Indiana University Purdue University Fort Wayne

Senior Design

IOT: Smart House

Authors:  
Arif Chowdhury  
Austin Doehrman  
Josh Zuccollo

Supervisor:  
Dr. Guoping Wang

Course Coordinator:  
Dr. David Cochran
# Table of Contents

Abstract 3

1 Introduction 4

2 Problem Statement 5
   2.1 System Boundary ........................................ 5
   2.2 Interface Requirements and Definition .............. 6
   2.3 State Customer Need(s) ................................. 7
   2.4 State Customer-Defined Constraints ................. 7
   2.5 Requirements and Specifications .................. 7

3 Detailed Design 8
   3.1 Product Design Map ................................... 8
   3.2 Conceptual Design Alternatives .................. 9
   3.3 Criteria for Selection and Testing Among Alternatives .. 9
   3.4 Risk Mitigation - DFMEA ............................ 11
   3.5 Design Verification Test Plan .................. 15
   3.6 Design Validation Test Plan ................... 16

4 Component Definition and Planned Build 16
   4.1 Hardware ............................................. 17
      4.1.1 Raspberry Pi ...................................... 17
      4.1.2 Sensors ........................................ 17
   4.2 Server .................................................. 17
      4.2.1 MQTT ............................................ 18
      4.2.2 Mosquitto ....................................... 18
   4.3 Mobile Application ................................. 18
      4.3.1 Android ........................................ 19
      4.3.2 User Interface ................................. 19
   4.4 Building Plan ....................................... 19
   4.5 Design schematics .................................. 20

5 Test Execution 20
   5.1 Software Testing and Analysis .................. 20
   5.2 Physical Testing and Analysis ................ 21

6 Bill of Materials and Cost 22
# TABLE OF CONTENTS

7 Project Plan and Timeline .................................................. 22

8 Implementation ................................................................. 25
   8.1 Final Build .................................................................. 25
   8.2 User UI ...................................................................... 28
   8.3 Testing Results ......................................................... 31

9 Accomplishments and Issues ................................................ 31

10 Conclusion ........................................................................ 32

11 References ......................................................................... 32

12 Appendix A - List of Figures ............................................... 33

13 Appendix B - List of Tables ................................................. 33

14 Appendix C - Source Code ................................................... 34
   14.1 Hardware Code .......................................................... 34
      14.1.1 Master Device Control ......................................... 34
      14.1.2 Camera on Motion ............................................... 39
      14.1.3 Motion ............................................................... 40
      14.1.4 Sound Detect ....................................................... 41
      14.1.5 Temp Sensor ......................................................... 42
      14.1.6 Water Detect ......................................................... 42
      14.1.7 PCF8591 .............................................................. 43
   14.2 Mosquitto Configuration .............................................. 45
      14.2.1 mosquitto.conf .................................................... 45
      14.2.2 password ............................................................ 46
   14.3 Web Client ................................................................. 47
Abstract

The Internet of things (IoT) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items—embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data.[1] IoT allows objects to be controlled and gather information remotely across the already established network.

The goal of this project is to create an IoT Smart House that can monitor specific criteria, as well as control specific devices. Data to be monitored are: temperature, humidity, movement, water, and power. An outlet, camera, and microphone can be controlled by the user from any mobile device. The user will be notified in any change in the monitored data, if it changes beyond their given range. The device must also work while power is out in the house; which means a battery back-up and Internet hot spot must be included as well.
1 Introduction

For the process of designing the Smart Home system, there are a few key steps covered in this report. First, the problem statement for this project is defined, which includes defining the boundary of the system, the interfaces needed between the system and outside factors, the needs of the customer, the customer’s constraints, and the requirements and specifications defined by the members of the project team.

After that, a detailed design is produced. This is where the functions of the system are defined and broken down into simpler functions, and where solutions to carry out these functions are found. There are generally several different possible solutions to implement each function, so analysis is performed to compare these solutions to find the right one for the project. Risk mitigation is performed and a test plan is created in order to ensure that the construction of the chosen design proceeds smoothly, and that any potential problems are accounted for.

The chosen design components of the IoT Smart Home are expanded upon in the component definition and planned build sections. First the design for the hardware is examined, which includes the physical devices used to construct the IoT network. After that the server software chosen is analyzed, which looks at the server implementation chosen to allow communication between devices in the network. The development of the mobile user interface is expanded upon after that, which includes the mobile platform used and the plans for the capability of the mobile application. The final plan for the implementation of all of these components are then shown in the building plan and design schematics.

As the Smart Home is being implemented, testing and analysis needs to be done to ensure that everything is working as it should and is meeting the customer’s needs. This is separated into testing of the software components and the hardware components.

Finally, a bill of materials is produced to ensure that the system can be implemented at a reasonable cost, as well as for the benefit of listing out everything needed to begin construction of the Smart Home system.
2 Problem Statement

The proposed project is to design and implement a system that will monitor specified house data to ensure the house remains secure. It will also be able to control specific functions within the house as well. In order to fulfill this requirement, our system will need to follow certain guidelines described below. It must operate within the specified guidelines, while also still be reliable and cost effective to make.

2.1 System Boundary

![System Boundary Diagram]

Figure 1: System Boundary Diagram

The system boundary diagram above shows what can be controlled within the system and what must be interfaced with outside the system. Everything within the system boundary can be designed and configured easily and minutely. In the diagram there is the Raspberry Pi, which is the development board used in implementing devices within the smart home. There is also the MQTT Server, which handles the flow of data between devices within the Smart Home’s IoT network. The mobile application is the software developed on a mobile device that the user can use to control and receive updates on devices within the Smart Home. The control circuit
describes the system that controls the power provided to the Raspberry Pi, and allows for the use of power from either a battery or a wall plug. Communication between components describes what methods of transmission between devices are being used within the system, such as through the Internet or through wired connections. Internet Protocols describes how data is formatted when sent through the Internet, and for this project involves the use of MQTT. The code used describes what languages the different components of this project are being developed in, such as Java and Python. The sensors represent the devices within the Smart Home that send their data through the Home’s network to provide up-to-date information on the house’s status.

The Raspberry Pi boards used can be loaded with whatever necessary software is needed to run Sensors within the Smart Home, and coded with the desired behaviors. The MQTT Server can be configured to restrict access to certain known devices, and data sent to and from it can be encrypted to preserve data security. The Mobile Application can be coded to interface with the chosen server and control any chosen devices within the Smart Home system.

2.2 Interface Requirements and Definition

The systems that cannot be directly controlled need to have the appropriate interface to allow the Smart Home to work with them. Data through the Internet has a possibility of being intercepted for nefarious purposes, so that data needs to be properly secured through device verification and data encryption. The connection to the Internet is not guaranteed, and steps must be taken to ensure that there is a minimum of downtime if the system loses access to the Internet through WiFi. The backup cellular hot spot connection is not guaranteed to have good reception, and this should be considered when choosing a provider for this capability. For the power grid, power outages are a possibility, and backup methods of powering devices should be considered. Government regulations will mainly need to be followed for wireless communication, so FCC regulations must be read and understood to ensure that they are not being broken. These regulations should be simple to follow, due to the fact that only WiFi and 4G signals are being produced by the system.
2.3 **State Customer Need(s)**

The customer for this project is the ECE Department of IPFW. A few elements are highlighted within the project proposal which guided the development of the design of the Smart Home. The central component of this proposal is the emphasis of connectivity of the Smart Home over the Internet through an Internet of Things style network. All the devices within the Smart Home need to have a connection to the Internet to allow a remote user to receive information on the state of the house and control devices within it.

2.4 **State Customer-Defined Constraints**

There are several restraints listed within the ECE Department’s project proposal. In terms of cost, a budget of $300 dollars is to be used in the construction of the Smart Home platform. The time constraint is that this system should be implemented before the end of the fall semester of 2017 at IPFW. Certain sensors need to be implemented in the system, which included detection of gas, humidity, temperature, and water leakage, as well as control of house lights and power outage detection. Systems for maintaining Internet connection in areas where WiFi are unavailable were also required, which involved the ability to create a cellular connection. The ability to connect to the Smart Home through a mobile device was one of the more important requirements for this project.

2.5 **Requirements and Specifications**

Building off of the customer’s requirements and restraints, the project team needed to create more concrete goals involving data usage system response time. The latency of messages sent between devices within the Smart Home and a user’s mobile device should be fairly low, generally less than a second. The data sent from the devices also shouldn’t exceed more than 10 MB per hour to prevent excessive cost of data usage and slowdown of wireless networks overall. To ensure consistent connection, a 4G hot spot module will be attached to the device control board within the Smart Home that will transmit through the 4G LTE network when WiFi is unavailable. There will also be an attached circuit to switch to battery
power if the house experiences a power outage.

3 Detailed Design

The detailed design is the current plan for the development of the Smart Home, where the functions that the system are expected to perform are broken down and solutions to carry out those functions are considered. These are described as Functional Requirements (FR) and Physical Solutions (PS) respectively. For each FR several PS’s are analyzed by comparing strengths and weaknesses to determine what is the best way to carry out each function.

3.1 Product Design Map

![Figure 2: Top Level Decomposition]
The top level of this decomposition shows the most basic functions that the Smart Home needs to work. These include the ability to know the status of the house through the use of sensors, the ability to control aspects of the house through a mobile application, and the ability to connect all of the components together within a network. The solutions to these functional requirements make up the basis of the Smart Home system.

### 3.2 Conceptual Design Alternatives

For design alternatives we focused on the high level FR’s, and chose different PS’s that could meet the same needs. The reason for this was the higher level PSs have more weight in the overall project than the lower ones do. Having a problem with a top level design is going to cause many issues once the building starts, and having an alternative ready right away, in case of an error, can help prevent the project from falling behind schedule.

For FR1.1 there exist a few choices on how to connect devices in the Smart Home to the IoT network to allow measurement of data and control over the devices. All of them involve different types of control boards running the appropriate software. The alternatives examined were the Raspberry Pi, TI Launchpad, and the Arduino Yun.

FR1.2 involves choosing server software to allow control over the connection between devices in the IoT network through the Internet. Two alternatives examined were the Mosquito broker and a cloud service called PubNub. These both use the MQTT message protocol which is considered ideal in an IoT system.

FR1.3’s alternatives were mainly a choice in which mobile platform to develop on, but it also involved choosing which language to code the application in. The main choices were developing on either Android, Windows, or iOS.

### 3.3 Criteria for Selection and Testing Among Alternatives

The Raspberry Pi was the board chosen for the implementation of the Smart Home, and when choosing a board to develop the system’s devices on, several important factors are considered. For one, as opposed to boards such as the Arduino Yun, multiple processes are able to run at
once on the Raspberry Pi. There is also a wide array of libraries available for Python development on the Raspberry Pi, allowing for simplification of implementation. The server software Mosquitto is chosen over PubNub for several reasons. Mosquitto is open-source, and is attractive due to the fact that developing with it doesn’t incur a cost and can be used flexibly, but it should also be noted that there is generally more work involved in the implementation of it. PubNub is only free for low traffic volume cases, where exceeding a certain amount of connected devices or sent data can incur a fee. The mobile application platform has several choices for development, and all seem equally good for development. The main factor in this decision is the availability of the platforms for testing purposes. All of the team members for this project possess Android phones, so that was what was chosen for development. The particular development environment platform was chosen to be Java JDK, and this was chosen due to the wide array of libraries available for use in MQTT communication through the IoT network.
3.4 Risk Mitigation - DFMEA

Table 1: Probability of failure Table

For any project, the team should be aware of any kinds of failure the system might go under and there probability of occurrence. Table 1 provides an insight and a guide to identify the probabilities of these expected and unexpected failures. As shown in Table 1: The red region is the defined to be highly likely, yellow region indicating Occasional failures and the green region indicates the least likely occurring failure regions. These probabilities shown are given rankings to give a weight to these failure occurrence probabilities according to their degree of impact on the final design. 10 being the most likely and the most impactful and, the 1 being the least likely and least impact to the design.
Table 2: Detection of Failure Table

The above table was used, in conjunction with table 1 and table 3 to create Design Failure Mode and Effect Analysis (DFMEA) Table. The purpose of this table is to check how an error can be detected. The green section is for when a detection will always be detected in someway, the yellow for when a detection is not detected and may allow for further errors, and the red section is for errors that are very unlikely to be detected. As with the other tables the goal is to end with every potential error have a solution that puts the error in the green zone.
For this project, it is necessary to define what different kinds of failure severities are there and what kind of effect they will have on the final design. These answers are on Table 3 above to help define them. Furthermore, different failure severities have been assigned rankings which defines their degree of impact on the final design. The most dangerous level at the top indicated in the red region is "Hazardous without warning" and this has been assigned a ranking of 10 and the least dangerous level indicated at the green region is "None" and has been assigned a ranking of 1. According to this table, the higher the ranking the more severe the failure is.
The goal of the project was to keep every design decision into the green and blue areas as much as possible. Any part that fell into the yellow must be monitored closely to ensure a complete failure will not happen. The red and brown areas were avoided at all cost, as the risk is too high to take for this project.

Finally, referring to Tables 1, 2 and 3, the team has been able to compute their DFMEA table (Design failure Mode and Effect Analysis). The mobile application can go through 2 kinds of common failure. The first being disconnection with the server and, the second the application freezing in the middle of running. When the application is disconnected from the server, transaction of useful data is stopped and without these data the entire smart system is dysfunctional, hence the severity of this failure is given a 9. Since this failure occurs rarely it has been assigned 3 and detectability is 8 as it can easily be noticed when the system is not running properly. To help the user of this problem a user notification would be generated saying “Cannot connect to server”. Next, when the application freezes the user is not able to be in control of the system, as a result the application shuts itself down. The severity of this failure is logically very high and has been given a 7. It again doesn’t occur often and has been assigned a probability of 3. Detectability of this issue is very easy, the user will just see that the application is not working and will shut down and hence it has been given a 7. This issue can be improved by shortening the code of the application which takes less processing power.
3.5 Design Verification Test Plan

In order to verify that the design is meeting requirements, an initial test was performed on software and hardware. The testing started with playing around with a sample android application already available on the Android playstore. Main goal of the initial testing was to see if the team can achieve to control an LED with the mobile application. The application was then registered to a specific MQTT server and so was the sensors. The team achieved connect an LED and a temperature and humidity sensor. The LED and the surrounding control was set as a specific topic and the application was subscribed to that topic. The team was successfully able to turn on the LED with a "1" command and turn off with a "0" command. Also the "temp" command resulted in the phone application to receive the
current temperature and the humidity sensor. All these commands were run through the sample application on an android device. With achieving positive results from the initial testing, the team was able to grasp a clearer vision of the potential of this design. Further research on the Internet showed the team a couple of similar IOT designs which allowed them to choose this choice of hardware and software to be the final design.

3.6 Design Validation Test Plan

In order to ensure that the final Smart Home design meets the customers needs, some questions must be asked. For one, is the design able to cope up with new up coming technology? The answer to that is yes. There might be future technological advancements for the sensors and hardware. It is highly unlikely to have a new operating system that is more popular than android. Even if there is then building and designing a new application will require the same amount of time to that of an android application. On the other hand, this design is very adaptable to change. If there are more efficient sensors out in the market, the old sensors can easily be replaced and installed. For the server, it is again the same. It will not take a lot of effort to install a new server as well. But any server being more popular than Mosquito is again highly unlikely, as it is already a free lightweight and incredibly fast server. All this, allows the team to call this design a valid one.

4 Component Definition and Planned Build

The Smart Home system can be separated into three main components. Those are the hardware within the Smart Home, the server software that manages the flow of information through the network, and the mobile application for the user. These components all need to be implemented and connected to each other in order for the Smart Home to perform its most basic functions. The specifics of these components will be described in the sections ahead, as well as how their connections are planned.
4.1 Hardware

The goal of the hardware is to collect data, and be able to control the specific functions that are needed. The main piece of hardware used is a Raspberry Pi with sensors. A circuit with a relay is used to switch between the wall power and a battery back-up in-case of a power outage; this ensures constant power to the Raspberry Pi and allows constant monitoring even when power is out.

4.1.1 Raspberry Pi

For this project the main piece of hardware used is a Raspberry Pi 2. The reason for choosing came from it being able to run multiple programs at once, while other competitive options can only run one program at a time. This allows for faster response time, and lower CPU usages of the Pi itself. A program can be run only when it is called upon, rather than having only a single program that must run every part of the project when a single item is asked to be updated.

4.1.2 Sensors

The sensors being used are: a camera, motion detection unit, water sensor, gas sensor, sound sensor, a RF transmitter and receiver kit.

Some sensors will work together to achieve a goal while others will work on their own. For example the motion detector will enable the camera to take a video or pictures when it is triggered. The water and gas sensor will each work alone to send a notification in the event of a change in either one. The RF transmitter and receiver kit will be linked to a wireless outlet switch allowing remote access to an outlet and control the power coming from the outlet.

4.2 Server

In order to control where and when sensor information is sent, a server is required. The purpose of the server is to relay information between devices connected to the IoT network. The server needs to be capable of connecting to a variety of sensors and mobile devices securely and efficiently.
4.2.1 MQTT

To meet our goals for server performance, the MQTT protocol was selected. MQTT stands for MQ Telemetry Transport, and it is a lightweight transport-layer protocol using a publish/subscribe model that allows connected devices subscribed to a given topic to be updated whenever information is published within that topic.[2] This is a commonly used protocol in Internet of Things networks and has become an OASIS standard. This message format is very efficient, with each message sent only containing essential information in the couple bytes of the packet. This protocol is ideal for systems where low bandwidth is desired due to the small amount of overhead as well as the method of distributing messages.

4.2.2 Mosquitto

Mosquitto is the MQTT server implementation chosen for this project, which is run on a Raspberry Pi. One of the reasons Mosquitto was chosen was that it is open-source, meaning that it is free to develop with it. Mosquitto also has a lot of flexibility in the configuration of the server. In order to ensure security of data, Mosquitto can restrict access to topics to certain users, and the addition of SSL will allow data to be encrypted before being sent through the internet.

4.3 Mobile Application

For this project, the ultimate goal is to control certain aspects of the home remotely. The most convenient way to do that is for the users to have access to a mobile application. This application will receive readings from the sensors connected to the house via the server and report to the user about the changes that took affect in the smart home. User will also be able to take actions related to the affects that took place using this mobile application on his smart phone. For example, turning the thermostat up to keep the house at a certain temperature or turning on and off specific lights around the house.
4.3.1 Android

For this project android application was selected. The decision was easy to make, due to all the senior design teammates’ availability to an android device. This will help the team to do individual testings and evaluations. Furthermore, an android application can be designed using JAVA, which is a very robust and versatile programming language. There are plenty of references on-line in making the design process of the application easier. Finally, the team tested out some sample android applications already available in the market with initial testing. The results from the tests done were all positive and gave the team a clean site that an android application for this project is very doable.

4.3.2 User Interface

The user interface should be designed to be easy to use by anyone. This will be accomplished by have a user friendly user interface (UI). The UI will be clearly labeled, and have simple button activation for controlling the required functions. The measured data will be displayed at all times; helping in making the application fast and easy to use. A refresh button will also be included, which will pull current data measured at that exact second. The ability to create custom functions in the application will also be included. A user can create a new icon, and select what functions they would like to activate when that custom function is ran. This helps lower resource cost, and allow each user to set up the application exactly how they prefer to best suit their needs.

4.4 Building Plan

There are a few key components to be built for the Smart Home system. Overall, all of the components must be connected together in an Internet of Things style network, where all devices are connected to a central server which controls the flow of data. More locally, the sensors within the Smart Home must be connected to a sensor hub that sends the data to a server. There are also some power control components connected to the sensor hub that prevent loss of power for the system.
4.5 Design schematics

One of the main schematics used for the design of the Smart Home is the diagram showing the layout of an IoT network. It provides the basis of the design, and shows how the system is basically laid out. The sensor/actuator portion is where the devices of the Smart Home are implemented, the IoT Middle-ware consists of the server used to transmit data between devices, and the Application is the mobile application used to keep up-to-date on and control devices within the Smart Home.

Another important schematic is that of the sensors of the Smart Home devices that are connected to a central development board. This board then has additional wired systems that control the source of power for the board. There is not a finalized circuit diagram for these connections, but it is planned to be completed before the building of hardware systems begins in the fall of 2017.

5 Test Execution

The test plan would include the team adding different kinds of sensors to the system and trying to get readings and being able to control those sensors. Also it would include testing of the server and the android application.

5.1 Software Testing and Analysis

The software components of the Smart Home system that require testing include the MQTT server and the mobile application. For the server, testing will involve ensuring that the implementation of different aspects of the server are successful. This includes the ability to connect to other devices
to the Internet, as well as the ability to properly secure data sent between devices.

The mobile application will be tested using the Android Studio. The entire code of the android application will be programmed using this software development tool. What is useful about this tool is that it also enables the users to simulate their application using an android phone emulator. The android emulator is an on-screen virtual android mobile device that has all the functions of an actual android phone. Through this emulator testing of the android application will be done for the connectivity to the server and also being able to send the right messages to the server for the raspberry pi to get them. Testing for the connectivity to the server would include the correctness of calling the MQTT protocols.

5.2 Physical Testing and Analysis

The hardware components of the Smart Home that are to be tested include the sensors of the devices within the Smart Home as well as the Raspberry Pi development boards that are used to implement the system. Sensor connections to the board will be tested to ensure proper function of connected devices. The backup power systems will also be tested to ensure that these devices can function for a while without a guaranteed power supply.
6 Bill of Materials and Cost

<table>
<thead>
<tr>
<th>Part</th>
<th>Part #</th>
<th>Supplier</th>
<th>Price</th>
<th>Quantity</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 2 Model B V1.1</td>
<td>Raspberry Pi 2 Model B</td>
<td>Raspberry Pi Foundation</td>
<td>$42.00</td>
<td>2</td>
<td>$84.00</td>
</tr>
<tr>
<td>Humidity and Temperature Sensor</td>
<td>B014KOSFLC</td>
<td>SunFounder</td>
<td>$9.99</td>
<td>1</td>
<td>$9.99</td>
</tr>
<tr>
<td>MQ-2 Gas Sensor</td>
<td>B00NUOB50</td>
<td>Waveshare</td>
<td>$7.99</td>
<td>1</td>
<td>$7.99</td>
</tr>
<tr>
<td>Sound Sensor</td>
<td>0DS</td>
<td>TolakO</td>
<td>$6.99</td>
<td>1</td>
<td>$6.99</td>
</tr>
<tr>
<td>AD/DA Converter</td>
<td>PCF8591</td>
<td>Uxcell</td>
<td>$2.67</td>
<td>1</td>
<td>$2.67</td>
</tr>
<tr>
<td>PIR (Motion) Sensor</td>
<td>HC~SR501</td>
<td>MPJA</td>
<td>$4.30</td>
<td>1</td>
<td>$4.30</td>
</tr>
<tr>
<td>Wireless Remote Control Outlet Switch</td>
<td>608939749917</td>
<td>Etekcity</td>
<td>$11.98</td>
<td>1</td>
<td>$11.98</td>
</tr>
<tr>
<td>433Mhz RF transmitter and receiver link kit</td>
<td>6461060</td>
<td>SMAKN</td>
<td>$6.66</td>
<td>1</td>
<td>$6.66</td>
</tr>
</tbody>
</table>

Table 5: Bill of Materials for Project

The budget for this project is $300, which is more than enough to purchase all the necessary components for the Smart Home system. The current bill of materials covers the cost of Raspberry Pi development boards, the required sensors for the Smart Home devices, a backup battery to prevent power loss, and an outlet switch and RF Tx/Rx link kit to control the source of power in the system.

7 Project Plan and Timeline

Over the course of the spring semester the project was developed following a standard where periodic checkpoints coincide with reviews that evaluate
the progress of the project’s design. These reviews the System Requirements Review (SRR), the Preliminary Design Review (PDR), and the System Verification Review (SVR). The time-line above shows that certain aspects of the design were to be completed by each review. For the SRR, the problem statement and system boundaries were defined in a conceptual design in order to limit the scope of the project and guide the Smart Home’s development. After that the PDR was prepared for by creating a detailed design decomposition, a testing plan, and a risk management plan to be used in the fall semester of 2017 in order to minimize risk while implementing the design. Dependencies between components were also examined at this stage in order determine the complexity of the implementation of the different components, as well as to find where these dependencies could be removed to decrease the complexity of the final design. The different alternatives for each Functional Requirement within the design decomposition were weighed against each other in order to determine which was best for the design. The last review, the SVR, is where the design is finalized. The final decomposition of the design is finalized, and the materials required by the Smart Home system are found and listed in a Bill of Materials.

![Timeline for Spring 17 Semester]

Figure 5: Timeline for Spring 17 Semester
The above figure shows the general plan for development moving into the fall semester of 2017. This semester is mostly focused on the building and possible refinement of the design. As a final prototype is constructed, it is tested for verification and validation in order to ensure that the design is meeting all of the customer’s requirements and needs. Testing is split into hardware and software components, as these require two different types of test plans. The hardware testing involves verifying that all sensors are providing accurate information and that all hardware systems are properly working, such as the power control system. The software testing involves ensuring that data is sent securely and efficiently between devices.
8 Implementation

Using the design and testing plan developed above, the IoT system is constructed. The different components of the system are each implemented using the guidelines set in place by the design, which allows these different parts to properly interface with each other. Issues were dealt with as they arise, and appropriate measures are taken.

8.1 Final Build

![Final Hardware Enclosure](image)

Figure 7: Final Hardware Enclosure
In the final design of the Smart Home, some components are changed to allow completion within the deadline, but most features in the design are successfully implemented. One of the biggest issue this design faced is with the user app. Initially the goal was to design a mobile app for the user to be able to control and monitor different aspects of the home. Upon that the final decision for the plan was to build an android app for this IOT Smart Home. Unfortunately, the team struggled to complete this goal within the allotted time. It became essential to find an solution to this problem or to find an alternative if sufficient progress was not shown. The reason behind this issue is lack of software development experience. An alternative was proposed which turned out to be the solution to the problems. Building a web application for this IOT smarthome was the next best option. This web application needs HTML, Javascript and CSS programming. MQTT protocols were available for HTML for this IOT system to work. Without any experience with HTML, Javascript or CSS the team was able to implement this webapp. The programming language was very easy to learn with the help of very informative, practical, simple tutorials and instructions found online. This webapp had a couple of major advantages over the mobile phone application. The main advantage was that this webapp could run on any web browser there is available. It was not dependant on operating system, android IOS or windows, all it needed was a web browser. The
second advantage was that it was very file size friendly. Most apps nowadays are a couple of megabytes in size but this app was only a couple of kilobytes. Hence this took much less space on the user’s device. On completion of this webapp, it had a very user friendly UI which had different buttons and information logs through which the user could simply monitor or control different aspects of their home.

The MQTT server has the ability to ferry data between the web client and the Smart Home, and is configured to only allowed registered devices to send and receive information through it. The web-client has a user-friendly interface that provides time-stamped information on important information the Smart Devices detect, and allows sufficient control of these devices.

The Smart Devices themselves are connected to a central controller within a hand-made casing seen in the Figure 7 above. The devices within are able to perform their functions and send accurate data to the web-client, and can be activated or deactivated according to preference. Apertures for the motion sensor and camera are created such that the devices fit snugly within them, minimizing the exposure of internal devices to the outside world.

In order to ensure the system is always powered a batter backup was installed on the system. The wall power and battery are controlled with the use of a relay, which will switch to battery power when the wall power is gone. The diagram showing the relay connections is shown in figure 8 above. Due to the nature of the relay the pi will restart on the switch from wall power to batter and vice versa when then wall power is active again. The Pi is able to then send a notification to the web client about the power source change.
8.2 User UI

Figure 9: User UI 1

Figure 10: User UI 2
Figure 11: User UI 3

Figure 12: User UI 4
The above images show the User UI that was created for this project. Each function that is being monitored, or controlled, can be activated from the UI. The goal was to create a simple to use and intuitive UI for this project. By having an image along with text, and simple to use on/off buttons that goal was achieved. The UI will also have pop-up notifications during certain criteria; such as when connecting and disconnecting and when asking for a specific function. At the top of the screen is also a status window, shown in figure 9, that will show the most recent request, or data
that was asked to be shown. This can also be used to ensure that user request have successfully have been sent and received by the server.

8.3 Testing Results

Each component of the IoT system has a variety of necessary tests to ensure that the system works as designed. The MQTT server’s ability to connect to the web client and Smart Home controller was tested with and without user verification, and all connection configurations were found to be successful. Each device connected to the controller hub was tested for responsiveness, proper function, and data usage. Commands sent from the web client to the controller were found to arrive and activate them in under a second for all devices. The camera, water sensor, and RF transmitter were all found to work completely as expected, but the motion sensor and sound sensor had some issues. The sensitivity of the sound sensor wasn’t able to be set to a sufficient amount where moderate levels of noise could be detected. The only sound the sensor consistently detected was air blown into the microphone, suggesting a very low sensitivity. The motion sensor had the opposite issue, where many times it would detect motion where there was none. Sometimes the sensitivity of the motion sensor worked properly, but it wasn’t consistent.

9 Accomplishments and Issues

Over the course of the Fall semester, most of the design specifications were implemented successfully. A variety of Smart Devices are connected to a central control system that allows the Smart Home to be controlled remotely using a web client. A water sensor, sound sensor, motion sensor, camera, and RF transmitter are successfully implemented as Smart Devices, with the ability to transmit relevant data online. The server has a successful implementation of user verification that makes use of a username and password to limit unauthorized connections to the system. The web client has a user-friendly interface implemented in HTML/Javascript that can obtain information and control the Smart Home’s devices. The Smart Home system is designed in such a way that it allows simple implementation of new devices into the system. A system for a battery backup
Some parts of the design had to be altered due to either time or physical restraints. There was not enough time to properly implement the web client within an Android application, so the platform was switched to HTML/Javascript, which allows usage of the client on many more systems and is easier to implement quickly. Full encryption of data using TLS was not possible to implement with the time allotted, so a simpler user verification system was created instead. 4G hotspot capability was also not implemented due to time restraints. Use of a 3D printed casing for the Smart Devices was found to be unfeasible due to the lack of required accuracy of the available 3D printer. Mountings for the devices could not be created with the degree of precision required. The solution to this problem was to craft the casing by hand using wood carpentry, which allowed for the mountings to be properly adjust as it was produced.

10 Conclusion

Overall, the implementation of the Smart Home system is a success. This senior design project has been a valuable learning experience over the past year, providing insight into design philosophies and the Internet of Things. Information learned from this project will be useful moving forward into a post-grad setting, and will lead to better engineering in the future.

11 References


12 Appendix A - List of Figures

List of Figures

1 System Boundary Diagram ........................................... 5
2 Top Level Decomposition ........................................... 8
3 Risk Mitigation Diagram ........................................... 14
4 IoT Architecture ..................................................... 20
5 Timeline for Spring 17 Semester .................................. 23
6 Timeline for Fall 17 Semester ....................................... 24
7 Final Hardware Enclosure .......................................... 25
8 Wire Diagram of relay .............................................. 26
9 User UI 1 ............................................................. 28
10 User UI 2 ........................................................... 28
11 User UI 3 ........................................................... 29
12 User UI 4 ........................................................... 29
13 User UI 5 ........................................................... 30
14 User UI 6 ........................................................... 30

13 Appendix B - List of Tables

List of Tables

1 Probability of failure Table ......................................... 11
2 Detection of Failure Table .......................................... 12
3 Severity of failure Table ........................................... 13
4 Design Failure Mode and Effect Analysis (DFMEA) Table .... 15
5 Bill of Materials for Project ........................................ 22
14 Appendix C - Source Code

14.1 Hardware Code

14.1.1 Master Device Control

```python
import RPi.GPIO as GPIO
import os
import paho.mqtt.client as mqtt
import Sound_Detect as sound
import base64
import Water_Detect as water
import Camera_on_Motion as motionCam
import time
import math
import random, string
import json
from datetime import datetime
import PCF8591 as ADC
import tempSensor as temp

packet_size = 3000
lightState = False
lightSet = False

def randomword(length):  # Generate random string of given length
    return ' '.join(random.choice(string.lowercase) for i in range(length))

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " +str(rc))
    client.subscribe("ctrl")

def on_message(client, userdata, msg):
    global flag
    global lightSet
    message = str(msg.payload)
    print(msg.topic+" "+message)
```
if message == "motionVID": # Activate devices and functions when certain messages received
    flag[1] = True
elif message == "motionPIC":
    flag[0] = True
elif message == "motionOn":
    print "Activate Motion"
    flag[2] = True
elif message == "motionOff":
    print "Deactivated Motion"
    flag[2] = False
elif message == "soundOn":
    print "Activated"
    flag[3] = True
elif message == "soundOff":
    print "Deactivated"
    flag[3] = False
elif message == "lightOn":
    print "Light On"
    flag[4] = True
    lightSet = True
elif message == "lightOff":
    print "Light Off"
    flag[4] = True
    lightSet = False
elif message == "picture":
    print "Taking Picture"
    flag[5] = True
elif message == "video":
    print "Taking Video"
    flag[6] = True
elif message == "waterON":
    print "Water Detector Activated"
    flag[7] = True
elif message == "waterOFF":
    print "Water Detector Deactivated"
    flag[7] = False
elif message == "temperature":
flag[8] = True

def on_publish(client, userdata, mid):
    print("Publish Successful")

def convertImageToBase64(fileName):
    # Convert file to base64 for sending
    with open(fileName, "rb") as img:
        encoded = base64.b64encode(img.read())
    return encoded

def publishEncodedImage(encoded, fileType):
    # Fragment and publish file
    end = packet_size
    start = 0
    length = len(encoded)
    picId = randomword(8)
    pos = 0
    no_of_packets = math.ceil(length/packet_size)
    while start <= len(encoded):
        data = {"data": encoded[start:end], "pic_id": picId, "pos": pos, "size": no_of_packets}
        client.publish("data", json.JSONEncoder().encode(data))
        end += packet_size
        start += packet_size
        pos += packet_size

loopDelay = 0
GPIO.setmode(GPIO.BCM)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_publish = on_publish
client.username_pw_set("user2", "test")  # User authentication
client.connect("192.168.43.105", 1883, 120)  # Server IP

GPIO.setup(26, GPIO.IN)
ADC.setup(0x48)
water.setup()

powerStatus = 0;
#Flags set to True when devices is activated, each flag associated with one function
flag = [False,False,False,False,False,False,False,False,False]

client.loop_start()

while True:
    power = GPIO.input(26)
    if (power == 0) and (power != powerStatus): #Detect and publish power outage info
        powerStatus = power
        client.publish("alert","powerOut")
        print "Battery Power"
        #Power switch to on battery
    elif (power != 0) and (power != powerStatus):
        powerStatus = power
        client.publish("alert","powerOn")
        print "Wall Power"
        #Power switch to on wall

    if flag[0] == True: #Capture picture on motion
        motionCam.motionPic()
        if os.path.isfile('img1.jpg'):
            encoded = convertImageToBase64('img1.jpg')
            publishEncodedImage(encoded, "jpg")
        flag[0] = False

    if flag[1] == True: #Capture video on motion
        motionCam.motionVid()
        if os.path.isfile('vid1.h264'):
            os.system("sudo MP4Box -add vid1.h264 vid1.mp4")
            encoded = convertImageToBase64('vid1.mp4')
            publishEncodedImage(encoded, "mp4")
            filename = datetime.now().strftime("%Y-%m-%d_%H.%M.%s.mp4")
            os.rename('vid1.mp4', filename)
            os.remove('vid1.h264')
        flag[1] = False

detection = motionCam.motion()
if detection == True:
    client.publish("alert", "motion")
    flag[2] = False
if flag[3] == True:
    detection = sound.loop()
if detection == True:
    client.publish("alert", "sound")
    flag[3] = False
if flag[4] == True:
    if (lightSet == True) and (lightState == False):
        os.system("sudo ./433Utils/RPi_utils/codesend 4216115 -1 200")
        lightState = True
        client.publish("alert", "lightOn")
    elif (lightSet == False) and (lightState == True):
        os.system("sudo ./433Utils/RPi_utils/codesend 4216124 -1 200")
        lightState = False
        client.publish("alert", "lightOff")
    flag[4] = False
if flag[5] == True:
    motionCam.picture()
    encoded = convertImageToBase64("img1.jpg")
    publishEncodedImage(encoded, "jpg")
    flag[5] = False
if flag[6] == True:
    motionCam.testVideo()
    os.system("sudo MP4Box -add vid1.h264 vid1.mp4")
    encoded = convertImageToBase64("vid1.mp4")
    publishEncodedImage(encoded, "mp4")
    filename = datetime.now().strftime("%Y-%m-%d_%H.%M.%s.mp4")
    os.rename('vid1.mp4', filename)
    os.remove('vid1.h264')
    flag[6] = False
if flag[7] == True:
    detection = water.run()
    if detection == False:
        client.publish("alert", "water")
    flag[7] = False
if flag[8] == True:
    # Detect temperature and humidity
    humidity, temperature = temp.run()
    toSend = {'temp': (temperature * 9/5) + 32, 'hum': humidity}
    client.publish('temp', json.JSONEncoder().encode(toSend))
    flag[8] = False
    time.sleep(loopDelay)

14.1.2 Camera on Motion

from gpiozero import MotionSensor
from picamera import PiCamera
from datetime import datetime
import time

camera = PiCamera()
pir = MotionSensor(22)
def motionPic():
    pir.wait_for_motion()
    filename = 'img1.jpg'
    camera.capture(filename, resize=(500, 281))
    print 'Picture Taken'

def motionVid():
    pir.wait_for_motion()
    filename = 'vid1.h264' #datetime.now().strftime("%Y-%m-%d_%H.%M.%s.h264")
    camera.start_recording(filename)
    print 'Started Recording'
    #pir.wait_for_no_motion()
    time.sleep(5)
    camera.stop_recording()
    print 'Ended Recording'

def testVideo():
    filename = 'vid1.h264'
    camera.start_recording(filename)
    print 'Started Recording'
    time.sleep(5)
    camera.stop_recording()
print "Ended Recording"

def picture():
    #camera.rotation = 180
    camera.capture('img1.jpg', resize=(500,281))

def motion():
    global pir
    limit = 0
    #print("actually running")
    while limit < 10:
        if pir.motion_detected:
            print("Motion detected!

...\n
")
            return True
        limit = limit + 1
    return False

14.1.3 Motion

from gpiozero import MotionSensor

def setup():
    global pir
    pir = MotionSensor(22)

def run():
    global pir
    limit = 0
    #print("actually running")
    while limit < 10:
        if pir.motion_detected:
            print("Motion detected!

...\n
")
            return True
        limit = limit + 1
    return False
14.1.4 Sound Detect

#!/usr/bin/env python
import PCF8591 as ADC
import RPi.GPIO as GPIO
#import paho.mqtt.client as mqtt
import time

#GPIO.setmode(GPIO.BCM)
#GPIO.setup(19, GPIO.IN)

def setup():
    #ADC.setup(0x48)

def loop():
    #count = 0
    limit = 0
    while limit < 5:
        voiceValue = ADC.read(0)
        if voiceValue:
            print ('Value:', voiceValue)
            if (voiceValue > 79) and (voiceValue != 139):
                print ("Voice detected!")
                return True
                #count += 1
                limit += 1
                #time.sleep(0.2)
            return False
                #count = 0
                #limit = 0
                #if __name__ == '__main__':
try:
    setup()
    loop()
except KeyboardInterrupt:
    pass

14.1.5 Temp Sensor

```python
import Adafruit_DHT as DHT
import RPi.GPIO as GPIO
#import JSON

GPIO.setmode(GPIO.OUT)

def run():  
    humidity, temperature = DHT.read_retry(11, 21)  
    print humidity  
    print "humidity"  
    print temperature  
    print "temperature"  
    return humidity, temperature
```

14.1.6 Water Detect

```python
#import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import math

DO = 6
GPIO.setmode(GPIO.BCM)

def setup():
    #ADC.setup(0x48)
    GPIO.setup(DO, GPIO.IN)

def Print(x):
    #if x == 1:
```
```python
def run():
    status = 1
    limit = 0
    while True:
        #print ADC.read(2)

        tmp = GPIO.input(DO);
        if tmp != status:
            #print(tmp)
            status = tmp
        return tmp

        #time.sleep(0.2)

if __name__ == '__main__':
    try:
        setup()
        loop()
    except KeyboardInterrupt:
        pass
```

14.1.7 PCF8591

#------------------------------------------------------
#This is a program for PCF8591 Module.
#
# Warning! The Analog input MUST NOT be over 3.3V!
#
# In this script, we use a potentiometer for analog input, and a LED on AO for analog output.
#
# you can import this script to another by:
# import PCF8591 as ADC
#
# ADC.Setup(Address) # Check it by sudo i2cdetect -y -1
# ADC.read(channal) # Channal range from 0 to 3
# ADC.write(Value) # Value range from 0 to 255
#
#------------------------------------------------------

import smbus
import time

# for RPI version 1, use "bus = smbus.SMBus(0)"
bus = smbus.SMBus(1)

# check your PCF8591 address by type in 'sudo i2cdetect -y -1' in terminal.
def setup(Addr):
    global address
    address = Addr

def read(chn):  # channel
    try:
        if chn == 0:
            bus.write_byte(address, 0x40)
        if chn == 1:
            bus.write_byte(address, 0x41)
        if chn == 2:
            bus.write_byte(address, 0x42)
        if chn == 3:
            bus.write_byte(address, 0x43)
        bus.read_byte(address)  # dummy read to start conversion
    except Exception as e:
        print ("Address: %s" % address)
        print (e)

#------------------------------------------------------

44
```
~~Ireturn bus.read_byte(address)

def write(val):
~~Itry:
~~I~~Itemp = val  # move string value to temp
~~I~~Itemp = int(temp)  # change string to integer
~~I~~I# print temp to see on terminal else comment out
~~I~~Ibus.write_byte_data(address, 0x40, temp)
~~Iexcept Exception as e:
~~I~~Iprint("Error: Device address: 0x%2X ", address)
~~I~~Iprint(e)

if __name__ == "__main__":
~~Isetup(0x48)
~~Iwhile True:
~~I~~Iprint('AIN0 = ', read(0))
~~I~~Iprint('AIN1 = ', read(1))
~~I~~Itmp = read(0)
~~I~~Itmp = tmp*(255-125)/255+125  # LED won't light up below 125, so convert '0-255' to '125-255'
~~I~~Iwrite(tmp)
~~I~~Itime.sleep(0.3)

14.2 Mosquitto Configuration

14.2.1 mosquitto.conf

# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

listener 1883
```
listener 9001
protocol websockets

allow_anonymous false
password_file /etc/mosquitto/passwd

#acl_file /etc/mosquitto/acc_ctrl

#cafile /etc/mosquitto/certs/ca.crt
#certfile /etc/mosquitto/certs/server.crt
#keyfile /etc/mosquitto/certs/server.key

#require_certificate true

#use_identity_as_username false

log_dest topic

log_type error
log_type warning
log_type notice
log_type information

connection_messages true
log_timestamp true

include_dir /etc/mosquitto/conf.d

14.2.2 password

user1:UgbFyZAH6BbTOuES$qqLrZ0S+puE2XKYFZzJLqExVD+tLFBAvd9kLVxtfxr09H5ehWe55m
user2:rxGyjsWUKbUioGDCsLXmbII5e7L5CcJ+NcEZxQ8451GlgyfVb0ecdPQ8FF80EF7b00U10u0k
user3:n1hsz/jX0INSR4ZASrVS7Q6yPTDknxR/ZitM4NOn1ZIN0knnElZowTYTHmhZ04jwc2g5zJq
14.3 Web Client

```html
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Mosquitto Websockets</title>
  </head>
  <script type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
  <script type="text/javascript">
    var mqtt;
    //var host="192.168.1.184";
    //var port=1883
    //var host="broker.mqttdashboard.com";
    var host = "192.168.43.105";
    //var port=8000
    var port=9001
    var reconnectTimeout = 2000;
    var pictures = {"0": null}
    /*
      //connection attempt timeout in seconds
      timeout: 3,

      //Gets Called if the connection has successfully been established
      onSuccess: onConnect,

      //Gets Called if the connection could not be established
      onFailure: function (message) {
        alert("Connection failed: " + message.errorMessage);
      }
    */

    function MQTTconnect() {
      console.log("in connect");
      mqtt = new Paho.MQTT.Client(host,port,"clientjs");
```
Ivar options = {

//connection attempt timeout in seconds
timeout: 3,

//Gets Called if the connection has successfully been established
onSuccess: onConnect,

//Gets Called if the connection could not be established
onFailure: function (message) {
    alert("Connection failed: " + message.errorMessage);
},

userName: "user1",
password: "test"

};

mqtt.onConnectionLost = onConnectionLost;
mqtt.onMessageArrived = onMessageArrived;

mqtt.onFailure = onFailure

//console.log("Host="+ host + ", port=" + port + ", path=" + path + " TLS = " + useTLS + " username=" + username + " password=" + password);

console.log("connecting")

mqtt.connect({onSuccess:onConnect});

mqtt.connect(options)

mqtt.connect({onSuccess: onConnect});

}

function onFailure(message) {
    console.log("Failed");
    $('#status').val("Connection failed: " + message.errorMessage + " Reconnecting in " + reconnectTimeout + " seconds.");
    setTimeout(MQTTconnect, reconnectTimeout);
}

function onConnect() {
    // Once a connection has been made, make a subscription and send a message.
    console.log("on Connect");
    mqtt.subscribe("ctrl");
    mqtt.subscribe("alert");
}
```
function onConnectionLost(responseObject) {
    console.log("Connection Lost");
    setTimeout(MQTTconnect, reconnectTimeout);
    $('#status').val("connection lost: " + responseObject.errorMessage + " Reconnecting");
}

function onMessageArrived(message) {
    var topic = message.destinationName;
    var payload = message.payloadString;
    if(topic == "data")
        reconstructBase64String(message.payloadString);
    if(topic == "ctrl")
        if(payload == "Hello"){
            $('#display').html($('display').html() + "<br>Hello Server!<br>");
        }
    if(topic == "alert")
        if(payload == "water"){
            $('#displayWater').html($('displayWater').html() + "Water Leakage Detected at " + Date());
        }
    if(payload == "sound"){
        $('#displaySound').html($('displaySound').html() + "Sound Detected at 
```
```javascript
if (payload == "motion") {
  $('#displayMotion').html($$('#displayMotion').html() + 'Motion Detected at ' + Date);  
}

if (payload == "lightOn") {
  $('#displayLight').html('Light is On<br>');  
}

if (payload == "lightOff") {
  $('#displayLight').html('Light is Off<br>');  
}

if (payload == "powerOut") {
  $('#display').html($$('#display').html() + '<br>Device power lost: switching to battery power at ' + Date);  
}

if (payload == "powerOn") {
  $('#display').html($$('#display').html() + '<br>Device power restored: switching to outlet power at ' + Date);  
}

if (topic == "temp") {
  info = JSON.parse(message.payloadString);
  temperature = info["temp"];  
  humidity = info["hum"];  
  $('#displayTemp').html($$('#displayTemp').html() + '<br>Temperature ' + temperature + ', humidity ' + humidity + '% at ' + Date).html();  
}

$(document).ready(function() {console.log("connecting to" + host);  
//$('#display').html("connecting to" + host);  
$('#display').html($$('#display').html() + '<br>connecting to ' + host);  
MQTTconnect();});
```
function lightOn() {
    alert('light is on');
    message = new Paho.MQTT.Message("lightOn");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function lightOff() {
    alert('light is off');
    message = new Paho.MQTT.Message("lightOff");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function gettemp() {
    alert('Getting Environment Info');
    message = new Paho.MQTT.Message("temperature");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function takephoto() {
    alert('Taking a Picture');
    message = new Paho.MQTT.Message("pic");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function soundOn() {
    alert('Turning on Sound Sensor');
    message = new Paho.MQTT.Message("soundOn");
```javascript
// function soundOff() {
  alert('Turning off Sound Sensor');
  message = new Paho.MQTT.Message("soundOff");
  message.destinationName = "ctrl";
  mqtt.send(message);
}

// function motionOn() {
  alert('Turning on Motion Sensor');
  message = new Paho.MQTT.Message("motionOn");
  message.destinationName = "ctrl";
  mqtt.send(message);
}

// function motionOff() {
  alert('Turning off Motion Sensor');
  message = new Paho.MQTT.Message("motionOff");
  message.destinationName = "ctrl";
  mqtt.send(message);
}

// function picture() {
  alert('Taking a picture');
  message = new Paho.MQTT.Message("picture");
  message.destinationName = "ctrl";
  mqtt.send(message);
}

// function video() {
  alert('Taking a video');
```
message = new Paho.MQTT.Message("video");
message.destinationName = "ctrl";
mqtt.send(message);
}

function waterON() {
    alert('Water detection ON');

    message = new Paho.MQTT.Message("waterON");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function waterOFF() {
    alert('Water detection OFF');

    message = new Paho.MQTT.Message("waterOFF");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function gasON() {
    alert('Gas detection ON');

    message = new Paho.MQTT.Message("gasON");
    message.destinationName = "ctrl";
    mqtt.send(message);
}

function gasOFF() {
    alert('Gas detection OFF');

    message = new Paho.MQTT.Message("gasOFF");
    message.destinationName = "ctrl";
    mqtt.send(message);
}
function motionPIC() {
    alert('Picture will be taken if motion is detected');
}

message = new Paho.MQTT.Message("motionPIC");
message.destinationName = "ctrl";
mqtt.send(message);
}

function motionVID() {
    alert('Video will be recorded if motion is detected');
}

message = new Paho.MQTT.Message("motionVID");
message.destinationName = "ctrl";
mqtt.send(message);
}

function reconstructBase64String(chunk) {
    pChunk = JSON.parse(chunk); //"d"

    // creates a new picture object if receiving a new picture, else adds incoming strings to an existing picture
    if (pictures["0"] == null) {
        pictures["0"] = {"count":0, "total":pChunk["size"], pieces: {}, "pic_id": pChunk["pic_id"]};
        pictures["0"].pieces[pChunk["pos"]]] = pChunk["data"];
    }
    else {
        pictures["0"].pieces[pChunk["pos"]]] = pChunk["data"];
        pictures["0"].count += 1;
    }

    if (pictures["0"].count == pictures["0"].total) {
        console.log("File reception complete");
        var str_image="";
    }
for (var i = 0; i <= pictures["0"].total; i++)
    str_image = str_image + pictures["0"].pieces[i];

// displays image
^^I^^Iif(pictures["0"].format == "jpg"){
    var source = 'data:image/jpeg;base64,'+str_image;
    var myImageElement = document.getElementById("picture_to_show");}
^^I^^Ielse if(pictures["0"].format == "mp4"){
    ^^^Ivar source = 'data:video/mp4;base64,'+str_image;
    ^^^Ivar myImageElement = document.getElementById("vidTest");
    myImageElement.src = source;

^^I^^Ipictures["0"] = null;
}
^~font-size: 60px;
   text-shadow: -3px 0 blue, 0 3px blue, 3px 0 blue, 0 -3px blue;
^~text-align: center;

h3{
color: black;
border-radius: 25px;
border: 2px solid #73AD21;
padding: 20px;
width: auto;
height: auto;
background-color: white;
}

.button {
   display: inline-block;
padding: 15px 25px;
font-size: 24px;
cursor: pointer;
text-align: center;
text-decoration: none;
outline: none;
color: #fff;
background-color: #4CAF50;
border: none;
border-radius: 15px;
boux-shadow: 0 9px #999;
width: 100%;
}

.button:hover {background-color: #3e8e41}

.button:active {
   background-color: #3e8e41;
   box-shadow: 0 5px #666;
   transform: translateY(4px);
}
<h1><img src="https://eclubprague.com/wp-content/uploads/2016/03/iot.png" alt="IOT" width="450" height="450"></h1>

<h3>Starting again</h3>

<div class="functions">

<h1><img src="http://smartelectricians.co.uk/wp-content/uploads/2016/06/Electrician-Sheffield-1.png" alt="light" width="300" height="250"></h1>

<h2>Turn light ON and OFF</h2>

<button onclick="lightOn()" class="button" style="vertical-align:middle"><span>Light ON</span></button>

<button onclick="lightOff()" class="button" style="vertical-align:middle"><span>Light OFF</span></button>

<h1><img src="http://icons.iconarchive.com/icons/igh0zt/ios7-style-metro-ui/512/MetroUI-Other-Sound-icon.png" alt="Temperature" width="250" height="250"></h1>

<h2>Get temperature and humidity of the house</h2>

<button onclick="gettemp()" class="button" style="vertical-align:middle"><span>Get temperature and humidity</span></button>

<h1><img src="http://neo-farm.net/neofarmENG/images/icons/water.png" alt="Temperature" width="250" height="250"></h1>

<h2>Turn sound detection ON and OFF</h2>

<button onclick="soundOn()" class="button" style="vertical-align:middle"><span>Sound detection ON</span></button>

<button onclick="soundOff()" class="button" style="vertical-align:middle"><span>Sound detection OFF</span></button>

</div>
Turn motion detector ON and OFF

Take a picture or record a video with or without motion detection

Turn water leakage detection ON and OFF
<button onclick=waterON() class="button" style="vertical-align:middle"><span>Water detection ON</span></button>
<p><button onclick=waterOFF() class="button" style="vertical-align:middle"><span>Water detection OFF</span></button></p>
<h3 id="displayWater"/>

<!-- <h1>Detect any gas leakage</h1> -->
<!-- <h2>Publishing from web to get gas leakage info</h2> -->

<!-- <button onclick=gasON() class="button" style="vertical-align:middle"><span>Gas detection ON</span></button> -->
<!-- <p> -->
<!-- <button onclick=gasOFF() class="button" style="vertical-align:middle"><span>Gas detection OFF</span></button> -->
<!-- <p> -->

</div>

</body>
</html>