or not.
- Print to serial (if connected) the send return whether if failed to succeeded.
- Set NRF24L01+ state back to listen and resume normal functions (main loop).

6.2.2.2 Write (Raspberry Pi writes to Arduino):

There are 2 different commands in write section both are exactly like their counterparts in living room program:

6.2.2.2.1 "lightReference"

"lightReference" command is used to set the desired light reference ± margin (read from LDR), this reference determines at which point the light will turn on or off.

When lightReference command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Send ACK to Raspberry Pi to let it know Arduino is expecting the reference.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.
- If sending succeeded then read the reference and update it in the corresponding variable.
- Set NRF24L01+ state to listen and go back to the main loop.

After Arduino sets the light reference (voltage) in its memory then it applies the control on this newly set reference at the next iteration.

6.2.2.2.2 "tempReference"

"tempReference" command is used to set/update the desired temperature reference ± margin (read from a temperature sensor), if the temperature drops below the desired reference, then a heater will turn on and if it rises above the desired temperature then a fan will start working to reduce the temperature.

When "tempReference" command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Send ACK to Raspberry Pi to let it know Arduino is expecting the reference.
- Empty TX buffer of NRF24L01+ in case an error occurred.
- If sending ACK failed then Set NRF24L01+ state to listen then exit function with error print and return to the main loop.
- If sending succeeded then read the reference and update it in the corresponding variable.
- Set NRF24L01+ state to listen and go back to the main loop.

6.2.2.2.3 Any other command

This was set to handle undefined or corrupted command to not let it fill the RX buffer of NRF24L01+ or cause Arduino to misbehave.

When an undefined command is received, Arduino does the following (assuming serial is connected):

- Set NRF24L01+ state to write.
- Print an error message with the command received as a string and floating point number.
- Send “unknown command” to Raspberry Pi and set NRF24L01+ to listen.
- Go back to resuming normal functions (main loop).

6.2.2.3 Sensors

In this sub-section, we will be covering the sensors used with the bed room Arduino program and their theory of operation.

There are a few sensor that are common between the Bed Room Program and The Living Room Program like LDR (Light Dependent Resistors) and Temperature sensor KY-013.

The temperature sensor we’re using in KY-028 although it looks different, it still has the same thermistor as KY-013 that we already discussed.

6.2.2.3.1 Fire Sensor (KY-026)

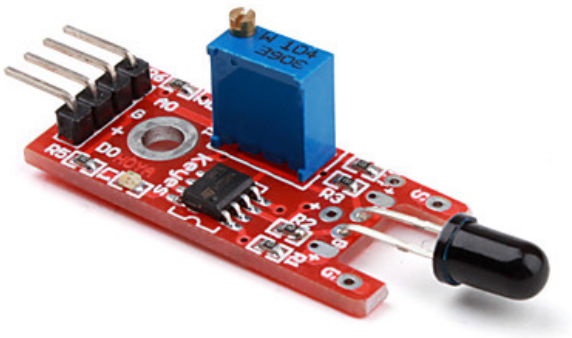


Figure 6-8: KY-026

This sensor is a part of the KY family just like most of the sensors we used and documented. The sensor detects fire or flame, it can detect light wavelengths between 760 nm to 1100 nm infrared. The wavelength band between 760 nm to 1100 nm is usually where the light emitted by flames or fire lies, however, not all flam light lies in this band. This wavelength range is mainly for the red flame, the blue flames have different wavelength and usually undetectable by this sensor.

In any case, the fact that this sensor might not detect blue flame does not create a problem. In fact, this is good. Blue flames are what referred to as “pure flame” and they are the result of a complete combustion meaning that only gas and oxygen were used in creating this flame, they are created by the combustion of natural gas which is safe. This combustion will be usually found in kitchens when cooking so it will not cause the sensor to detect flames.

In the case of yellow flame, we get the sign of incomplete combustion meaning that there is something other than gas is being burned. In this case, the sensor will detect those flames and will pull up its digital output to high (active high) indicating the detection of a yellow flame.

The KY-026 flame sensor has also an Analog output. This is an extra feature in the sensor that allows us to detect exactly how much wavelength was detected (lower voltage is for higher wavelength detection) in the range of the sensor of course.

6.2.2.3.2 Gas Sensor MQ2

MQ-2 is a heated low cost gas sensor that has high sensitivity to LPG, Propane, and Hydrogen, also could be used to Methane and other combustible steam with a small size, high power combustion (draws a lot of current) and easy to read datasheet.

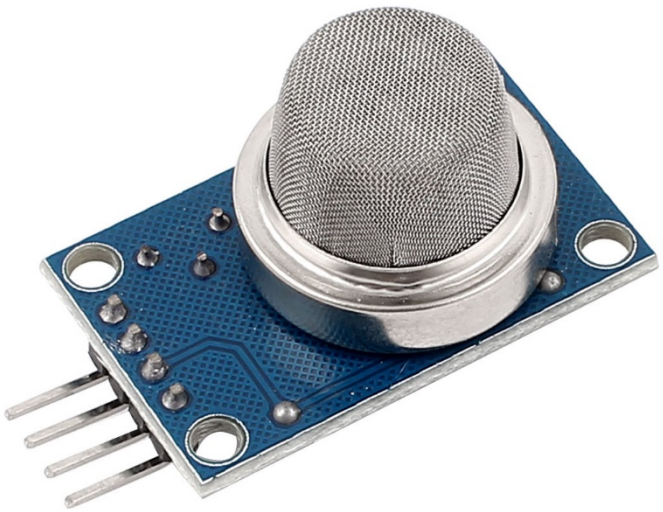


Figure 6-9: MQ2 Sensor

As we mentioned before this sensor has a wide variety of dangerous gases and combination that it can detect. It can detect smoke as well in the sense that it might have some Propane that was not combusted.

MQ2 is an active low sensor meaning it pulls its digital pin to Gnd when dangerous gas level is detected. The sensor module has a trimmer potentiometer on the board to allow the user to set the exact point/concentration at which the digital pin of the sensor will high detect gas.

The sensor also has an analog output that can allow you to detect the ration of gases with a simple calculations that get the reading of the sensor from an open area and compare it with the normal amount of each gas in the atmosphere (setting its standard readings). By placing the sensor in whatever room you want to measure the gas concentration, you can get the ration of most of the gas in this room.

6.2.2.3.3 Water leakage (rain) sensor

The sensor was initially made to detect rain but we repurposed it to detect water leakage.

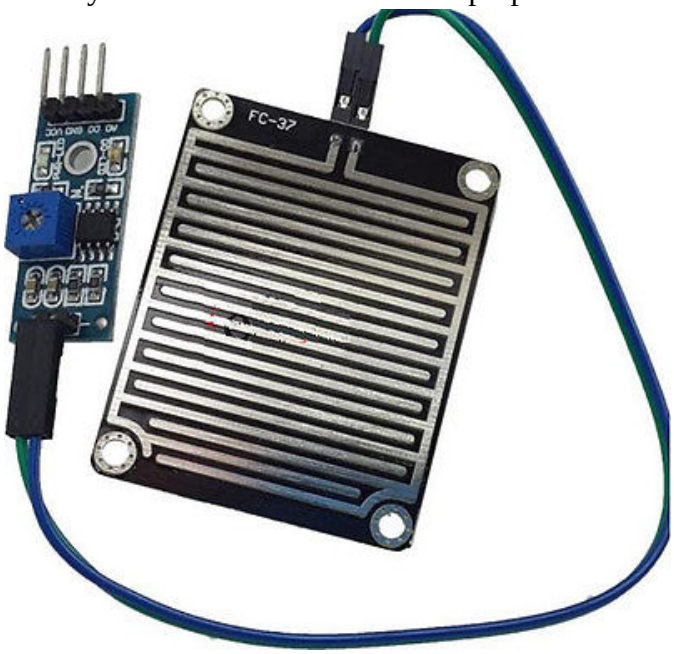


Figure 6-10: Rain Sensor

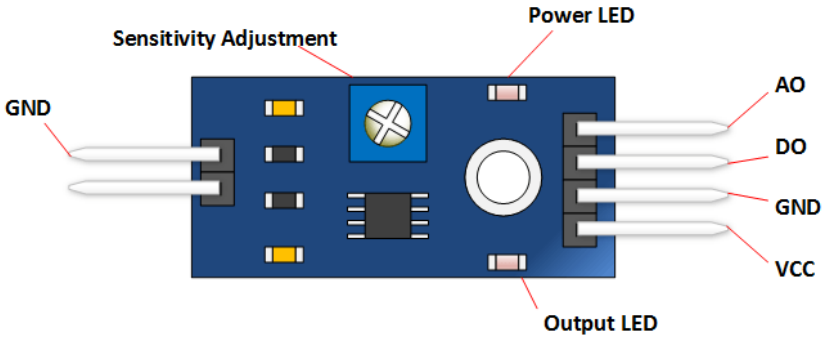


Figure 6-11: LM393 chip

When water is dropped or touches the sensor's board, the digital output will change to high (active high), we can change the amount of water that needs to be touching the board in order for it to give a high output by changing the resistance on the LM393 chip of the sensor using the built-in trimmer potentiometer.

The idea behind this sensor is very simple. The sensor detects water that completes the circuits on its double sided sensor boards' printed leads. The sensor board acts as a variable resistor that will change from 100k ohms when wet to 2M ohms when dry. The wetter the board the more current that will be conducted.

The sensor provides an analog output to know exactly how much water is on touching the board.

6.2.2.3.4 PIR sensor

The PIR sensor stand for Passive InfraRed, it's a passive sensor that detect infrared light emitted by living beings, it can't detect any object that doesn't emit infrared. In short, it can only detect living beings like a human or an animal.

Figure 6-12: PIR sensor

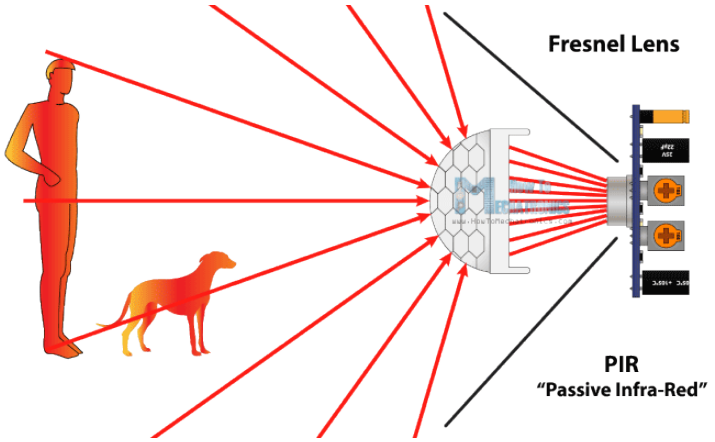


Figure 6-13: PIR sensor operation

The PIR sensor is made of a special two material that are sensitive to IR and a lens used to increase the angle of the sensor. When the sensor is idle, both slots detect the same amount of IR. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. This is how the sensor is able to detect movement.

When a movement is detected by the PIR sensor, the signal pin is pulled to high to indicate the movement of an object (living object).

The sensor stays in state of high until the object is no longer within range of the sensor (this can be set by a jumper on the sensor), this is a feather in the sensor that we enabled, also the amount the sensor stay active (high) is around 15 second after the object leaves it range. The amount of time to wait and the range are adjusted by two potentiometers at the back of the sensor. All these features can be found in the datasheet of the sensor.

6.2.2.4 Control Functions:

6.2.2.4.1 lightControl()

This function has the sole purpose of turning on and off the light of the room based on the light reference set by the user.

6.2.2.4.2 tempControl()

This function compares the current temperature of the room with the desired temperature set by the user and then decide to either turn on the fan or the heater.

If the temperature is within the desired range then the function will turn off both the fan and the heater.

6.2.2.4.3 gasControl ()

This function has a simple task, to turn on an alarm (buzzer or turn on an LED) when the gas level becomes dangerous.

6.2.2.4.4 waterLeakageControl ()

This function is like gasControl() function, it's used to turn on an alarm (buzzer or turn on an LED) when the water leakage sensor detects water.

6.2.2.4.5 fireControl ()

This function is like waterLeakageControl() and gasControl() function, waterLeakageControl to turn on an alarm (buzzer or turn on an LED) when the sensor detects fire.

6.2.2.5 Flow Chart of Living Room Microcontroller

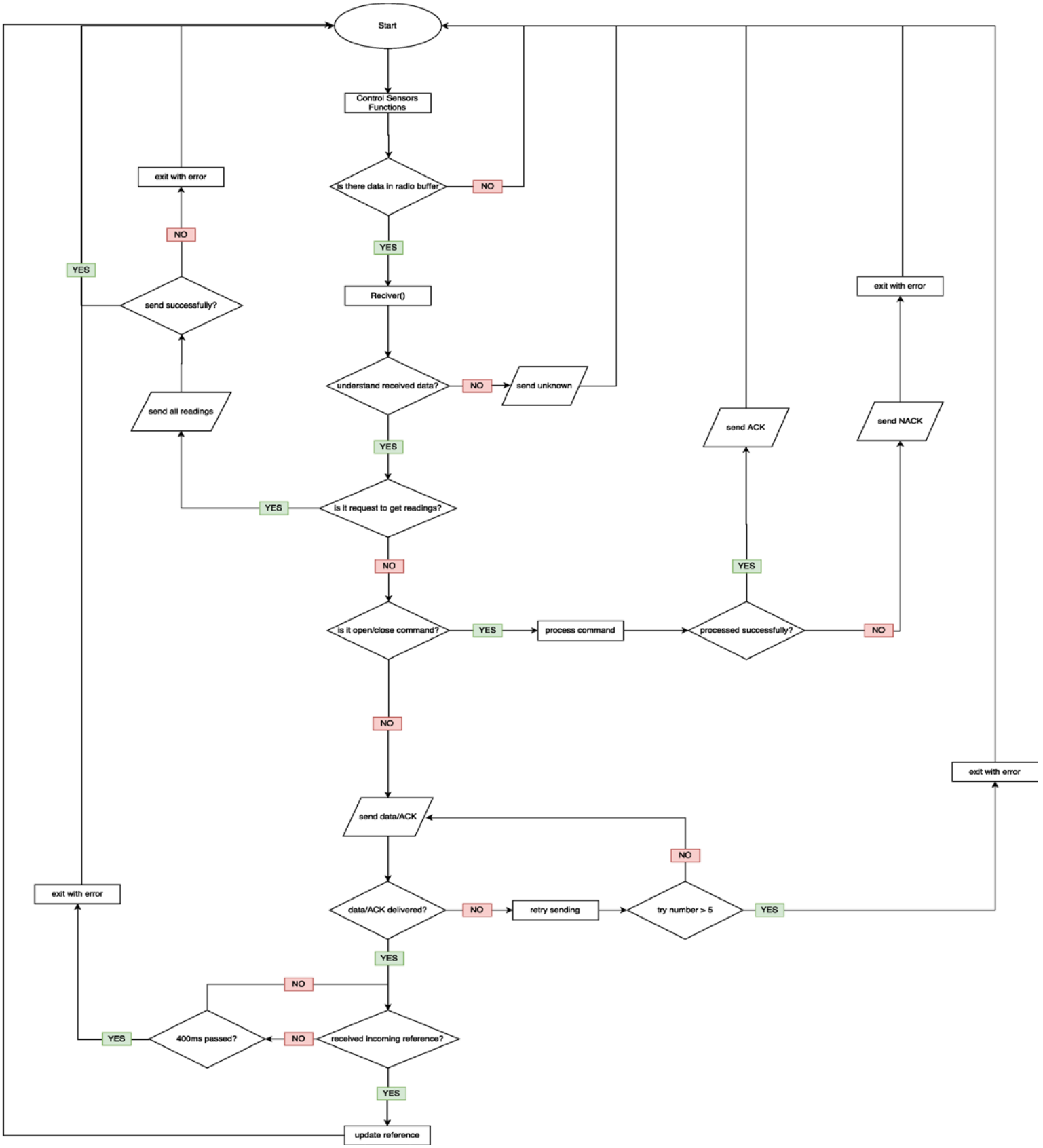


Figure 6-14: Flow Chart of Living Room Microcontroller

6.3 Raspberry Pi

In all communications between Raspberry pi and Arduino, Raspberry Pi acts as the master and Arduino as the slave.

Arduino is not permitted to TALK (send data) unless Raspberry Pi request it.

This method avoids any conflict between the multiple Arduinos in the house and Raspberry Pi in writing, as we mentioned before in NRF24L01+ Disadvantages that it has only one writing channel and can only send or receive at a time, half duplex.

We used Modular Programming Technique in building the raspberry pi software end for easier debugging and understanding.

In Raspberry Pi programming we have:

- Two reading programs, one of them is a fixed downgrade of the other designed to run only once at startup.
- Four writing programs, two of them are for updating temperature and light reference and the other two are for opening and closing TV and garage door.
- A mapping program to translate the sensors id (SID) coming from the database when the app or the website request something to the corresponding Arduino and the name of the command that execute what the user requested.
- A reading loop program that keeps on getting the sensors readings from Arduinos and updating the database. This program runs on startup and keeps on running.
- A guard program that is embedded in every program. The program guards the SPI bus that is connected to the NRF24L01+ module to prevent two programs from using it at the same time.
- A GUI program that acts as an interface like the web and the app but it can provide more detailed information that isn't available on other interfaces.

6.3.1 Guard program

6.3.1.1 Overview

The guard program guards the SPI bus as we mentioned before, the transceiver NRF24L01+ uses the SPI bus; we also mentioned this before in Advantages. The reason we need this program is because there is no method –in Raspberry Pi- to know when the SPI bus is used.

One may think that when the transceiver initializes its library then it occupies the SPI bus and this makes it easy since we can ask Raspberry Pi if someone uses the bus, however, this is not the case with NRF24L01+ modules. The module only used the bus when it’s sending or receiving to or from the Tx and or sending to the Rx buffers. This makes the bus used in different periods of the program lifetime, thus we need to prevent access to the bus altogether when any program want to use the SPI bus even for 1% of its lifetime.

If two programs were to use SPI at the same time then both programs will halt since the both entered a state of deadlock. The only way to resume normal function is to kill both PIDs manually.

This program does not exist on its own, the program is embedded at the beginning of main() in each program that used the NRF24L01+ module.

You can understand what the program does form the simplified Flow Chart. Or if you like you can always read the source code of the separated guard program which we highly recommend if you want to read any other program, the all have the queue module in their code directly or indirectly.

6.3.1.2 Guard Flow Chart

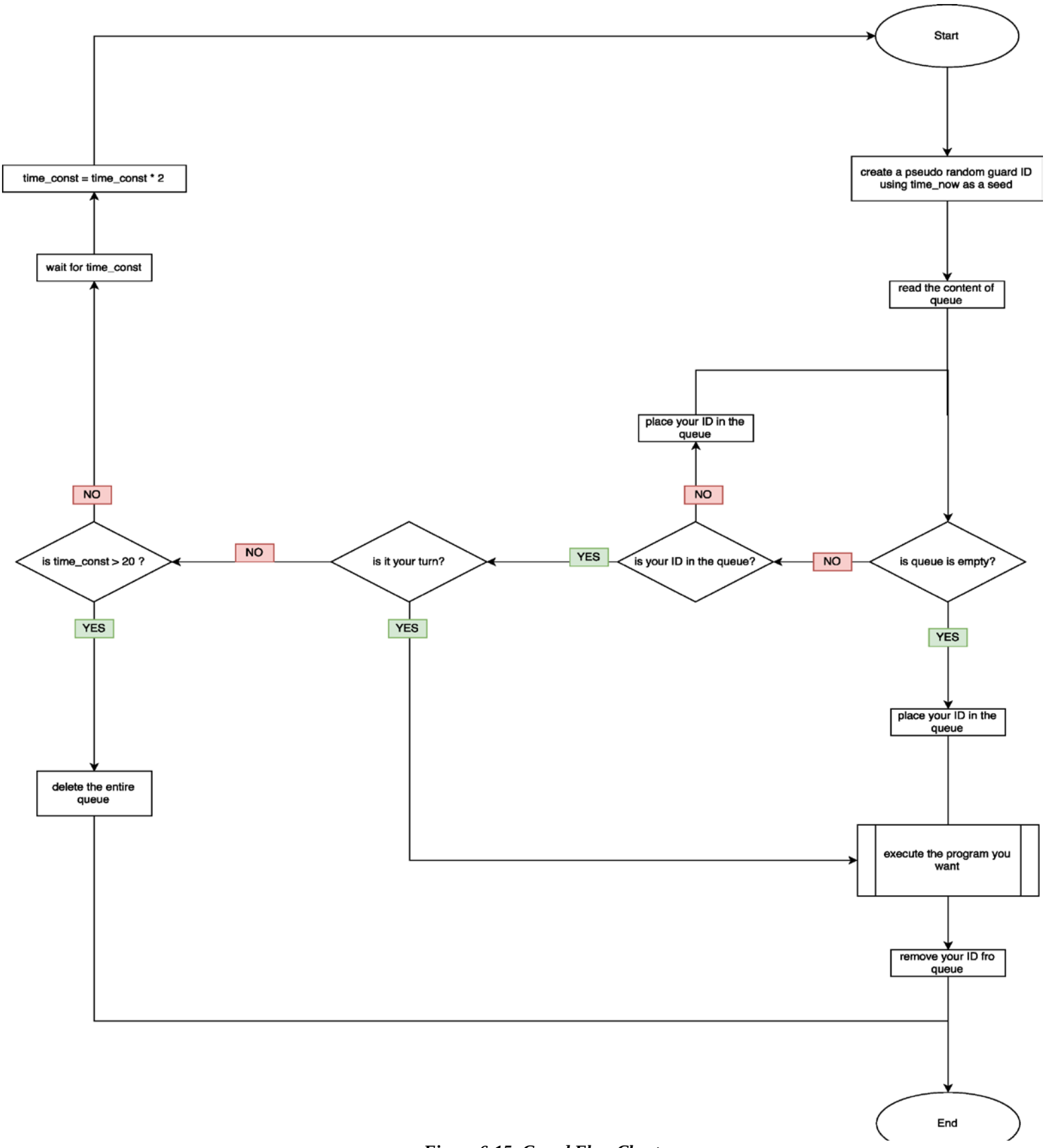


Figure 6-15: Guard Flow Chart

6.3.2 Reading Programs

The two programs that request to read from Arduino.

6.3.2.1 collector_get_readings.cpp

This program request sensors readings from all available Arduinos. The program sends a request to one Arduino after the other until collects all the readings from all the Arduinos. If one Arduino at the middle or at the end or even the first one failed to send then the program will return an error and will retry again from the beginning.

What the program does is the following:

- First, the guard module checks the queue and let the core program execute when it's our turn.
- The program invokes the setup function for SLV1 to open the correct channel for communication with this specific slave.
- Then the program invokes readSLV function to get sensors readings from slv. It has a max of five retries to get data before exiting and reporting error.
- Then it moves to read from the next slv if the first read was successful by invoking readSLV2. It also has a max of five retries to get data before exiting and reporting an error.
- Then it invokes the setGUIReadings function that updates the reading files for the GUI program.
- At the end guard module will remove the program ID from the queue to let other programs use SPI or NRF24L01+.

This program has multiple features that will not be shown in the flow chart since their main purpose was the debugging process. You may use the features together in the same command.

The command supposed to be called this way:

Note: <> means whatever inside is optional

, means that you can choose only one of the two.

```
~ $ sudo collector_get_readings <print> <v,%> <server,server_all>
```

Those features are:

6.3.2.1.1 ‘print’

```
~ $ sudo collector_get_readings print <v,%>
```

You can activate this features by passing “print” to the program when calling from command line. What print does is print all the values of the sensors you read.

You can chose to print then as voltage ‘v’ or percentage ‘%’. By default it prints the voltage.

6.3.2.1.2 ‘server’

```
~ $ sudo collector_get_readings <’server’ , ‘server_all’ >
```

Including this in the command will cause the program to check if any safety sensor indicated danger after reading then send them to the server.

‘server_all’ will force the program to send all the reading of all the servers.

6.3.2.2 collector_get_readings_startup.cpp

This program is a replica of collector_get_readings.cpp made by request from server so they can request us (Hardware Team) to make change to it without effecting the main program.

No changes were made to this program except for removing some function that were not used and send some specific sensors reading instead of the slandered send safety or send all.

6.3.2.2.1 collector_get_readings.cpp Flow Chart

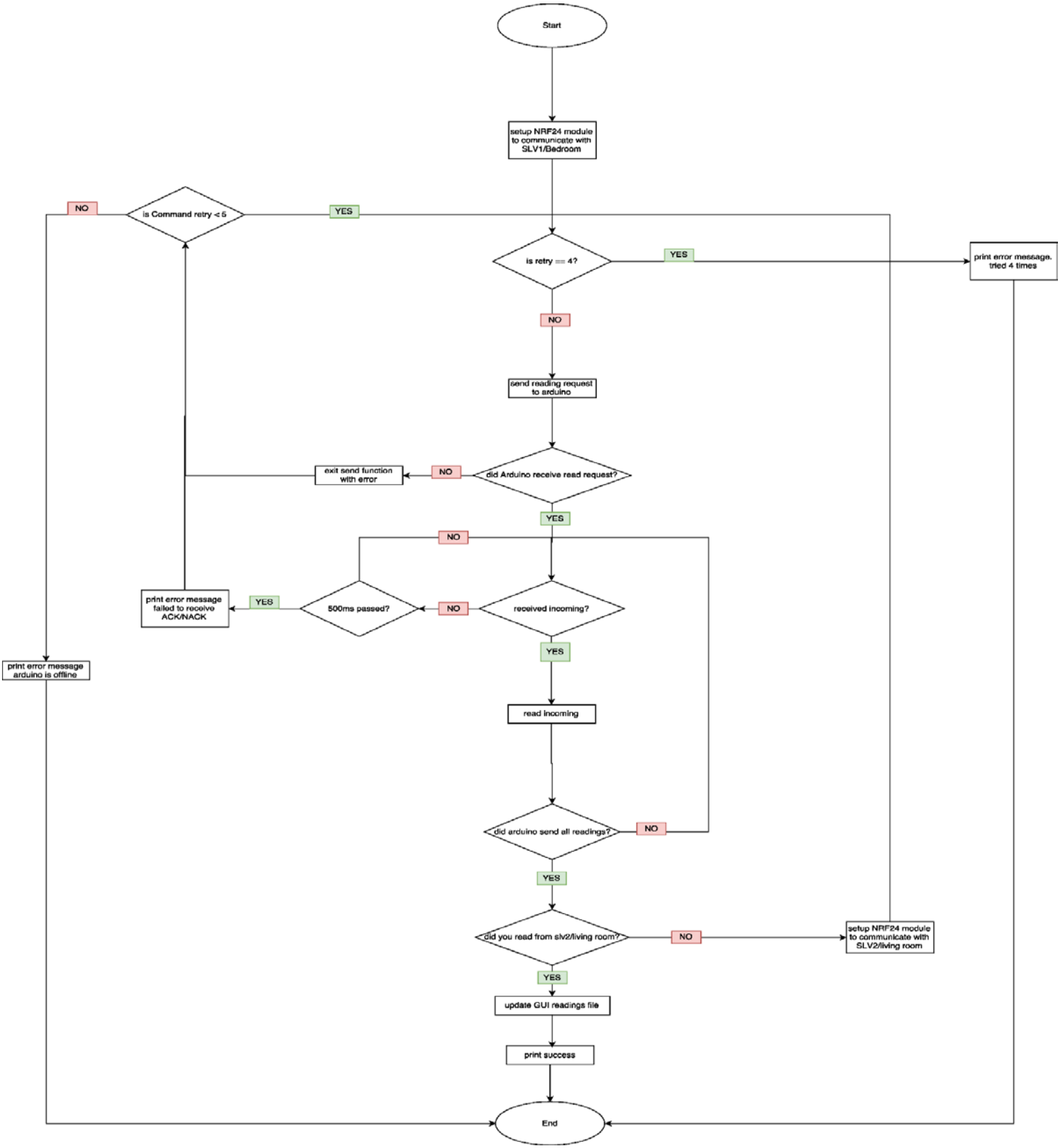


Figure 6-16: collector_get_readings.cpp Flow Chart

There programs are used to requests from the server to Arduino, the available requests are: update temperature or light reference and turn on or off TV and garage door.

6.3.2.3 collector_temp.cpp and collector_light.cpp

What the program does is the following:

- First, the guard module checks the queue and let the core program execute when it's our turn.
- The program checks which room the server sent the reference to and then invoke setup_slv1 or setup_slv2 function to open correct channels for communication.
- Then the program invokes loop function to send the correct command for either light or temperature reference update. It has a max of five retries to send data before exiting and reporting error.
- After sending the reference update command to Arduino, the program will receive an ACK if everything is OK, after receiving the ACK the program will send the reference.
- At the end guard module will remove the program ID from the queue to let other programs use SPI or NRF24L01+.

The two programs have a sending option that let them know which Arduino Raspberry Pi is to send to.

The command supposed to be called this way:

Note: [ ] means that you have to pass an argument

, means that you can choose only one of the two.

```
~ $ sudo collector_temp [‘room1’ , ’hall’]
~ $ sudo collector_light [‘room1’ , ’hall’]
```

‘room1’ means that you’re sending to bedroom1

‘hall’ means that you’re sending to living room.

6.3.2.3.1 collector_light.cpp/collector.cpp temp Flow Chart

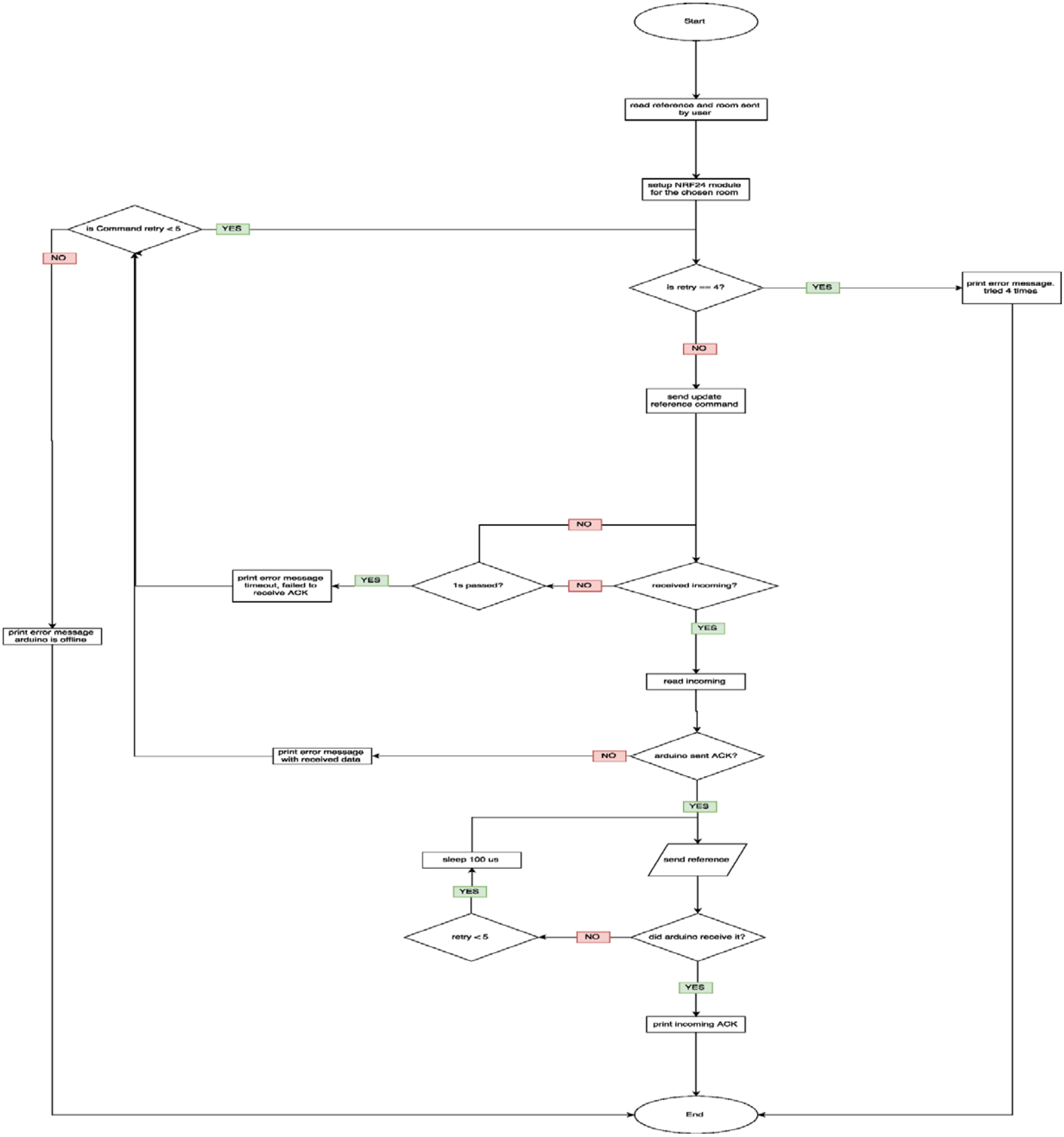


Figure 6-17: collector_light.cpp/collector.cpp temp Flow Chart

write_tv.cpp and write_door.cpp

write_tv.cpp is used to turn the TV on and off while write_door.cpp is used to open or close the garage door.

What the program does is the following:

- First, the guard module checks the queue and let the core program execute when it's our turn.
- Then the program invokes loop function with TV/garage state that the server sent to forward it to living room Arduino. It has a max of five retries to send command before exiting and reporting error.
- After sending the change TV/garage state command to Arduino, the program will receive either an ACK if the command executed successfully on Arduino's end or NACK if Arduino couldn't execute the command successfully.
- At the end guard module will remove the program ID from the queue to let other programs use SPI or NRF24L01+.

The two programs have a sending parameter that let them know which Arduino Raspberry Pi is to send to.

The command supposed to be called this way:

Note: [ ] means that you have to pass an argument

, means that you can choose only one of the two.

```
~ $ sudo write_tv [ 0 , 1 ]
~ $ sudo write_light [ 0 , 1 ]
```

0 means that the server wants to close TV or garage door.

1 means that the server wants to open TV or garage door.

6.3.2.3.2 write_door.cpp Flow Chart

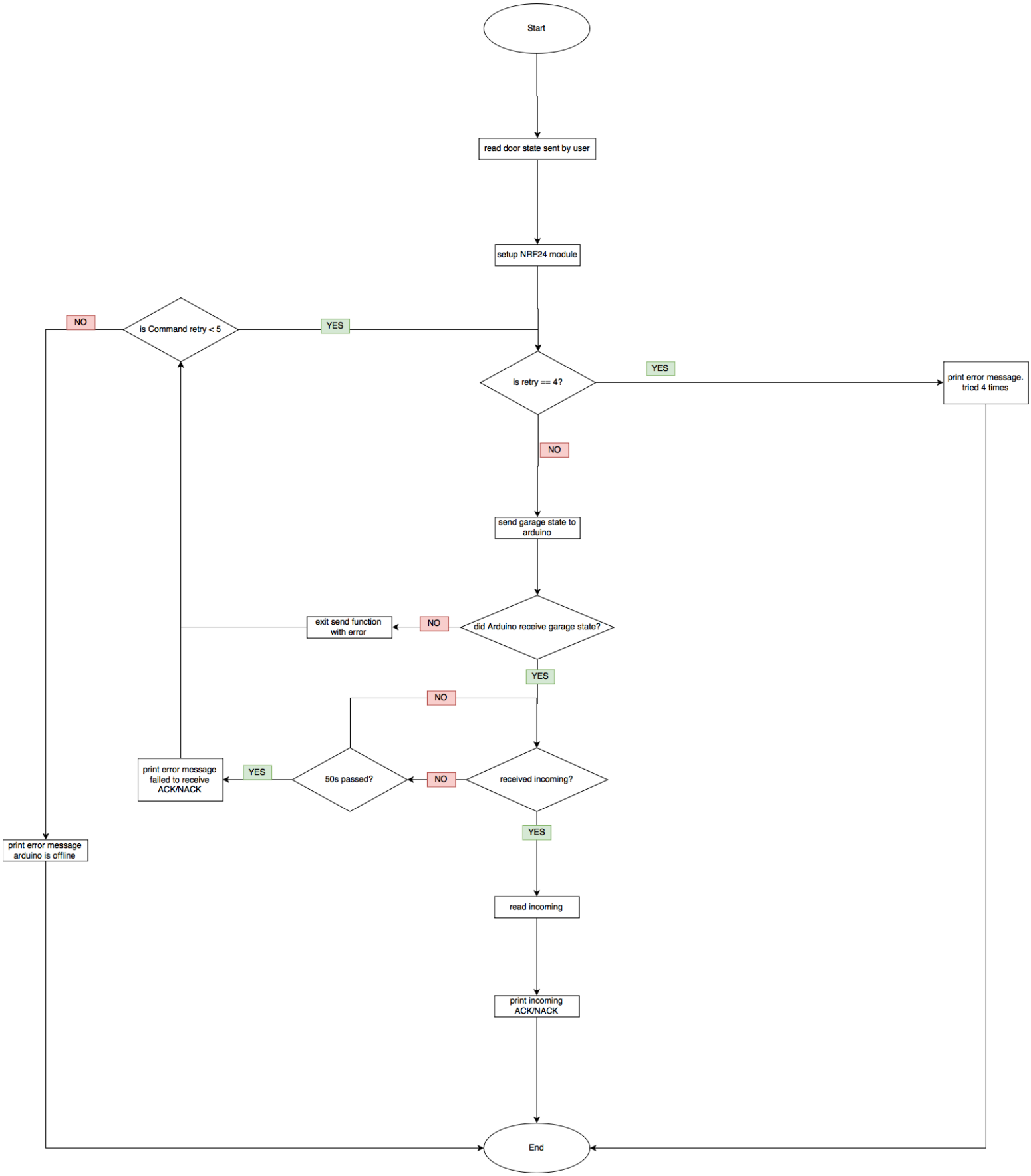


Figure 6-18: write_door.cpp Flow Chart

write_tv.cpp Flow Chart

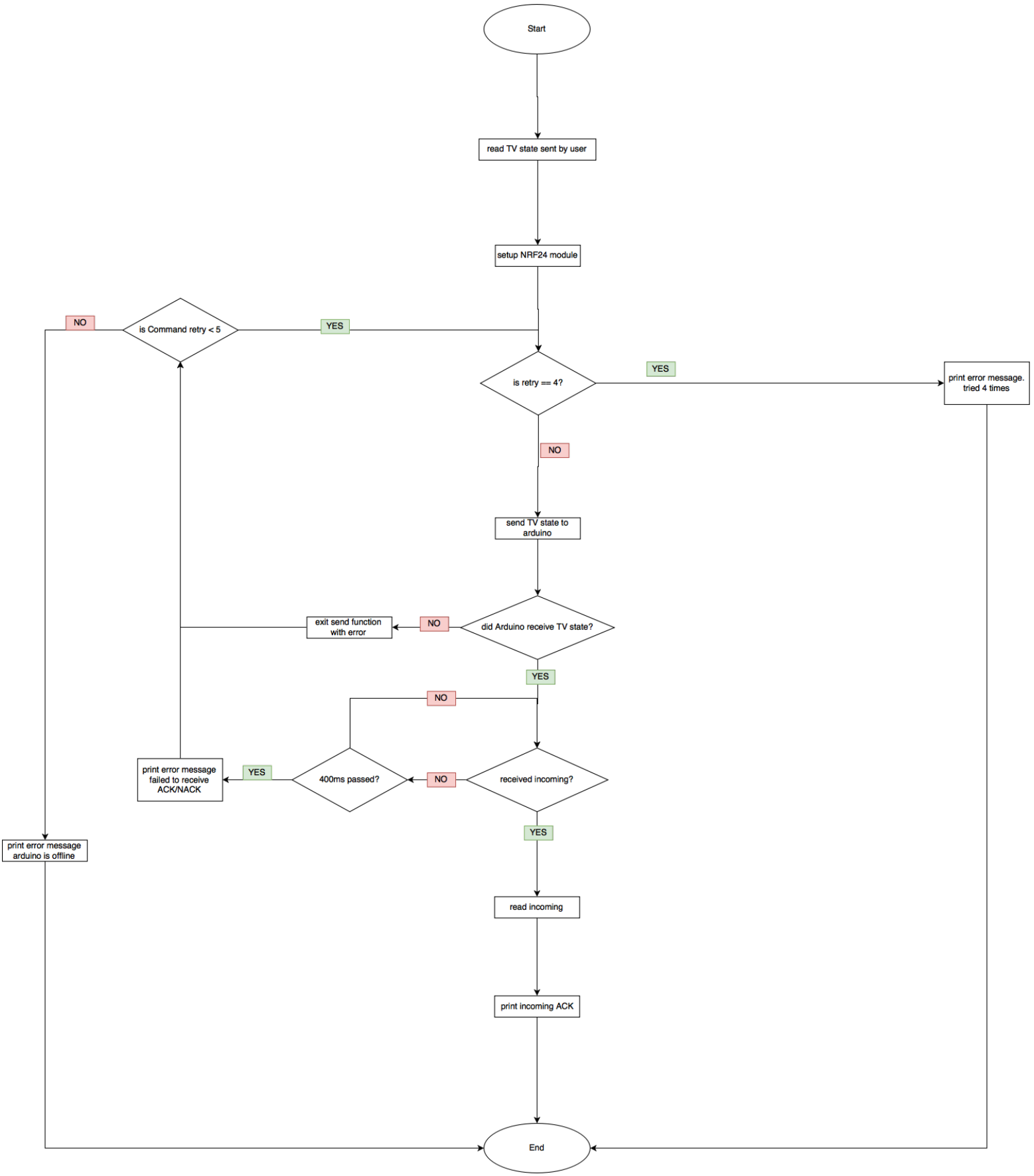


Figure 6-19: write_tv.cpp Flow Chart

6.3.2.4 Overview

The respond program was designed by request from Server Team to translate the SID of each used sensor to the correct command and to be sent to the Arduino that has this sensor.

The server will call the respond program like this:

```
~ $ sudo respond [sid] [state] <value>
```

Some sensors have only a state while other have a state and a value. So sometimes the value is optional and other times it's mandatory. If the sensor has a SID only but the server sent a value as well then the respond program will simply ignore the redundant value and carry on its function as expected.

If a sensor has a value then it must be sent otherwise the respond program will return an error and exit.

If the SID that was sent by the server is not used in the hardware then the program will print an error message that this sensor is not defined and it will exit.

Respond program is the glue that links the server to the microcontrollers. As you might have noticed, we didn't show how to use a single program from the programs we described above, mainly because we don't really use them. Respond program is the one who uses them.

Thanks to the system call SYSTEM respond program is able to call any of the program we made to communicate with Arduino using the protocol we made over the span on the project duration, when respond program maps the SID correctly to a certain sensor then it only needs to call system() with the program name and the parameters it received from the server and the program respond called will handle the transfer of the command to the microcontroller.

6.3.2.5 Respond Program Flow Chart

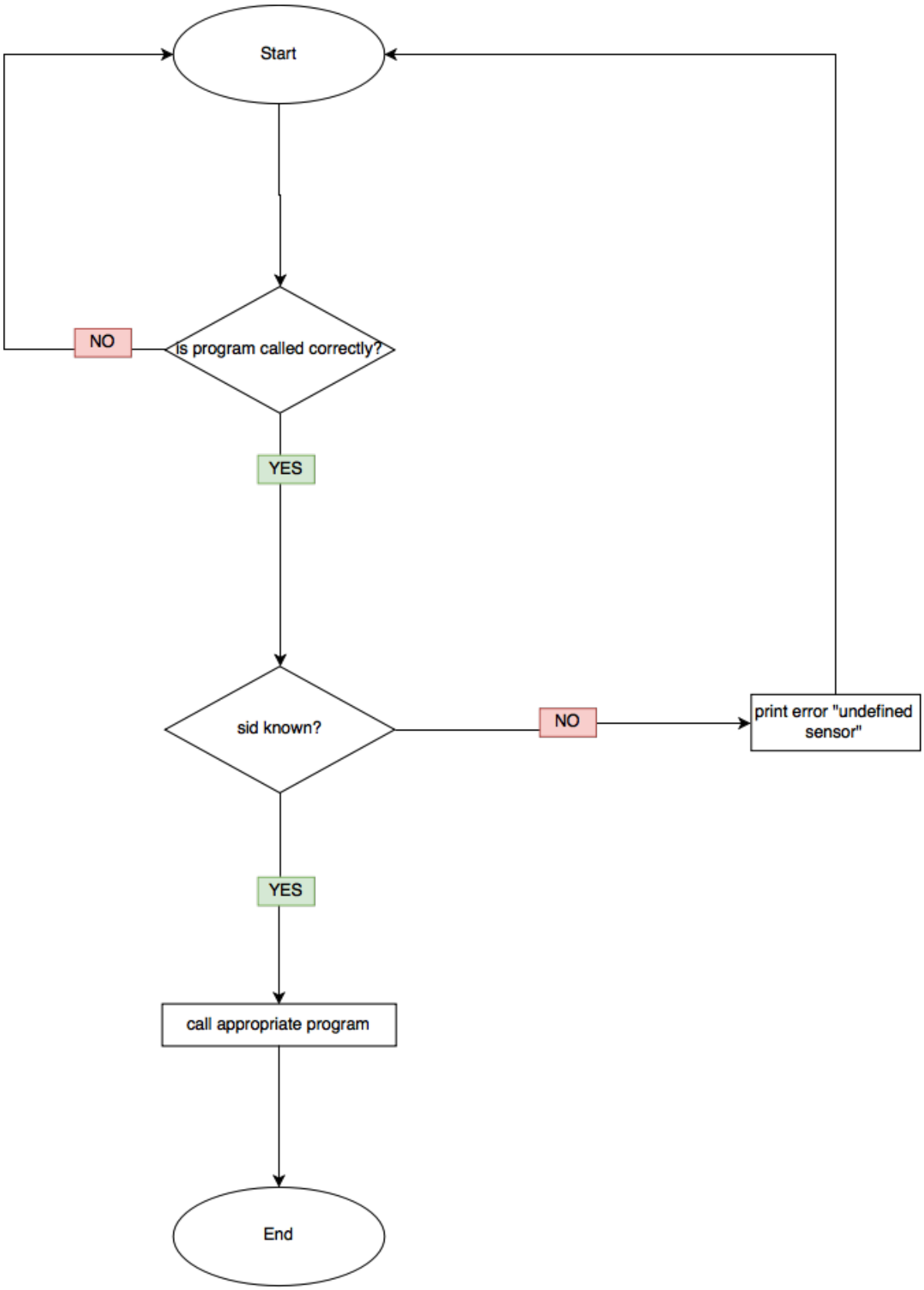


Figure 6-18: Respond Program Flow Chart

Read_loop program

6.3.2.6 Overview

The read loop program runs on startup and keeps running. Every 5 seconds the program gets the readings of the safety sensors and if the sensor found any danger then it will update the database. The program updates the database using a program we requested from Server Team named update.php and temp.php, you can refer to their description in the server documentation.

6.3.2.7 Read_loop Flow Chart

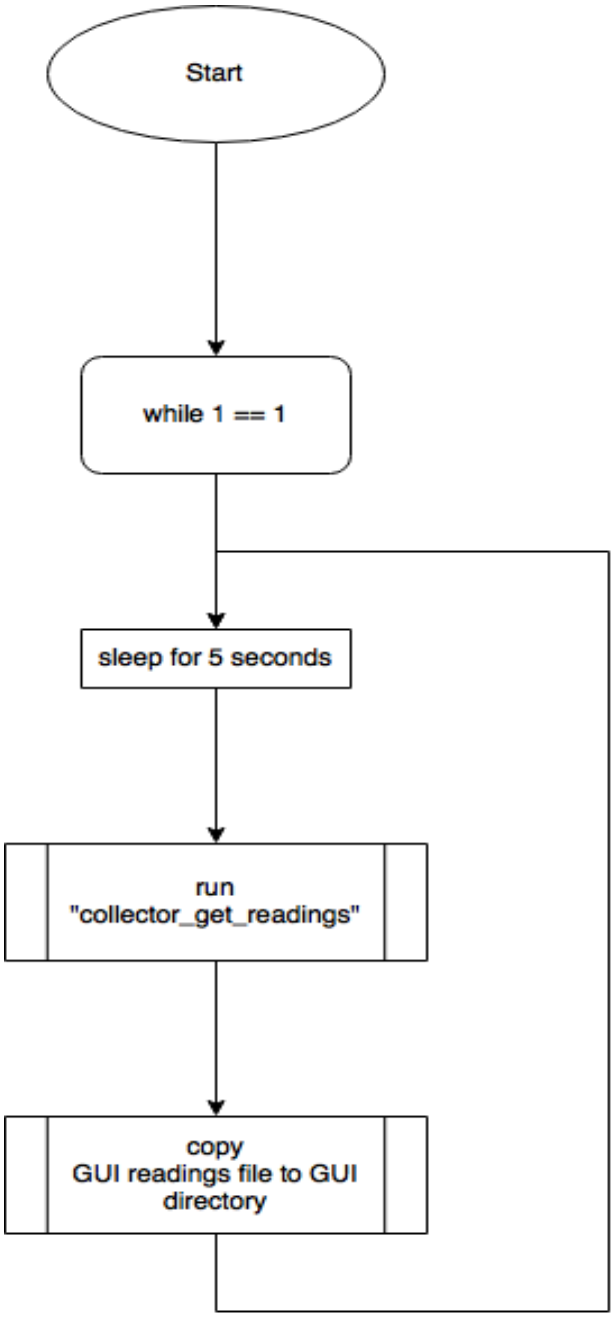


Figure 6-19 - Read_loop Flow Chart

6.3.2.8 Overview

The GUI program is like the Android App except it has more readings and we only added the available sensors to its interface.

Most of the analog output for most of the sensor is no concern to the server or the app, but since they're already connected we took those reading and display them in the GUI Read interface.

There is also a write interface so we can send references to the microcontroller just like the App. The Write interface uses system() to call the Writing programs just like Respond Program.

6.3.2.9 GUI interface

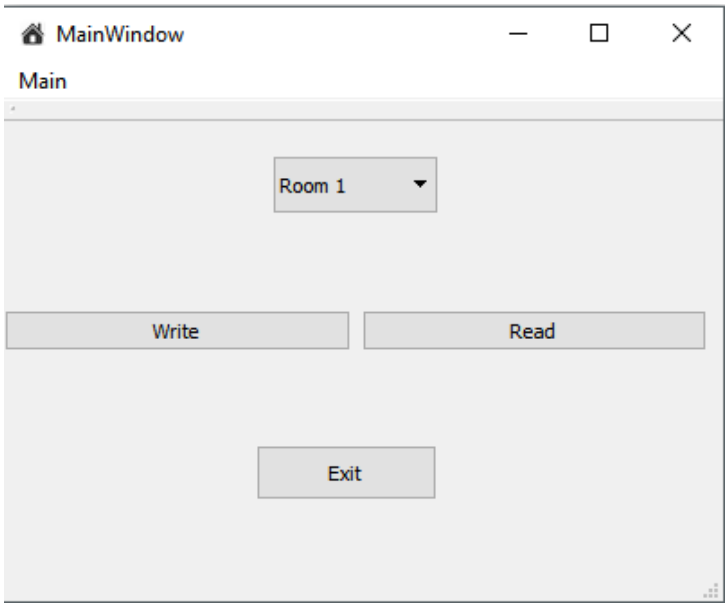


Figure 6-20 - GUI Main Interface

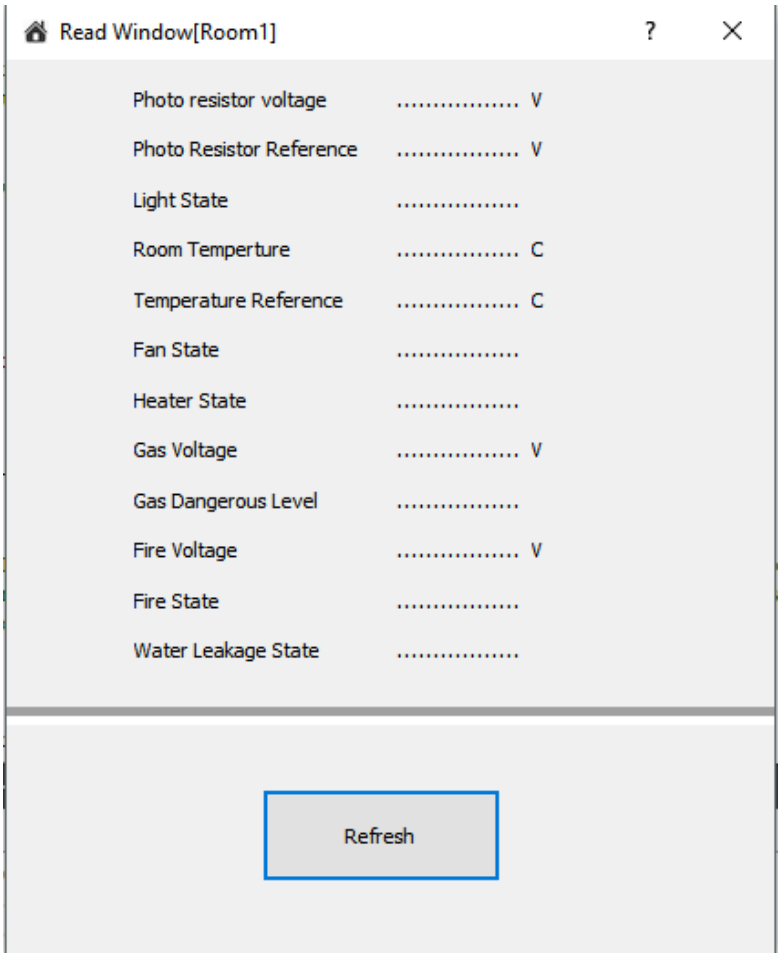


Figure 6-21 - GUI Read Interface

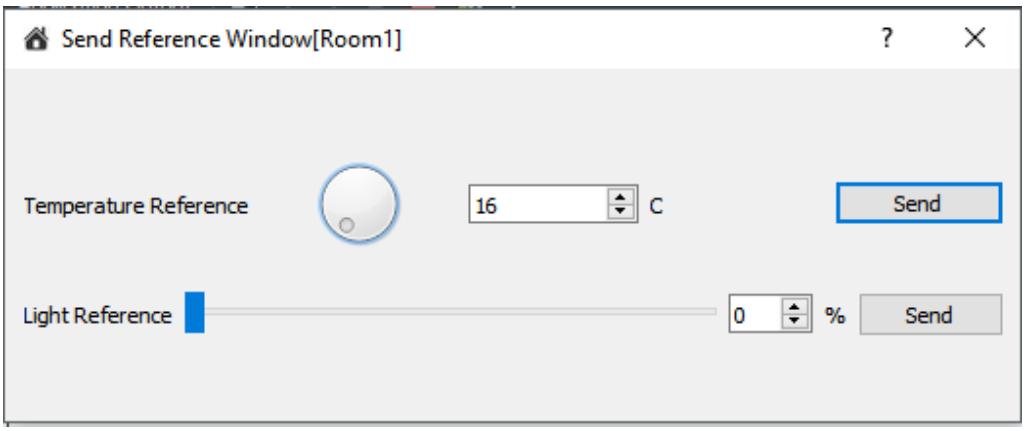


Figure 6-22 - GUI - Write Interface

6.3.3 Multi-Processing

The question that comes to mind when you come across a code that applies the modularity principle is that why did you use multiprocessing?

In fact, we actually wrote the code at the very beginning to multi-process all the different command that won't interfere with one another. But when we tried multiprocessing (using OpenMP) the single process was always faster. We neglected to read the Raspberry Pi zero Specs which stated that it had a 1GH single core processing unit. So any attempt to multi-process was converted to multithreading by the compiler so it took more time to execute.

6.3.4 Inter-process communication

At not so early stage of the project we used named pipes to communicate between the programs. The named pipes were a FIFO pipes.

This was the alternative of us making our own interposes communication technique which was the Guard program. The guard program used a queue which is FIFO file.

There were many problems with the named pipes that lead us to create our own inter-process communication protocol. To name some, the fact that everything in the named pipes method moves through the kernel so a simple error could halt our entire OS or crash it. Another problem was that when a program wants to send something to another program it will block the sending program until the receiver program collect the data. And the same goes for the receiver if we wanted to open the pipe to read it will also block the receiver until someone sends data on the same pipe. This method was meant for two programs only and in our case, there are multiple programs that need to use the SPI.

Of course we tried to solve these problems but we only make it worth, for example setting the transmitter program to a non-blocking I/O when there is no receiver only means to leave the data in the kernel. What if you keep sending but there is no one there?

We could have set the O_RDONLY for non-blocking flag and ran the chance of letting our program to turn into a halt and catch fire program.

So at the end, we decided that Named Pipes were too much of a trouble and make out own.

Chapter Seven

7 Conclusion

As we discussed before that our project's idea is an IOT system which can be built in buildings (i.e. homes, hospitals or hotels) to keep them safe from unexpected dangers (i.e. fires or leakage) and make it easy and safe for the humans to live in. we applied this idea on maquette which simulates a real home. Our system can connect the home and its owners by giving them the ability of monitoring and controlling some objects in their home remotely via internet from anywhere around the world using mobile application or website interface or via raspberry pi user interface. The advantage in our system is that it isn't limited to control one home but one user can connect to several homes.

Our project is helpful and has many benefits. Those are some: allowing the users to monitor and control their homes remotely, allowing the owners to keep track of what is happening now in their homes, allowing parents to keep eyes on their children, Safe the homes from fires, leakage and theft, make it easy for old people to control everything without others' help, facilitating living alone for the physical challenges and finally it's useful for the persons who are travel a lot to reassure their homes.

We used many technologies to implement this project successfully: raspberry pi, Arduino, android development, PHP, HTML, CSS, JS, and REDIS.

We have done many features with those technologies: Transferring data between the raspberry pi and the user interfaces via internet, making a connection between a raspberry pi (master) and multi-arduino kits (slaves) using WIFI, using REDIS to transfer data from the server to the raspberry pi, providing sync between all the connected devices, using notifications to concern the online users of most recent updates, collecting the data of sensors and updating the database by their current states instantly.

We will explain the scenarios that could be happened in our system, the first is when the user make update from the web or mobile interface, and the other is when the raspberry pi sends and update (i.e. according to alarm detection or user made update through raspberry pi user interface). For the first, we have a mobile interface (i.e. android mobile application) and web interface or website (i.e. the user can access the system using any internet browser), both of them are connected to the webserver which is connected to the database server and the raspberry pi. The raspberry pi is connected to multiple arduino kits via Wi-Fi and the arduino kits are hardwired to all the sensors. When the user updates the status of any object, from the mobile app or from the website, or change the preferred value for any of them, the server will update this data in the database and send it to the raspberry pi instantly via REDIS. Then the server will send a notification to all mobile clients via firebase cloud messaging. The second scenario is when a sensor detects an alarm (i.e. fire or gas leakage), or the user edit a value from raspberry pi GUI then the raspberry pi sends it to the server using http request, and the server will take the data and update the database and notify mobile users for an update occurrence.

At the first we should have done 31 sensors in the maquette but because of the high cost we sufficed with only one type of each different types of sensors. We implemented sensors like temperature sensor, light sensor, PIR sensor, flame sensor, gas sensor, water leakage sensor,

and humidity sensor and we implemented one actuator to control garage door. We used other electrical components to simulate the real home like TV, fan, LEDs as heaters and lamb for the light.

In our future plan, we hope that we can implement the 31 sensors around the maquatte, improve the raspberry pi GUI and add extra features to it, improve the mobile application and website design and finally implement this project in a real home.

8 References

8.1 References

[1] Professional Android 4 Application Development, (Reto Meier).

8.2 References to Electronic Sources

[2] (Redis, Redis documentation, n.d.). https://redis.io/
[3] (diyhacking, How to Make a Raspberry Pi Web Server, n.d.). https://diyhacking.com/raspberry-pi-web-server/
[4] (predis, How to user predis, n.d.). https://github.com/nrk/predis#how-to-use-predis
[5] (khan, n.d.). Firebase cloud messaging. https://www.simplifiedcoding.net/firebase-cloud-messaging-android/#CreatingPHP-Scripts
[6] (MohammadaliMirhamed, Firebase API, n.d.). https://gist.github.com/MohammadaliMirhamed/7384b741a5c979eb13633dc6ea1269ce
[7] (reids, Redis listener, n.d.). http://redis4you.com/code.php?id=012
[8] (Joshi, Using redis for windows, n.d.). https://www.linkedin.com/pulse/using-redis-windows-php-shekhar-joshi
[9] (Oloruntoba, Getting started with redis in php ,n.d.). https://scotch.io/tutorials/getting-started-with-redis-in-php
[10] (Sevilleja, The ins and outs of token based authentication ,n.d.). https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication
[11] (diyhacking, Web Based Automation For Your Home with Raspberry Pi!, n.d.) https://diyhacking.com/raspberry-pi-home-automation/
[12] https://www.php.net
[13] https://getbootstrap.com
[14] https://w3schools.com
[15] https://stackoverflow.com
[16] https://bradhussey.ca
[17] https://developer.android.com/reference/classes.html
[18] https://firebase.google.com/docs/cloud-messaging/
[19] https://stackoverflow.com/
[20] https://jsonformatter.curiousconcept.com/
[21] https://tutorialspoint.com/json/
[22] https:/ /github.com/nRF24/RF24
[23] http://users.cs.cf.ac.uk/Dave.Marshall/C/node23.html
[24] http://www.cplusplus.com/
[25] http://www.kegel.com/dkftpbench/nonblocking.html
[26] http://tldp.org/LDP/lpg/node7.html
[27] http://bisqwit.iki.fi/story/howto/openmp/
[28] http://supercomputingblog.com/openmp/tutorial-parallel-for-loops-with-openmp/
[29] http://bisqwit.iki.fi/story/howto/openmp/#IntroductionToOpenmpInC

9 Appendices

APPENDIX I:

Webserver source code on github.
https://github.com/Mustafa-Kamel/Connectvia

APPENDIX II:

Mobile App source code on github.
https://github.com/ahmedorabi94/MrHome

APPENDIX III:

Raspberry pi and microcontroller source code on github.
https://github.com/AnasKhedr/Smart-Home-2017