Z-Wave Protocol Based Home Automation and Security System

By

Roberto Unzaga

Advisor: Dr. Afsaneh Minaie

Utah Valley University

Computer Engineering Department

Spring 2018

Table of Contents

Abstract …………………………………………………………………………………….….. 5

Introduction ………………………………………………………………………..……….….. 6

Theory/Discussion of Simulation and Design Methods ………………………………………. 11

Code Explanation ……………………………………………………………………………... 49

Description of Design …………………………………………………………………………. 65

Measurement Methods ……………………………….…………………………………….…. 68

Project Constraints ……………………………………………………………………………. 69

Problems Encountered/What was Learned ………….……………………………..…………. 69

Student Experience …………………………………..……………………………………….. 71

Conclusions/Recommendations ……………………..………………………………………... 71

Future Work …………………………………………………………………………………... 72

References …………………………………………………..……………….………………... 73

Appendices

Appendix A: Original proposal ………………………………………..………….………....... 79

Appendix B: Source Code ……………………………………………..…………………….... 84

Appendix C: Slide Show ………………………………………………………..…………….. 103

Table of Figures ………………………………………..…………………………………….… 3

Z-Wave’s Ideal Home …………………………………………………...………………….….. 7

Z-Wave Frame Example …………………………………………………………………...…. 17

Collision Avoidance Zone ………………………………………………………………….…. 18

Basic Frame Format ……………………………………………………………………..….… 18

Repeating Process …………………………………………………………………………….. 19

Raspberry Pi 3 Model B …………………………………………………………………...….. 21

Home Assistant Core Architecture …………………………………………………………… 25

Z-Wave Z-Stick ……………………………………………………………………………….. 29

Z-Wave Chip …………………………………………………………………………………. 29

Contact Sensor ………………………………………………………………………………... 30

Normally Open Reed Switch …………………………………………………………………. 31

Switch Closes When Magnet Is Near ………………………………………………………… 31

PIR Motion Detector and Temperature Sensor ……………………………………………….. 32

Smart Switch ………………………………………………………………………………….. 33

Kwikset Smartcode Door Lock ………………………………………………….……………. 34

Web Camera …………………………………………………………….…………………..… 35

808 CANZ Bluetooth Wireless Speaker ……………………………………...………………. 36

Door Prototype …………………………………………………………………….………….. 40

Window Prototype ……………………………………………………………….…………… 40

Layout Prototype ……………………………………………………………...………………. 41

Built Model …………………………………………………………………………………… 41

Command Tabs ……………………………………………………………………..………… 42

Final View of Home Assistant ……………………………………………………...………… 43

iPhone Push Notifications …………………………………………………………………….. 44

iPhone Home Assistant Settings ………………………………………………….…………… 45

Home Assistant Menu ………………………………………………………………..……….. 46

Door Lock, Camera, and Smart Switch ………………………………………....…………….. 47

Door with Contract Sensor ……………………………………...…………………………….. 47

Window with Contact Sensor and Motion Sensor …………………………………….……… 48

Speaker, Z-Stick, and Raspberry Pi ……………………………………………………...…… 48

Schematic of Wireless Devices ……………………………………………………………..… 65

Schematic of Wired Devices …………………………………………………..……………… 66

Circuit Design of Raspberry Pi ……………………………………….………………………. 67

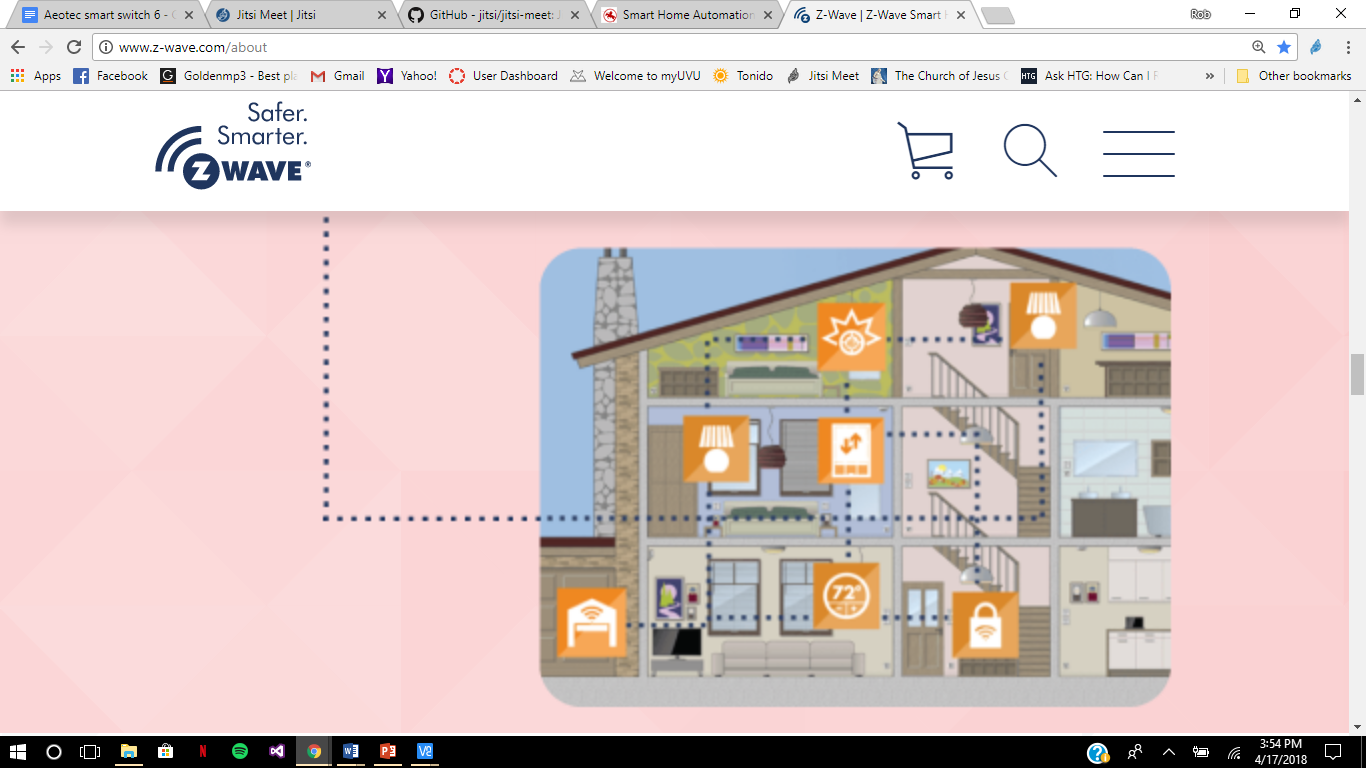
Cost of Project ………………………………………………………………………………… 68

**Abstract**

Home Security has been one of the highest priorities to obtain for some time. This is due to the fact that fewer homes with Home Security are likely to be broken into and it gives people a feeling of safety. Home Automation has been a growing industry in recent years with the Internet of Things and Wireless Sensor Networks becoming popular. These networks of Home Security use devices that run on Bluetooth, ZigBee, Wi-Fi, and Z-Wave protocols. Home Security systems have been more convenient for users to control and monitor devices in the home. This is because the use of mobile applications have grabbed the interest of users everywhere. The paper presents a system implementing the connection of Wireless sensors to a Raspberry PI 3 using the z-wave protocol and it is controlled from a mobile application. The user will be able to monitor the door, window, door lock, motion detector, and camera in the home and from the application using the Home Assistant program. As a result this Home Security and Automation system will help provide the personal self satisfaction, ease of usage and a sense of safety.

**Introduction**

Home Security has been a growing industry and is getting more popular in the residential areas in the world. This is widely because a home without security is three times more likely to be broken into compared to those with state-of-the-art security systems and in 2007 there were four million household break-ins in the United States, of which 500,000 resulted in bodily injuries and 20,000 resulted in homicides [1]. [2] also comments on how monitoring can only help so much. When someone breaks into a home the authorities are informed but it can take a while for them to arrive. During this time there can be a loss of lives and injuries could happen. Having a Home Security system is very crucial and can mean life or death. Creating a system can be hard and/or expensive depending on experience and the companies that have the products wanted. In the past, home security has, for the most part, been wired and have cost a lot to have someone install because that required rerouting of power to where it was needed for sensors, lights, and cameras. In today’s world, there is a lot of wireless access using Bluetooth, ZigBee, Z-Wave, Wi-Fi, and other protocols. Using multiple devices with these protocols would be considered a network. A Wireless Sensor Network seems to be the best way to put a home security and automation system together with the desired sensors and autonomous devices connected. Figure 1 shows a desired system with sensors and automation devices that are implemented in a home.

Figure 1: Z-Wave’s Ideal Home [3]

Machine to machine communication (M2M) is when devices, such as sensors, retrieve analog data and sends it to a backend server to be used. The data can be temperature, inventory level, motion, light detection and so on [4]. The growth of these machines has been fast and the projected growth is between 20 and 50 billion devices connected globally to the internet by 2020 [4]. It is said that “devices and systems can communicate with each other and can be controlled automatically to interact with the household members and improve the quality of their life” [4]. WSNs have grown to be M2Ms because most of the sensors now have microcontrollers to communicate with other devices.

**Wireless Sensor Networks**

Wireless Sensor Networks (WSN) are used for monitoring the status of patients in hospitals, control of video and remote sensor security systems, and of building gates and doors. They reduce the running costs in modern factories, shops, offices, and residences. Smart buildings can save costs by automatically adjusting heating levels when people leave a room [5].

[6] defines a WSN as a large-scale, wireless, ad hoc, multihop, unpartitioned network of homogeneous, tiny, mostly immobile sensor nodes that would be randomly deployed in the area of interest. Each WSN can be completely different depending on the specific needs that are required. For example, deployment might be random or installed at deliberately chosen spots; deployment might be a continuous process or might be a one-time event. WSN might need to have mobile sensors meaning they can move, be placed on moving objects, or might be an incidental side effect. The reason to choose a WSN is because of the cost, size, and energy of the whole system. The system intended on building doesn’t require the nodes to be the size of a shoebox and doesn’t require a lot of energy. A WSN can take batteries and often doesn’t require a user to change out the batteries for years at a time. The cost of a single device also varies from hundreds of dollars to a few cents. For a home security, the sensors cost a medium amount [6].

WSNs communicate through a variety of ways such as radio, light, laser, inductive and capacitive coupling, and even sound. Cost, surroundings, and purpose play a key role in which way a WSN communicates. The most common way to communicate is through radio because there doesn’t need to be a line of sight, is implemented with low power consumption and uses small antennas. Light beams require a line of sight and may interfere with ambient light and daylight [6].

A WSN can have different topologies in which they communicate. The two that are the most familiar and most common are the star network and a mesh network. The star network consists of sensor nodes sending their data to a central location. The mesh network has all the sensor nodes capable of receiving data and forwarding that data to another sensor node until it reaches the place it is meant [6].

Implementing a WSN into a home security and automation system can be risky because the users are mostly everyday people who don’t have backgrounds in coding and setting up systems like this. There is a growing interest in users of smart homes but sometimes they have little to no interest in understanding everything about the smart home products they use. It takes time to learn about a system and a lot of people aren’t willing to invest time to learn. If one person learns it they’re most likely to be the only one in the household [7]. Some worry that by automating the lights it reduces opportunities for parents to teach their children how not to be wasteful [7].

A WSN node generally has four parts to it: data collecting unit, data processing unit, wireless communication unit and power management [2]. There is a good amount of security threats that can come into a Wireless Sensor Network. [4] gives a list of attacks:

1. Eavesdropping: An outside attack where an attacker can listen in and overhear and take the information.
2. Denial of service: DoS is when a PC is used to meddle with the frequencies being used by the network in the WSN.
3. Node Compromise: When a node in the network is taken and compromised or reprogrammed.
4. Sinkhole and Wormhole Attacks: Sinkhole attack is where there is a fake node that takes the information and creates fake routes for the data to go in and results in the information coming back to that node. Wormhole is when a fake node in the network acts like a regular node but replays the information it has received and gives false information.
5. Physical Attack: Attacker gains physical access to the device in the network.

**Advanced Encryption Standard**

The Advanced Encryption Standard (AES) protects against most of these attacks except for DoS, which all machines have. To avoid this specific threat, the protocol usually has a few different channels to operate in and can randomly hop from one node to another. The AES replaced the widely used Data Encryption Standard (DES) but then the National Institute of Standards and Technology (NIST) decided it needed to be replaced because the key size needed to be enlarged. An encryption needs a key to change the original message and uses the same key to decrypt the message to get the original. The key size of the DES was small enough that it was easier to try all the different keys to crack the encryption instead of cracking the algorithm.

In 1998, the NIST decided there needed to be a change and opened it to the world with certain specifications: would be unclassified and publicly disclosed, offered key sizes of 128, 192, and 256 bits, had a 128-bit block size, and would be available royalty free anywhere in the world. Fifteen candidates started and by the end there were five algorithms they could choose from. There was speculation that the NIST would be biased to selecting the algorithm by IBM because they created the DES. To the surprise of everyone involved the Rijndael, created by Joan Daemen and Vincent Rijmen, won and was adopted as the Advanced Encryption Standard. With this new encryption standard some have suggested that it would survive for several centuries [8].

There are four steps to encryption: substitute bytes, shiftrows, mixcolumns, and addroundkey. SubBytes Transformation is where a substitution table is used to change the value of the original message. Doing this makes it faster to compute because of the lack of complexity.

The goal of this project is to create a WSN implemented into a home security/automation system that an average person can setup on their own. This project consists of many elements that serve specific purposes and will be detailed, respectively, later.

**Theory/Discussion of Simulation and Design Methods**

There are two Protocols that have been considered for design: ZigBee and Z-Wave. These have been considered because they are the leading protocols for Home Security and Automation. The goals with these two protocols are to reduce running costs, reduce the installation costs of wires, and keep a low data rate. Vivint Smart Home uses Z-Wave for their smart home applications.

IEEE 802.15.4 is a standard that supports ZigBee, ISA100.11a, WirelessHART, MiWi, SNAP and Thread specifications [9]. ZigBee is one of the focuses of this paper and should be envisioned when speaking about IEEE 802.15.4. The main goal of IEEE 802.15.4 is to achieve extremely low manufacturing and operation costs and technological simplicity, without sacrificing flexibility or generality.

**ZigBee**

IEEE 802.15.4-conformant devices may use one of three possible frequency bands for operation, which are 868 MHz, 915 MHz and 2450 MHz. There are two node types in ZigBee: full-function device and reduced-function device. The full-function device can serve as the coordinator of a personal area network and function as a common node. It talks to any other device and may relay messages. A reduced-function device is meant to be an extremely simple device with very modest resource and communication requirements. They never act as coordinators and can only communicate with full-function devices [9]. ZigBee devices have a peer-to-peer (mesh) or a star topology. One attribute that sets ZigBee apart from the other protocols is that one can have as many as about 65,000 devices on the same network at the same time. This attribute is also a flaw because if one has 1000s of nodes there tends to be some technical limitations.

ZigBee uses the Advanced Encryption Standard (AES) for security. There are three types of keys used in ZigBee – master keys are used only for key exchange, network keys are shared among devices within network and used for network data encryption, and link keys shared between two nodes and encrypt communication between them [10].

ZigBee has a Green Power initiative where they are focusing their attention to the power that is required for their devices. According to them, batteries are the most toxic waste streams on Earth. Batteries can go out quickly at times and if one has 60 wireless devices and they have a battery that lasts five years, the average of the household is changing one battery per month. What they want to move towards are devices that use alternate forms of harvesting energy such as using light, heat, and much more. Ultra-low power silicon is another idea they have that runs on microamperes of energy that support the IEEE 802.15.4 protocol [11].

Another way ZigBee Green Power is helping is network integration. This improves the range of sending messages and when a sensor is down the network can heal itself; this is a mesh network. This helps because ZigBee doesn’t rely on internet connections for communication. ZigBee is protected against Distributed Denial of Service (DDoS) attacks because they don’t use the internet and are protected against some of the other more challenging internet security concerns. ZigBee Green Power devices would only need up to 500 uJ of energy to transmit data. That puts into perspective how much energy they are using as a system and points out their goal of low power consumption [11].

ZigBee Green Power is part of the ZigBee Alliance and is there to solve the power obstacles that burden and hinder the potential around the IoT. ZigBee Green Power is already being built into millions of smart devices so they don't require battery changes. To go along with the benefits of cost savings and energy, ZigBee Green Power can also be put anywhere, especially in places that are hard to wire. To do all of this, energy harvesting is required and an exposed solar cell is all it takes. Other things that help are smart bulbs, smart power outlets, window shades, appliances, etc. They forward data packets on behalf of the other devices in the network.

A design has already been made by students in an automation school that includes a key fob that arms the ZigBee system. The design includes a star topology, which means that all the sensors are only communicating to the microcontroller and not bouncing off each other. The microcontroller they used is the MSP430F135 and uses a transceiver to receive the signals from all the devices. When no interruptions occur, the sensors are asleep to keep from wasting battery life. This doesn’t mean they are off but that they are using as less battery life as possible just to detect because keeping everything else on would drain the battery for no reason. The authors to this design specify that “It is easy to install, upgrade, and network home monitoring and control system without wires.” [12]

Mentioned earlier was the ZigBee Alliance, which is a group that is over IEEE 802.15.4 and helps with the changes that occur in their designs. Another alliance out there is the Z-Wave Alliance that covers Z-Wave technology in the same way.

**Z-Wave**

The Z-Wave alliance was formed in 2005 by a group of home control product manufacturers. Founding members included Intermatic, Levition, Wayne Dalton, Danfoss, and Universal Electronics. The creator of Z-Wave technology was Zensys and more than 450 companies have joined the Z-Wave alliance since the start [13]. Z-Wave is said to be “The leading smart home technology found in millions of products around the world. [14]” Some of the most well-known smart home systems use Z-Wave products such as ADT Pulse, AT&T Digital Life, Lowe’s Iris, Vivint, Alarm.com, SmartThings, Nexia and more. There are over 375 manufacturers and more than 1700 Z-Wave smart products to choose from. It is a wireless technology that won’t interfered with Wi-Fi signal and has the goal to operate on low power. When Z-Wave is utilized inside a product, that product becomes “smart” giving them the ability to talk to each other and enable control, from a user, from anywhere. Z-Wave operates at 908.42 MHz in the US and operates at other frequencies in other countries. The disadvantage of the Z-Wave protocol is that there can only be up to 232 devices on a single network. This limitation is a good thing though because it ensures that there is fast action and no chance of slowing down the network; the more devices means the longer it takes to control [14].

Z-Wave devices can communicate with each other in a mesh network to get the data back to the smart hub as soon as possible; each Z-Wave device, in this case, acts as a repeater. There are two types of Z-Wave products on the market at this time and those are Z-Wave and Z-Wave plus. Z-Wave is the same as what we’ve been talking about but the Z-Wave plus has a higher level of security and compatibility between products. Z-Wave plus has also been optimized for easy set-up and installation. Z-Wave is fully interoperable but may require physically placing the smart hub near the product being added to the network [14]. Z-Wave plus has 50% improved battery life, 67% improved range, 250% more bandwidth, 3 RF channels for noise immunity and a higher bandwidth, and over the air firmware updates [15] [16].

A smart hub is really the brain of the home because it controls and sends commands to other connected devices. A Wi-Fi router is like a smart hub because it sends signals to laptops, cable boxes, Amazon Echo, etc. The smart hub sends a command via Z-Wave. It is recommended to have a Z-Wave device roughly every 30 feet or closer for maximum efficiency. The range is 328 feet (100 meters) but due to building materials the range drops significantly [14].

Z-Wave devices have the S2 security and was implemented in 2017. S2 divides the logical Z-Wave network into three dedicated security classes, which each have a unique network key. A given S2 security class not only identifies the network key to use but also dictates the rules applying to authentication of a new node during inclusion. The “S2 Access Control” class is the most trusted class, intended for access control devices like door locks and garage doors. The “S2 Authenticated” class is used for all normal household devices such as sensors and light dimmers. The “S2 Unauthenticated” class is the least trust class and is only intended for the most constrained controllers that, due to a limited user interface, are not capable of authenticating a node joining the network. An example is a key fob used to control a few lamps in a country cabin.

The S2 uses the AES-128 encryption to prevent issues with information being hacked and troubles with attacks. The US Government considers AES-128 safe enough for classified information up to the SECRET level [17]. S2 also includes Nonce scrambling, which means Number used Once, is arbitrary, and are often random or pseudo-random numbers issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks [18]. Together with the AES-128 and Nonce scrambling, there is no known way to break this protection – even with the aid of a super computer.

Z-Wave devices all have batteries that vary from 1 year to 9 years and bring up a similar problem of having to replace batteries in all the devices in your network that has been mentioned earlier. If one has 60 devices, in 5 years the average amount of batteries that need to be replaced is 1 every month.

**ZigBee Vs Z-Wave**

Z-Wave and ZigBee have a lot of features that are similar and competitive like battery life and energy efficiency. They both use AES encryption so they are both considered safe by the US Government and they both have keys to enter the network. A big difference in ZigBee and Z-Wave protocols is the number of nodes in a single network; ZigBee can have 65,000 devices and Z-Wave can have up to 232 devices. This can be a deciding factor depending on who is setting up their network. If a big company needs to set up devices in their building and need more than 232 devices, Z-Wave is not for them. ZigBee is more for customers of the commercial sort whereas Z-Wave is more geared towards residential customers. The other difference between these protocols is the price for devices; ZigBee is less expensive than Z-Wave. The big benefit of having Z-Wave is the frequency that it broadcasts at because it doesn’t interfere with Wi-Fi, Bluetooth, or ZigBee, which run around 2.4 GHz, whereas ZigBee runs at these frequencies and may be interrupted.

Due to the information provided above, the project will include the Z-Wave protocol implemented into the design. It’s important to decide which protocol to use and then to decide on which device it should run on because research must be done to see what can run that particular protocol. A device that can run this protocol and run other devices for this project is the Raspberry Pi. The Raspberry Pi 3 model B was chosen for the design over other devices because it can run the software for the Z-Wave sensors as well as manage the use of a camera at the same time.

A home automation and security is only important if one can see the results so using an application that displays the components connected is ideal. The Home Assistant has been chosen to be the software in this design. Reasons that lead to this are that it can implement the Z-Wave protocol and talk to other devices connected on the same network. It is very user friendly and customizable for whatever the user needs it to be. This project’s design will be explained in the next section and how it is to be implemented onto the Raspberry Pi and with what code is needed for everything to function.

**Design Components**

The design of this project consists of the Z-Wave protocol, a Raspberry Pi, Z-Wave sensors and devices, the Raspbian Operating System, Home Assistant, Duck DNS, a webcam, Jitsi Meet, and an iPhone. Each of these items will be discussed in this section and how they are used in the design.

**Z-Wave Protocol**

The Z-Wave protocol, as discussed earlier, is a protocol that enables sensors to communicate without interruptions from each other and other frequencies. It’s also been mentioned that the network can have up to 232 devices connected and needs a protocol to keep everything running smooth. The Z-Wave protocol architecture contains four layers: MAC Layer, Transfer Layer, Routing Layer, and Application layer [19].

The Media Access Layer (MAC) controls the radio frequency (RF) medium which is also controlled by hardware that is wireless and is independent of the RF medium. Figure 2 shows the

Figure 2: Z-Wave Frame Examples [20]

data stream, a sequence of digits encoded, data packets and consists of a preamble. The MAC layer in the Z-Wave protocol has the data split into “8-bits frame” which is forwarded throughout the network. The MAC layer also contains the collision avoidance (CA) technique, which lets a node talk to another node or the central hub when the channel is available and not being used by another node. If there is a collision, the transmission attempt is deferred for a random amount of time. The CA is accomplished by having each node that has no data to send to be set to receiving. Figure 3 shows a frame ending and another frame beginning making it collision avoidance [20].

Figure 3: Collision Avoidance Zone [20]

The transfer layer administers the connection between two devices including retransition, checksum screening and the acknowledgement (ACK), which shows if the connection was

Figure 4: Basic Frame Format [20]

successful or not. Figure 4 shows the Z-Wave basic frame format and each have 8 bits in each frame [20].

The routing layer manages the forward of the frames through the network. As mentioned above, every device can act as a repeater and happens when they are listening and have a steady venue all the time. The best path to the destination will be found using a routing table which contains information about every node connected to the network including which nodes can see other nodes [20]. There are two types of frames in the routing layer:

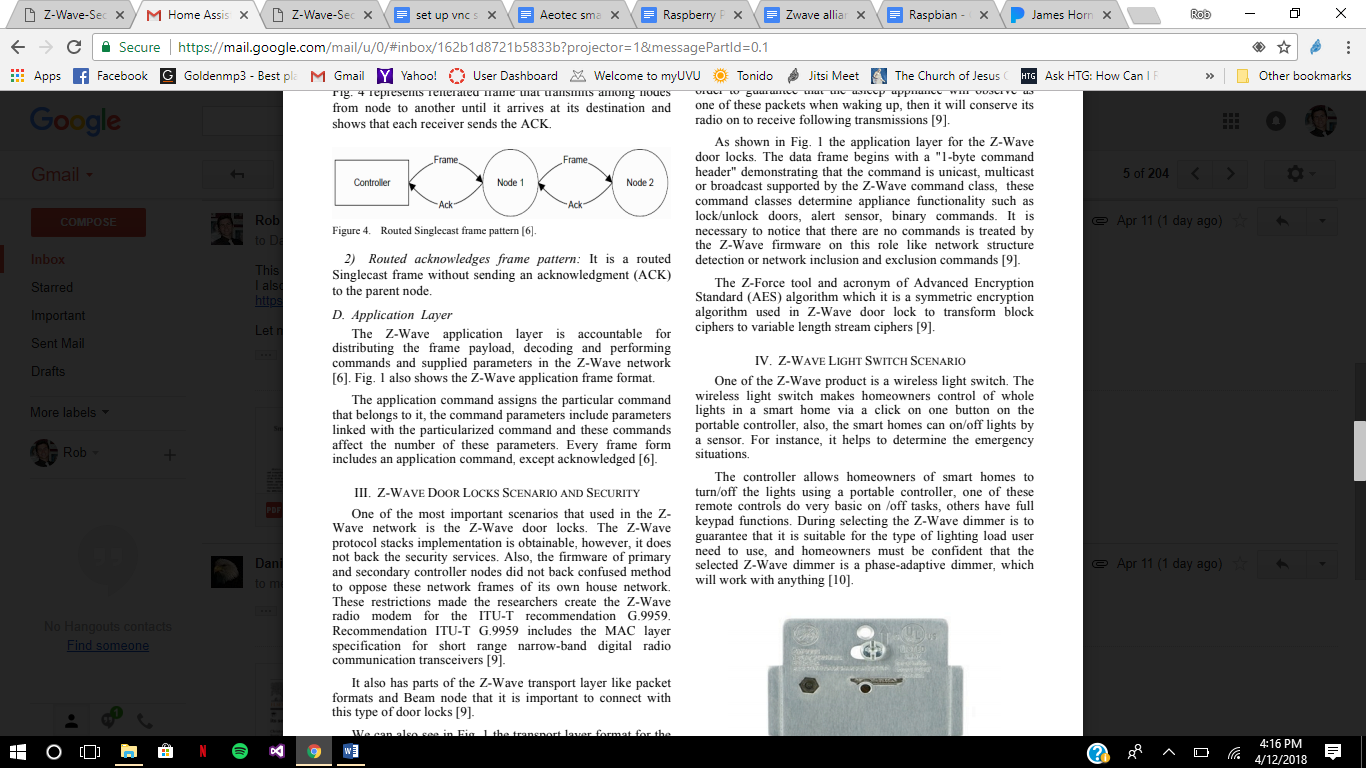
1. Routed Singlecast Frame Pattern: a single destination frame with an ACK that gets sent to the parent node. Used to repeat until it either gets to the destination or has 4 hops. Figure 5 shows a frame being transferred and an ACK being sent back.

Figure 5: Repeating Process [20]

1. Routed Acknowledges Frame Pattern: A routed Singlecast frame without sending an ACK to the parent node.

The application layer is responsible for distributing the frame payload, decoding and performing commands and supplied parameters in the Z-Wave network. Figure 4 shows the application frame format. The application command assigns the command to be done. The command parameters include parameters linked with a command. These commands affect the number of these parameters. Every frame form includes an application command, except ACK [20].

**Connecting Devices**

Linking up Z-Wave devices into a network isn’t hard but there is a risk of a foreign node connecting accidentally or due to evil intent. Depending on the User Interface, an including controller may allow the installer of the device to physically enter a Device-Specific Key (DSK) which is a string of digits that can be read visually or scanned as a QR code. The DSK is the first 16 byes of the 32-byte long ECDH public key of the joining node. ECDH is the Elliptic Curve Diffie Hellman and is used for the purposes of key agreement [21]. The QR code is abbreviated for Quick Response Code and is a two-dimensional barcode that can be read by machines [22]. For the most part, the Z-Wave won’t ask for this and will work as easy as Bluetooth. A user wishing to add a device to the network first puts the Z-Wave controller and the new device into a pairing mode. This is most likely achieved by pressing a button or physically resetting the device. Doing this will add any device that is also in pairing mode. Removing a node is done in a similar manner [23].

Each device is then given a 4-byte home ID during the pairing process is assigned by the controller, which is a device that has a list of all the devices on the network. All devices on the network share this home ID, which is assigned to the controller by the vendor in the factory. Another form of identification is the node ID which is a byte value assigned by the controller to a device during the pairing process. The controller itself always has a node ID of 1 and the second device will have a node ID of 2 and so forth. In this design, the Z-Stick is the controller, which has the capability of receiving the signals and sending signals out. Once a device is added to the controller, with the Home Assistant up and running, the node is represented by an entity\_id. The entity is an individual component of the node and may be a sensor that is read from or a control that can be operated. Each Z-Wave device has a command class associated with it; these classes define what information is being passed around in the Z-Wave network. These classes help with programming the device because a light switch does not need to know how to respond to a request for a temperature reading. Each device belongs to one or more command classes and the associated application layer protocol functionality is included during compilation of its firmware image. The device gives the controller this information during the pairing operation [23].

Figure 6: Raspberry Pi 3 Model B

**Raspberry Pi**

The Raspberry Pi 3 model B is comprised of many components. It is a single board computer that has a SoC Broadcom BCM2837 chip. This chip is an ARM Cortex-A53 central processing unit with a quad core 64bit processor. It is running at 1.2 GHz. It also has a Broadcom VideoCore IV, that runs at 400MHz [24]. The Raspberry Pi has 1 GB LPDDR@ of RAM memory. RAM memory is known as Random Access Memory. It is volatile memory that is in the form of storage for the Raspberry Pi 3 [25]. It has a 10/100 Ethernet and a BCM43143 2.4GHz 802.11n wireless internet LA/N and Bluetooth connections. Ethernet is a way to connect computers over an internet connection. Usually this is on a network over a wired connection. It will allow the connections of multiple devices [26]. There is a Bluetooth 4.1 Classic radio support connection on board as well. It has a microSD slot for loading the operating system. MicroSD is a type of removable flash memory card used for storing information. SD is an abbreviation that stands for Secure Digital [27]. The board also has a GPIO 40-pin header. For connection for other devices. GPIO stands for General-purpose input/output. They are generic pins that are used for other digital control lines for the system [28]. It has an HDMI and 4 USB ports. HDMI is a high-definition Multimedia interface technology that is the leading standard for connecting High definition and ultra-high definition equipment [29]. USB stands for Universal Serial Bus. “It is an industry standard that was developed to define cables, connectors and protocols for connection, communication, and poser supply between personal computer and peripheral devices [30].” This also includes a DSI, or a Display Serial Interface, connector and a 2.5 Amp – 5-volt micro USB port for power [31].

**Raspbian**

The Raspberry PI 3 Model B is a computer that runs an operating system and cannot function without this. There is a good amount of operating systems to choose from but the Raspberry Pi Foundation provides the Raspbian Operating System as the primary operating system. Raspbian comes with over 35,000 packages, pre-complied software bundled in a nice format for easy installation on Raspberry Pi. It is an easy to use Operating System and built for Raspberry Pi. It comes with Chromium, a web browser similar to Google Chrome, Minecraft, Mathematica, and other programs that are good but are not necessary to report.

To make Raspbian the Operating System one will need to do a few things. Go to Raspberrypi.org and then downloads for operating systems. Download “Raspbian Stretch with Desktop” zip file. Then unzip the image and write image to file. Insert the SD card into the SD card reader. Then use the SD card slot if there is one, or an SD adapter in a USB port. Note the drive letter assigned to the SD card. It is visible in the left-hand column of Windows Explorer, for example “G:”. Download the Win32DiskImager utility from https://sourceforge.net/projects/win32diskimager/ as an installer file, and run it to install the software. Run the Win32DiskImager utility from the desktop or menu. Select the image file that was extracted earlier. In the device box, select the drive letter of the SD card; select the correct drive because if the wrong drive is chosen it could destroy the data on the computer's hard disk. Click 'Write' and wait for the write to complete. Exit the imager and eject the SD card [32].

The SD card is now ready to be put into the Raspberry Pi Micro SD card holder and can be turned on. Connect the Raspberry Pi to a TV or monitor that has HDMI capability. Powering on the Raspberry Pi will automatically log in and brings the desktop to be viewed first. One of the downsides to using the Raspberry Pi when working on a project like this is the need to use the TV or monitor while using another computer for reference. To get around this it is recommended to use a Virtual Network Computing (VNC) server. VNC is a graphical desktop sharing system that uses the Remote Frame Buffer (RFB) protocol to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network [33].

**Virtual Network Computing Server**

To implement the VNC server on the Raspberry Pi the first thing to do is set up the Wi-Fi on the Raspberry Pi. Before downloading anything onto the Raspberry Pi it is wise to update the system. Open a terminal and then the command “sudo apt-get update” does this. The next step is to download the VNC software. The command “sudo apt-get install tightvncserver” will do this. After installation of the VNC server type “sudo raspi-config” in the terminal, then navigate to Interfacing Options and scroll down to select VNC and change it to Yes. This will take some time to implement but after that one can start the VNC server by typing “vncserver” in the terminal. The best thing to do after this is to change the password so one doesn’t have others controlling the Raspberry Pi. The other computer, that will be controlling the Raspberry Pi, will need to download a VNC viewer to view what the Raspberry Pi is doing and control the desktop. An easy way to figure out the IP address of the Raspberry Pi is to open a command window and type “ping raspberrypi”. This should give the IP address which one needs to access the VNC server [34].

**Home Assistant**

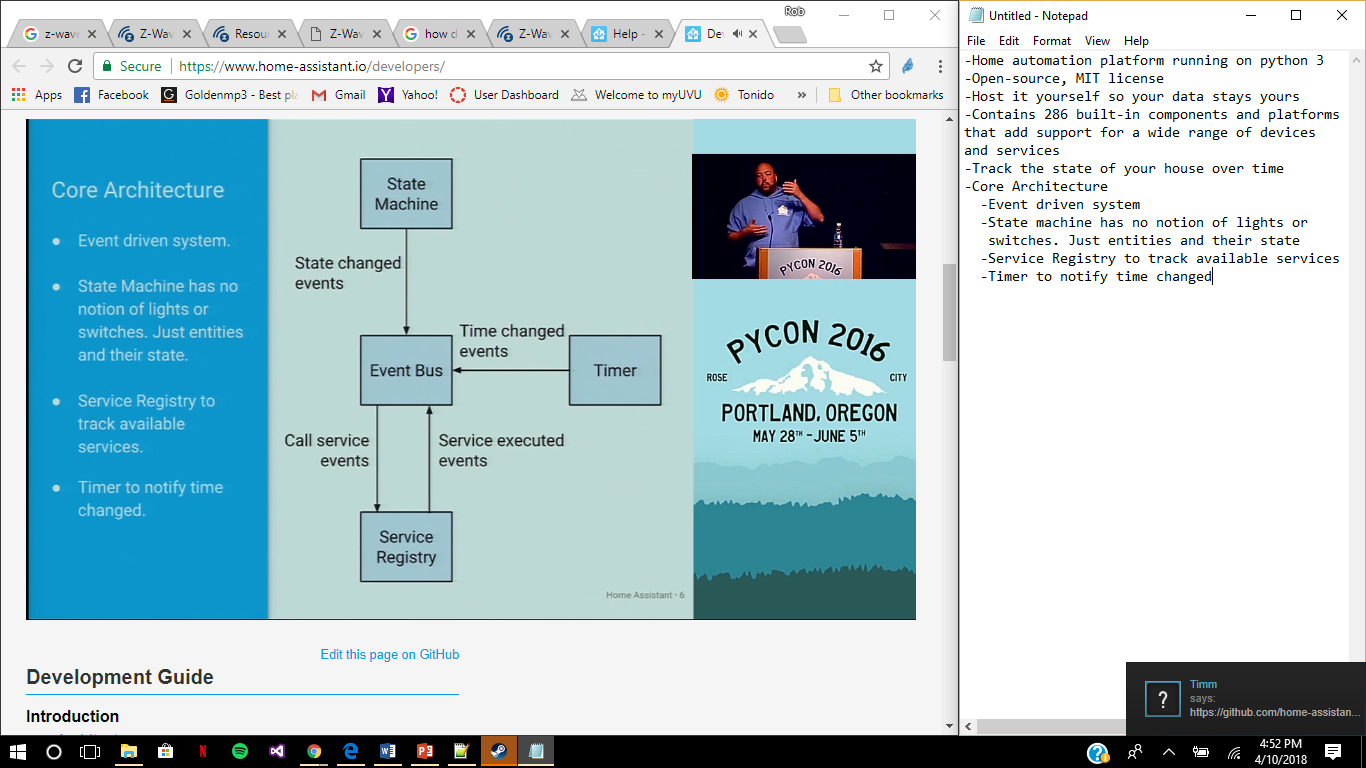
The Home Assistant was created by Paulus Schoutsen in 2013 “as a simple script to turn on the lights when the sun was setting.” Home Assistant supports over 286 different smart devices and services and provides an interface with provided data from the devices. It is an open source software that runs on any operating system that can run Python 3. Home Assistant has a desktop browser client and a mobile client to control the smart home devices from afar. The Home Assistant can connect to a lot of devices through the internet network it is connected to and can connect to a variety of devices through USB dongles that have radios in them [35]. The Home Assistant is a home automation “hub” and consists of three main functions: States, Events, and Automation Rules. States are driven by the state of the devices connected. An example is whether a light is on, the brightness of the light, and the color of the light. This data is reported to the controller and then can be displayed. An Event is something to be triggered like motion being detected or the state of the light has been changed. When an Event occurs, it triggers an Automation Rule which is action(s) to be performed. With this, there are conditions that are optional based on the current state of devices. The Core Architecture is described to be an event driven system state machine which has no notion of lights or switches but has entities and their state, service registry to track available services, and a timer to notify time changed. An

Figure 7: Home Assistant Core Architecture [35]

illustration of this is given in Figure 7. The web app that is connected to the Home Assistant uses Event Stream HTML5 javascript and doesn’t specify any sort of protocol because venders don’t make devices that connect with Home Assistant; Home Assistant makes it so it connects to all sorts of devices [35].

**YAML**

Home Assistant uses the YAML syntax for configuration and “is really powerful in allowing one to express complex configurations. [36]” YAML is a human friendly data serialization standard for all programming languages [37].” “It is a human-friendly, cross language, Unicode based data serialization language designed around the common native data structures of agile programming languages [38].” YAML is useful in a wide variety of things which include applications where data is being stored or transmitted. YAML uses python style indentation for nesting, [] for lists and {} for maps. It is a superset of JSON [39]. JSON is a javascript object notation [40]. YAML connects the configuration files, stored on the Raspberry Pi, to the Home Assistant program using a variety of syntax. Some of the basics include block collections and mappings that contain key pairs. Each of the collection items include a dash, that map a value to the key item. It is similar to a hash table or a dictionary in Python. If the indentations and dashes in YAML are not properly placed or used, the whole configuration will not work and will break the Home Assistant program. The rule is to use at least 2 spaces for each indentation. Using a hash tag (#) will allow for comments to be placed and are ignored during compilation. To include multiple files, it is necessary to use “!include “ followed by the file name for example, “Key: !include Key.yaml” would be a file that the Home Assistant can take in [39].

All the files that come with the Home Assistant are configuration.yaml, customize.yaml, groups.yaml, automations.yaml, and known\_devices.yaml. The configuration.yaml acts like the header file for the program and includes essential data for the location of the Home Assistant, templates for sensors, alarm arming and disarming, if desired, themes, passwords, Z-Wave configuration, and discovery of devices. The customize.yaml file includes renaming the devices to a friendlier name compared to the manufactured name. The groups.yaml file is used to group sensors and devices and allows for groups to form for visualization. The automations.yaml file includes most, if not all, the notifications and automation of events triggered. This is where the messages will be to be sent off to the user when events occur. The last file used is the known\_devices.yaml and includes all the known devices for the application on the phone and for connectivity for the notifications.

**Home Assistant Installation**

To install the Home Assistant onto the Raspberry Pi, open a terminal and type the following: “sudo apt-get update” and “sudo apt-get upgrade -y” to update the system. Then install python onto the Raspberry Pi by typing “sudo apt-get install python3 python3-venv python3-pip” into the server. To add an account for Home Assistant onto the device type “sudo useradd -rm homeassistant” into the terminal. Now it’s important to create a directory for the installation of the Home Assistant and change the owner to the “homeassistant” account by typing “cd /srv”, “sudo mkdir homeassistant”, and “sudo chown homeassistant:homeassistant homeassistant” into the terminal. Now the “homeassistant” account is the owner and will be the owner of a virtual environment by typing “sudo su -s /bin/bash homeassistant”, “cd /srv/homeassistant”, “python3 -m venv”, and “source bin/activate” into the terminal. The homeassistant will now be a different prompt and this is to show that the virtual environment is running. The terminal will now show “homeassistant@raspberrypi:/srv/homeassistant $” and is now the virtual environment. It is important to install a required python package by typing “python3 -m pip install wheel” into the current terminal. Finally, it is time to install Home Assistant by typing “homeassistant@raspberrypi:/srv/homeassistant $ pip3 install homeassistant” into the current terminal. To start Home Assistant for the first time and complete the installation, create the .homeassistant configuration directory in the /home/homeassistant directory and install any basic dependencies, type “hass” into a new terminal. Home Assistant is now running and is available at <http://ipaddressofPi:8123>. “8123” is the port address which the Home Assistant uses to export. To update to the latest version of Home Assistant type “sudo su -s /bin/bash homeassistant”, “source /srv/homeassistant/bin/activate”, and “pip3 install –upgrade homeassistant” into the terminal [41].

Here are some tools to navigate using the Home Assistant. Typing “sudo systemctl restart home-assistant.service” into a terminal will restart the Home Assistant and “sudo su -s /bin/bash homeassistant” will take one to the virtual environment of the “homeassistant” account to edit the files. Once in the virtual environment type “homeassistant@raspberrypi:/home/pi$ cd /home/homeassistant/.homeassistant” into the terminal to go to where the files are located. To edit the files, type “nano” and then the file name. For example, “nano configuration.yaml” will let one edit the configuration file. Typing “ls” will display all the files in the Home Assistant to choose from.

**Parts**

To implement the Z-Wave protocol, it is necessary to have a Z-Wave controller. The Aeotec by Aeon Labs Z-stick Z-Wave Plus Gen5 ZW090 is the controller in this design that is connected through USB to the Raspberry PI. In the configuration file, the following must be added to the file:

zwave:

usb\_path: /dev/ttyACM0

This will allow the Aeotec by Aeon Labs Z-stick Z-Wave Plus Gen5 ZW090 to work [42]. Ask described earlier, there are multiple sensors and devices connected to this design and are as follows.

Figure 8: Z-Wave Z-Stick [43]

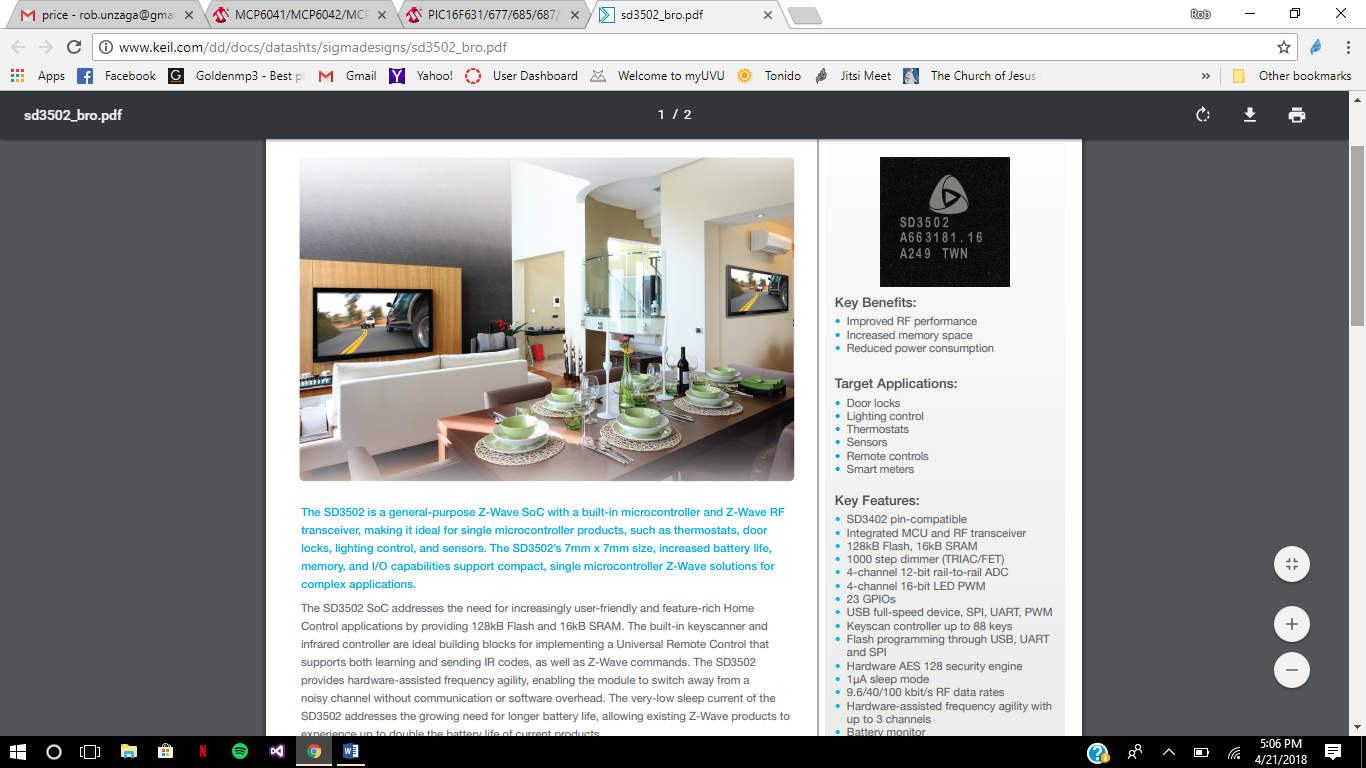
The Aeotec by Aeon labs Z-Stick Z-Wave Plus Gen5 ZW090-A has the Z-Wave frequency of 908.42 MHz and a power supply through the USB and is between DC 4.75 to 5.25 Volts. There is also built-in rechargeable lithium battery of 3.7 V. The max standby current is 30 uA and the max operating current is 40 mA. The operating temperature is between 32 to 122 degrees farenheit and should be stored in -4 to 158 degrees farenheit. The operating distance is up to 500 feet. The color is white and the weight is 2.5 oz [43] . The Z-Stick is sealed and is secret so that no one can recreate what that particular company have done.

Figure 9: Z-Wave Chip [44]

The SD3502 General Purpose Z-Wave SoC (System-on-a-chip) designed by Sigma Designs is the heart of the Z-Wave protocol. It has a built-in microcontroller and Z-Wave RF (Radio Frequency) transceiver making it ideal for single microcontroller products. The SD3502 is 7mm x 7mm, has increased battery life, memory, and I/O capabilities. The memory is 128kB Flash and 16kB SRAM. The battery life has a very-low sleep current for longer life and is backward-compatible for up to ten years. SD3502 has an integrated MCU and has 1000 step dimmer. It has 4-channel 12-bit rail-to-rail ADC and a 4-channel 16-bit LED PWM. There are 23 GPIOs and a USB full-speed device, SPI, UART, PWM. There is flash programming through USB, UART and SPI. Hardware is AES 128 security engine and has a sleep mode of 1uA. It is powered by 2.3-3.6 Volts [44].

Figure 10: Contact Sensor [45]

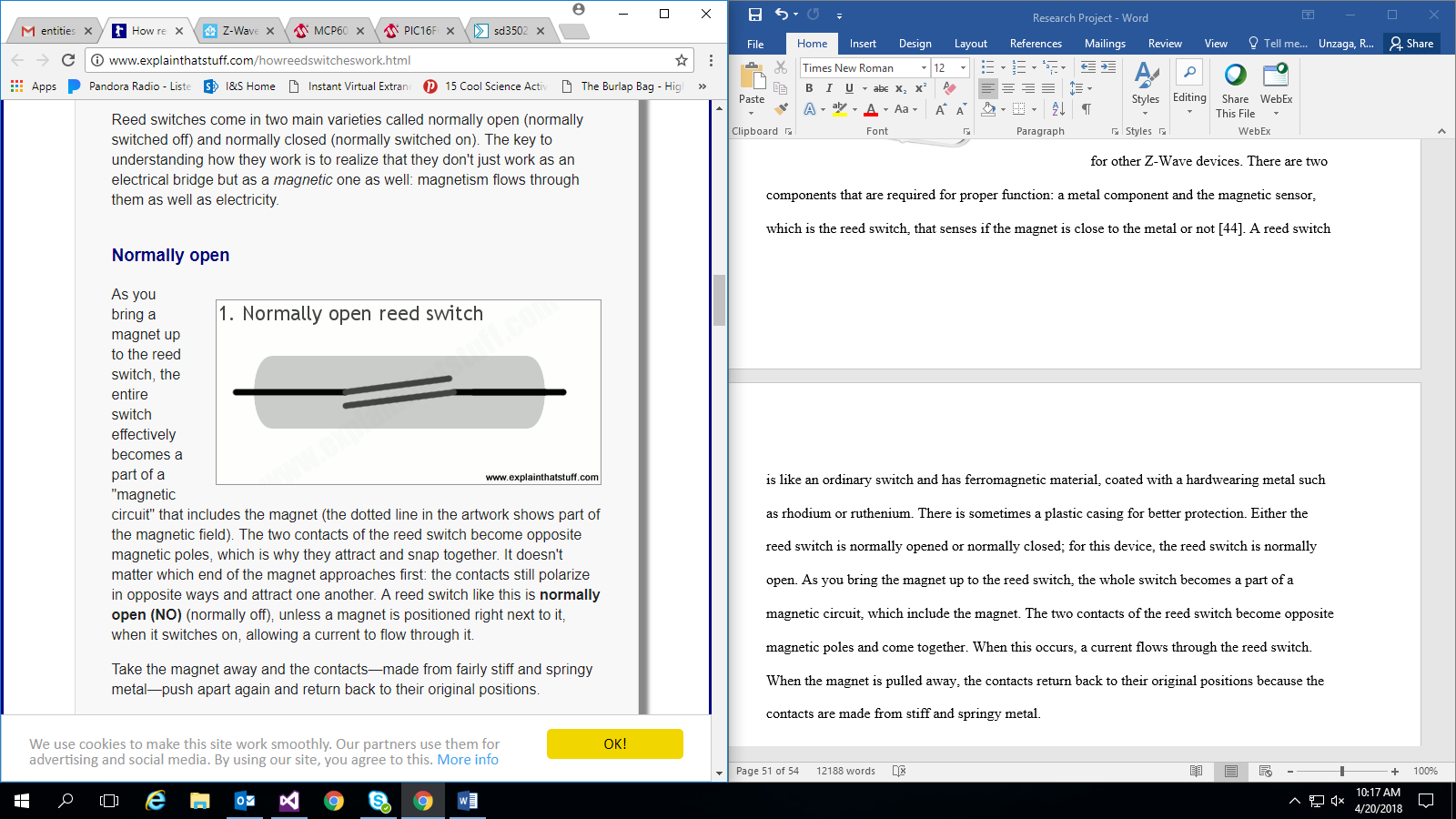
The Zooz Z-Wave Plus Contact Sensor ZSE08; this project contains two of them for the window and the door. The Contact Sensor has the Z-Wave frequency of 908.42 MHz and uses 1 CR2 battery. The power consumption is a low .13 W and is wireless up to 100 feet. The operation temperature ranges from 32 to 104 degrees Farenheight and is meant for indoor use only. It is 2.75 x 0.75 inches and weighs 0.5 oz. This can act as a Z-Wave repeater for other Z-Wave devices. There are two components that are required for proper function: a metal component and the magnetic sensor, which is the reed switch, that senses if the magnet is close to the metal or not [45].

Figure 11: Normally Open Reed Switch [46]

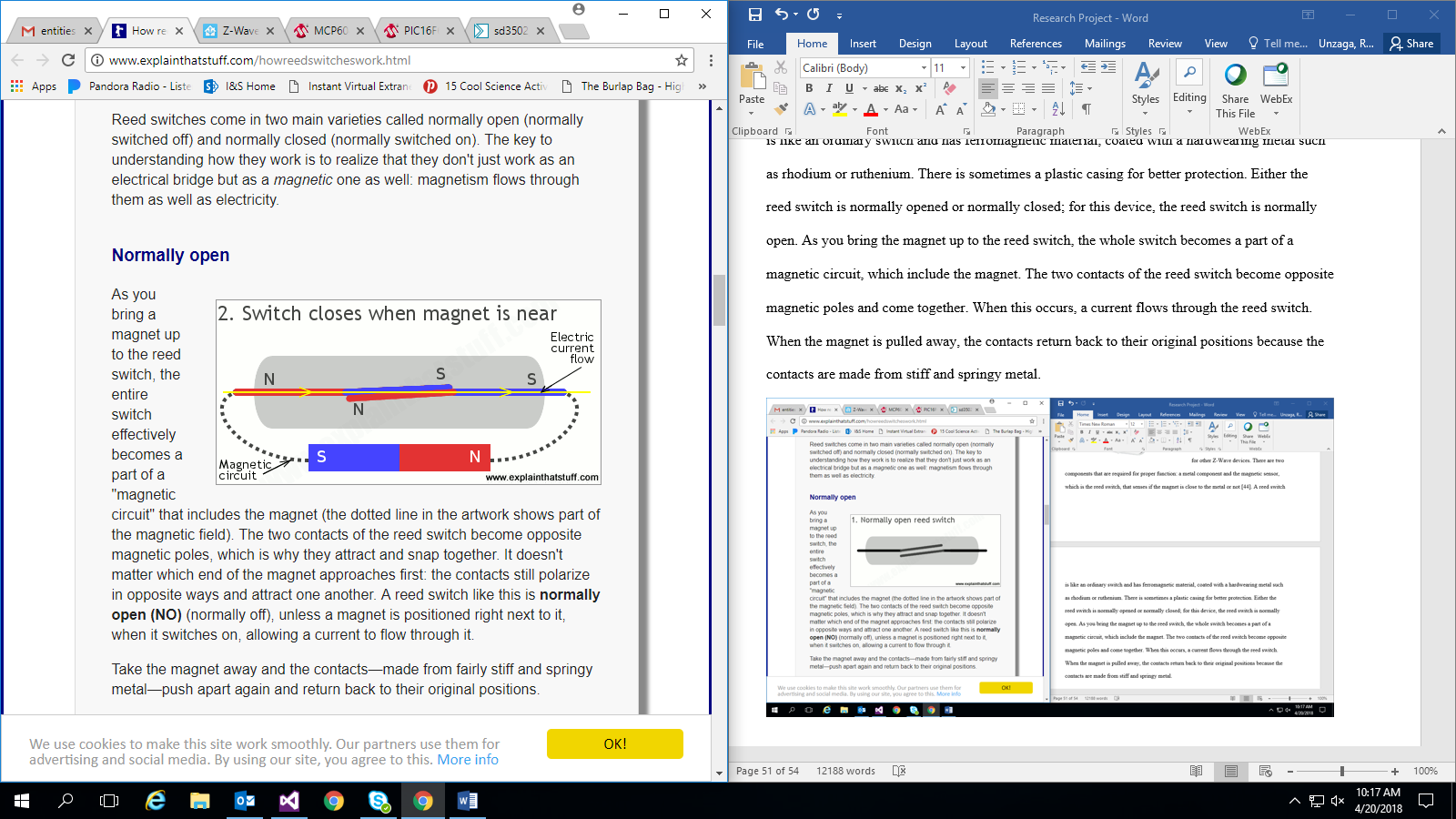
A reed switch is like an ordinary switch and has ferromagnetic material, coated with a hardwearing metal such as rhodium or ruthenium. There is sometimes a plastic casing for better protection. Either the reed switch is normally opened or normally closed; for this device, the reed switch is normally open like Figure 11. As one brings the magnet up to the reed switch, the whole switch becomes a part of a magnetic circuit, which includes the magnet.

Figure 12: Switch Closes When Magnet Is Near [46]

The two contacts of the reed switch become opposite magnetic poles and come together like in Figure 12. When this occurs, a current flows through the reed switch. When the magnet is pulled away, the contacts return back to their original positions because the contacts are made from stiff and springy metal [46]. Along with the reed switch, the Contact Sensor also comes with the Z-Wave chip.

Figure 13: PIR Motion Detector and Temperature Sensor [47]

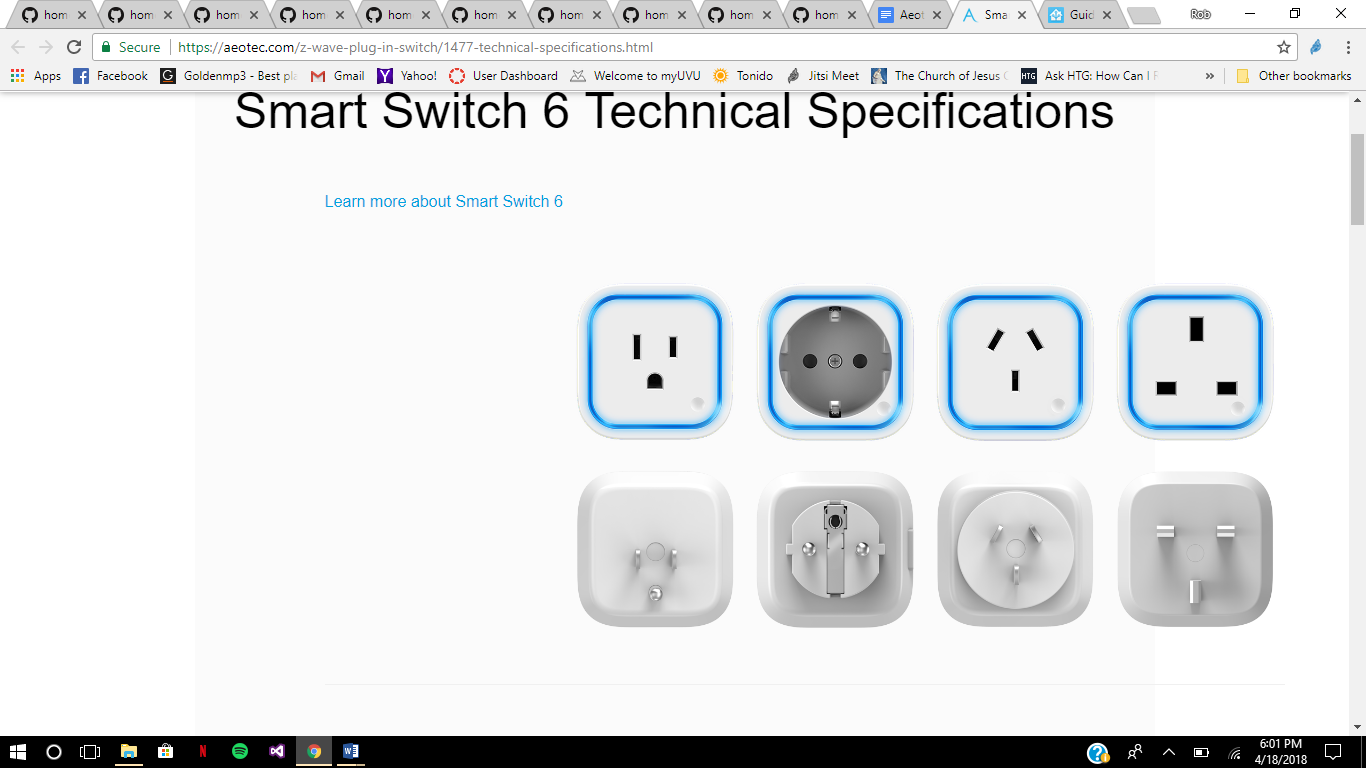
The Z-Wave Plus PIR Motion Detector with Temperature Sensor is a device that uses a passive infrared sensor and sends a Z-Wave Plus signal when motion is detected within its detection range. It can detect moving objects within line of sight at distances up to 32 feet away with a field of view up to 120 degrees. The temperature sensor detects changes in temperature and reports the temperature to the network whenever a significant change has occurred. This device acts as a repeater for other Z-Wave devices. This comes with mounting screws and adhesive tape. This is powered by a CR123A Lithium Battery [47]. The Motion Detector has a MCP6041/2/3/4 mircochip, which is 600 nA, Rail-to-Rail Input/Output Op Amp. The Motion Detector also has PIC16F631/677/685/687/689/690, a microchip that is a 20-pin Flash-Based, 8-bit CMOS Microcontroller. It features low power by having a Standyby Current of 50nA at 2V; Operating Current of 11uA at 32 kHz at 2V and 220uA at 4MHz at 2V, Watchdog Timer Current of less than 1uA at 2V. It has 17 I/O pins and 1 Input-Only Pin and has an Analog Comparator Module, A/D Converter, Timer, Enhanced Capture, Compare, PWM+ Module, Synchronous Serial Port, and I2C [48]. This will be placed in the main area and will detect motion and temperature.

Figure 14: Smart Switch [49]

Aeotech Z-Wave Plus Smart Switch 6 model ZW096-A02 is a 120 Volt AC, 5 amp smart plug with a 5 volt, 1000mA USB port on the side. It has a frequency of 908.42 MHz. The dimensions of the plug are 56mm x 56 mm x 35.5mm. The smart plug can operate in a temperature of 0 to 40 degrees celsius. It had a wireless Z-Wave Plus range of up to 100 meters. It can act as a Z-Wave repeater for other Z-Wave devices and is a 5th generation Z-Wave Plus Device [49] . The Smart Switch is sealed and is secret so that no one can recreate what that particular company have done. This is used to allow anything plugged in to be turned on/off remotely or manually by pressing the sensitive touch button on the bottom right. This can be seen in Figure 14.

Figure 15: Kwikset Smartcode Door Lock [50]

Kwikset Smartcode Door Lock has a touchpad electronic deadbolt and is a one-touch locking deadbolt that is motorized. It takes a personalized code which is entered for convenience of being keyless. The keypad has a back-lit keypad to help with visibility. This is easy to install and takes minutes with a screwdriver. The Door Lock needs a door with a 2-1/8 inch diameter face hole and a 1 inch diameter latch hole. The door thickness needs to be 1-3/8 inch to 1-3/4 inch. The faceplate is 1 x 2-1/4 inch and the bolt is 1 inch throw. The Lock takes 4 AA batteries [50]. The Door Lock is sealed and is secret so that no one can recreate what that particular company have done. There is a circuit board that can be looked at. On the board, the Z-Wave chip can be seen but everything else is blocked out, meaning there is literally black marking on the components where there would be Parts Identification. The Door Lock is used so that anyone can type in the code, if they know it, and also lock/unlock the door remotely by using the app. To add the door lock takes extra steps to make it more secure. It can’t be added like the other devices and will require a secure network key code and be added as a secure node [51]. First, there needs to be a secure network key added to the configuration.yaml file on the Raspberry Pi. Under the “zwave:”section, that will be discussed in greater detail later in the paper, “network\_key” will be placed with a 16 byte value hex random numbers. After this has been implemented, the door lock can then be added as a secure node; this must be done on the Home Assistant display and can be down by going to the menu, then Config, then hitting the Z-Wave Network and pressing “Add Secure Node” [52]. The Z-Stick will be in Learning Mode where the door lock will need to be added to the network and can be done by pressing the z-wave button on the top left side of the door lock on the inside.

Figure 16: Web Camera [53]

Cimkiz A871 Web Camera has a USB cable length of 56 inches and has a recommended resolution size of 640 x 480 pixels. The speed rate is 30 fps/s under VGA model. The output format and interface type is USB2.0 with an imaging distance of 50 Hz. The dynamic image capture format is AVI [53]. This will be used for surveillance of the home and can provide a live feed video for the user. The Web Camera will not be included with the Home Assistant but is controlled by the Raspberry Pi and uses Jitsi Meet for two-way communication. To have this two-way communication, a speaker is added to the design.

Figure 17: 808 CANZ Bluetooth Wireless Speaker [54]

The 808 CANZ Bluetooth Wireless Speaker SP880 is 3.25 x 2.5 inches and uses Aux-in Jack that uses a 3.5mm cable. It has a high capacity Li-Ion Rechargeable Battery that provides up to 6 hours of playback on a single charge. It has a micro USB port for recharging. The weight is 0.5 pounds [54]. This will be used to communicate with whoever is at home as the speaker for them to hear the user’s voice.

**Adding Devices**

To add the devices to the Z-Wave network the Z-Stick needs to be unplugged from the USB and the button on the outside needs to be pressed and held down until the lights start to flicker yellow. When this happens the Z-Wave controller is ready to receive any Z-Wave device. Hold the Z-Stick up to a device while that device is waiting to connect. Pressing the button on the device that is being added to the network repeatedly helps connect quicker. Once the lights blink blue on the Z-Stick the device has been added successfully and is ready for configuration. Going to <http://ipaddressofPi:8123> will show what the sensor names are which will be used in the configuration to give as friendly names such as “Window” or “Front Door”. The iPhone Application is a copy of what is displayed on the web. To download, go to the Apple App Store and search Home assistant. One will need the URL <http://ipaddressofPi:8123> and password, which will be created in the configuration file, and then press save. Any time the settings regarding the notifications is changed it is necessary to update the push settings in the settings menu on the Home Assistant app.

**Away from Home**

To make the Home Assistant available while away from home, which is one of the most useful features, Port Forwarding is required and a Domain Name System (DNS). To port forward, it is needed to log into the network router and port forward. This is a different process for every router and will most likely be the modem. External port 8123 needs to be forwarded to internal port 8123 on the internal IP Address of your Raspberry pi. A Dynamic DNS service is required to access the Home Assistant outside the network and is the next step. While forwarding ports, Port 80 needs to be forwarded to Port 80 on the Raspberry Pi as well as 443 to 443. This is to verify the certificate.

Duck DNS is a Dynamic DNS (DDNS) service and anyone can get a subdomain and use one of their provided scripts to update their records. This is good because this makes it so that there is no need to remember an IP address. Instead, there is a domain name that is kept up-to-date by a computer at home. The DDNS server allows an easier way to access the Home Assistant from anywhere around the world from a web browser [55].

Go to [www.duckdns.org](http://www.duckdns.org) and login using one of the choices of methods at the top of the website. It will automatically create an account and is ready for a name of the subdomain. This project is under “uvuautomation”. The Raspberry Pi needs to have Duck DNS installed so go to the top and click install. Under “Operating Systems” select “pi” and then choose the domain that was created. In this case, select “uvuautomation”. In a terminal on the Raspberry Pi, type “mkdir duckdns” into a terminal to make a directory for the duckdns script. Next, type “cd duckdns” to go to the directory that was just created. Type, “nano duck.sh” to create a duckdns script. Then, type “echo url="https://www.duckdns.org/update?domains=uvuautomation&token=4d226050-a039-4e0b-9239-f2e60e82166c&ip=" | curl -k -o ~/duckdns/duck.log -K –“ into the file and then save and exit. To have the script run, type “chmod 700 duck.sh” into the terminal and then type “crontab -e” and then type “\*/5 \* \* \* \* ~/duckdns/duck.sh >/dev/null 2>&1” into the bottom to have the program run every 5 minutes and then save and exit. To run everything, type “./duck.sh” into the terminal. To check if everything was done correctly, type “cat.duck.log” into the terminal. If it says “KO” something went wrong; if it says “OK” then everything went well. Return to the root directory by typing “cd” into the terminal. To make sure everything is working, go to the website that was made by duckdns. This project is “https://uvuautomation.duckdns.org:8123” and should work outside the local network [56].

For the Secure Sockets Layer (SSL), which is the standard security technology for establishing an encrypted link between a web server and a browser and ensures that all data passed between the web server and browsers remain private and integral [57], type “mkdir certbot” to make a directory for the SSL certification bot. Next, go to that directory by typing “cd certbot”. Next, type “wget <https://dl.eff.org/certbot-auto>” to download the certbot. Type “chmod a+x certbot-auto” to allow the certbot to run. To run the certbot correctly, type “./certbot-auto certonly --standalone --standalone-supported-challenges http-01 --email YOUR@EMAIL.ADRESS -d YOURSUBDOMAIN.duckdns.org” and replace the email and subdomain respectively. There will be some questions that need to be answered as they are posed. Now, remove the port 443 and port 80 forwarding from the router. Now, type “sudo chmod -R 777 /etc/letsencrypt” to allow the Home Assistant to read the certificates. In the Home Assistant configuration.yaml file, add the ssl\_key: and ssl\_certificate under “http:” section and also have the “base\_url:” and have the duckdns address in there. Now, try and access the Home Assistant by going to <https://YOURDOMAIN.duckdns.org:8123> [58].

**Jitsi Meet**

Jitsi is a set of open-source projects that allow easy build and deployment of secure videoconferencing over the internet. Jitsi passes everyone’s video and audio to all participants and doesn’t mix them first, like a lot of other videoconferencing and results in a lower latency, better quality, and a much more scalable and inexpensive solution. Jitsi was started in 2007 and is written in Java [59]. Jitsi Meet is used to have a videoconference with other users and is secure with encryption by default and advanced security settings. While on Jitsi Meet, a password can be placed so no one else can view the video and audio unless they have the password [60].

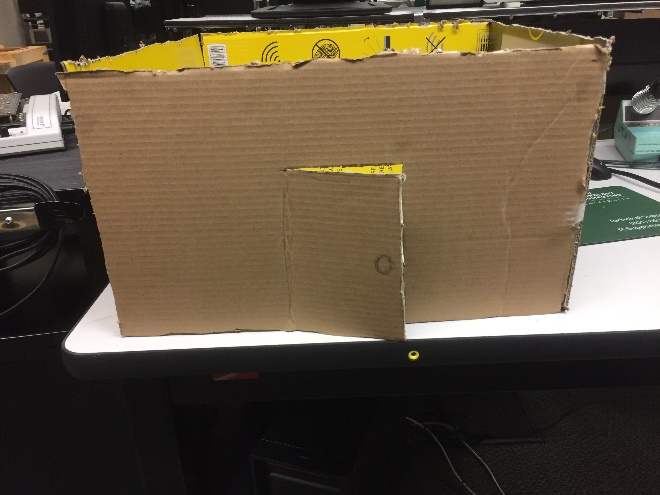
To use Jitsi Meet is easy and can be started by going to <https://meet.jit.si/> and then creating a conference by typing a name in. This design’s name is UVUCamera and can be accessed at <https://meet.jit.si/UVUCamera>. To have the Raspberry Pi connect to Jitsi automatically, connect the Web Camera to the USB and restart the Raspberry Pi. Go to <https://meet.jit.si/UVUCamera> or the website that has been set up by another project. Then the Raspberry Pi must grant access to the camera and microphone manually, if not automatically, and then it’s set to go. To connect the Raspberry Pi automatically to Jitsi Meet, type “sudo nano ~/.config/lxsession/LXDE-pi/autostart” in the terminal and add “@chromium-browser --kiosk --disable-session-crashed-bubble --disable-infobars --disable-restore-session-state <https://meet.jit.si/UVUCamera>” to the file that is brought up. This will make the web browser come up when the Raspberry Pi restarts. Adding a password to the conference is critical because without this, others can view the videoconference.

Figure 18: Door Prototype

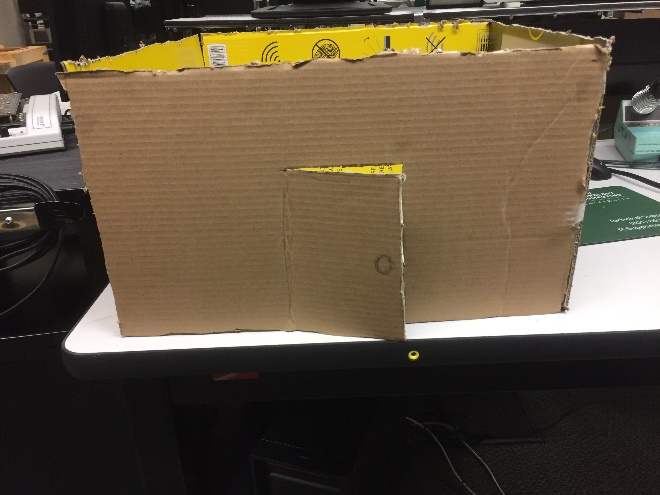
To visualize what this project can do, a home has been designed and built. Figure 18 is the front of the prototype and is of the hypothetical door.

Figure 19: Window Prototype

Figure 19 is the prototype for the window and is the side of the home.

Figure 20: Layout Prototype

Figure 20 is the prototype general space that is expected when creating the final design. The Prototype was created to get a general idea of space that is wanted for the project.

Figure 21: Built Model

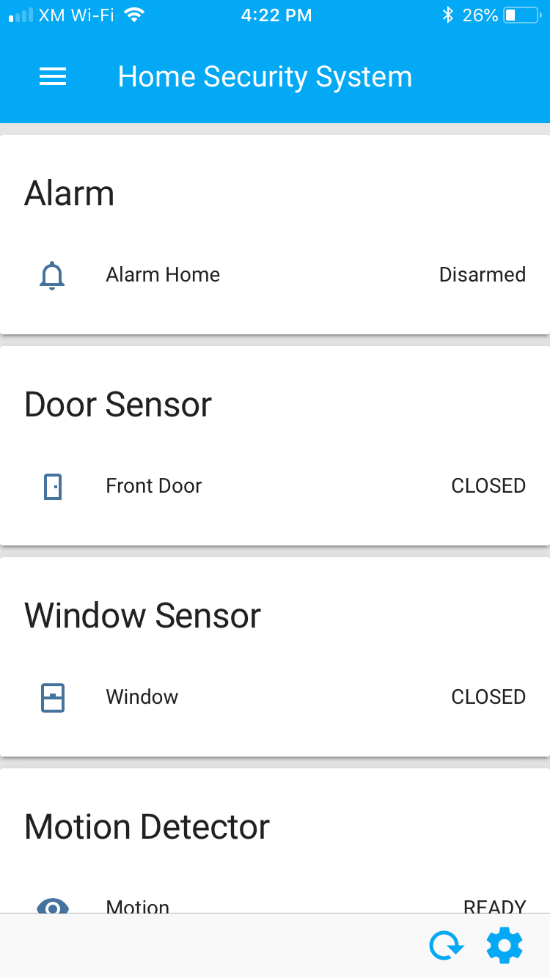
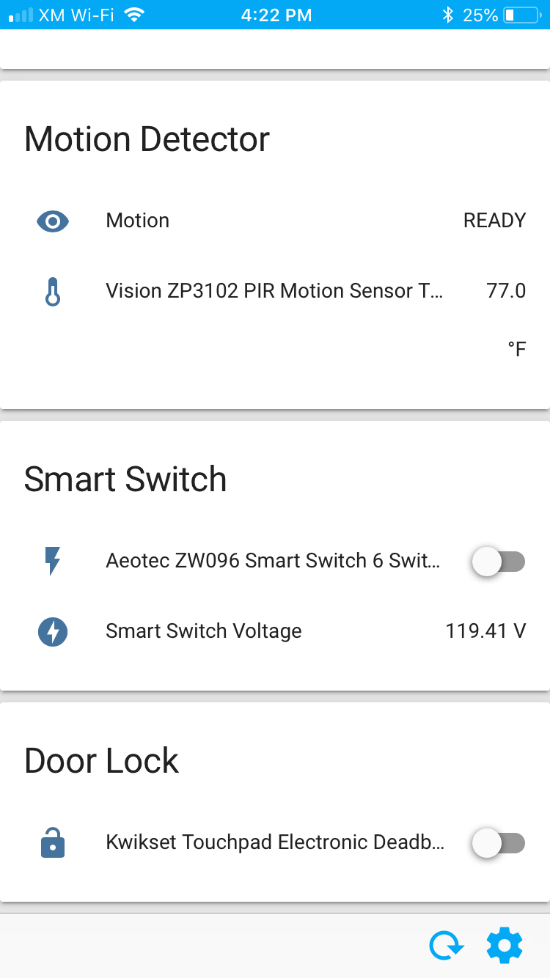
Figures 21 is the final design that was built and put together in 9 hours. The window and door are clear plastic and the devices are to be placed with Command Brand Damage-Free Hanging tabs.

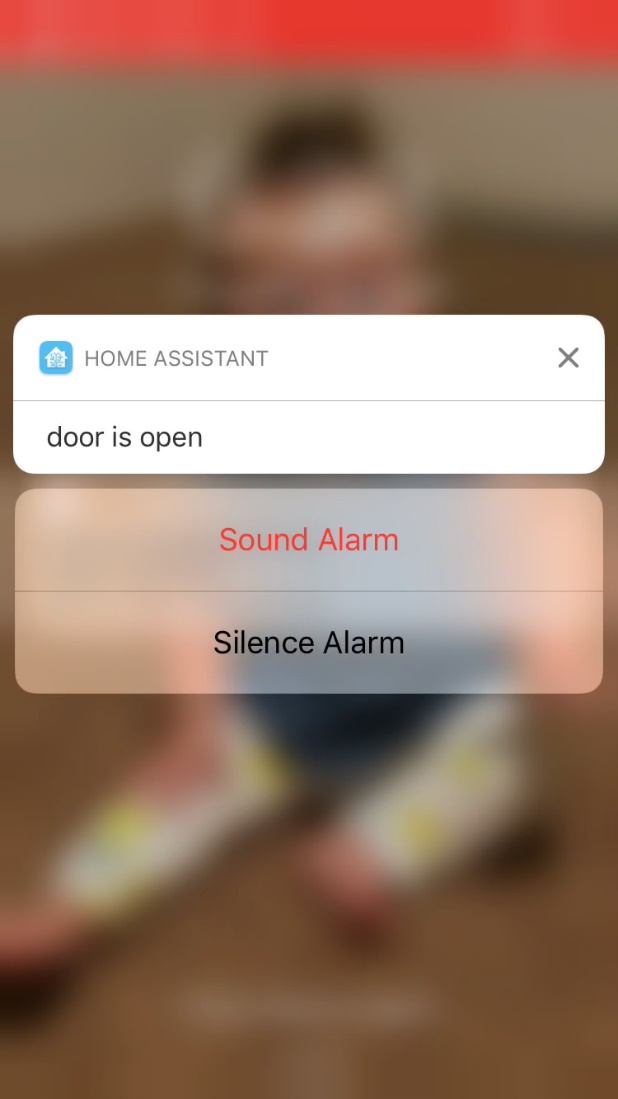
Figure 22: Command Tabs

Figure 22 shows the Command Tabs which look a lot like Velcro. This approach was to ensure nothing in the design was stolen because the devices were put in place and then removed daily.

Figure 23 is the final look to the design with the code written for reference in Appendix B. There is an alarm arming and disarming which has two ways of arming. The Arm Stay only arms certain devices while one is at home because motion detectors will go off while moving around at home. This helps to keep notifications to a minimum. The Arm Away will enable all devices

to send notifications when motion is detected, the door is opened, and the window is opened. There is a code to arm this that helps keep it secure and gives 15 seconds before securing the home allowing time for the user to exit the home.

Figure 23: Final View of Home Assistant

Figure 24: iPhone Push Notifications

The notifications that are sent have actions associated with them on the iPhone. Figure 24 shows what this looks like. These actions can only occur when the fingerprint on the iPhone is authorizing the action making it more secure.

The Contact Sensors connected to the door and window, respectively, only are displaying the current state, which could be “OPEN” or “CLOSED”. The Motion Detector also shows the current state, which could be “READY” or “DETECTED”. The Smart Switch has two sensors that are being utilized in the design: the switch and the current voltage reading. The switch allows the user to turn on the outlet and the current voltage reading allows the user to see how much voltage is currently being used. The Door Lock shows whether the door is locked or not. The switch is there to lock or unlock the door. Whenever the switch is set to “off” the door is unlocked and when it is set to “on” the door is locked.

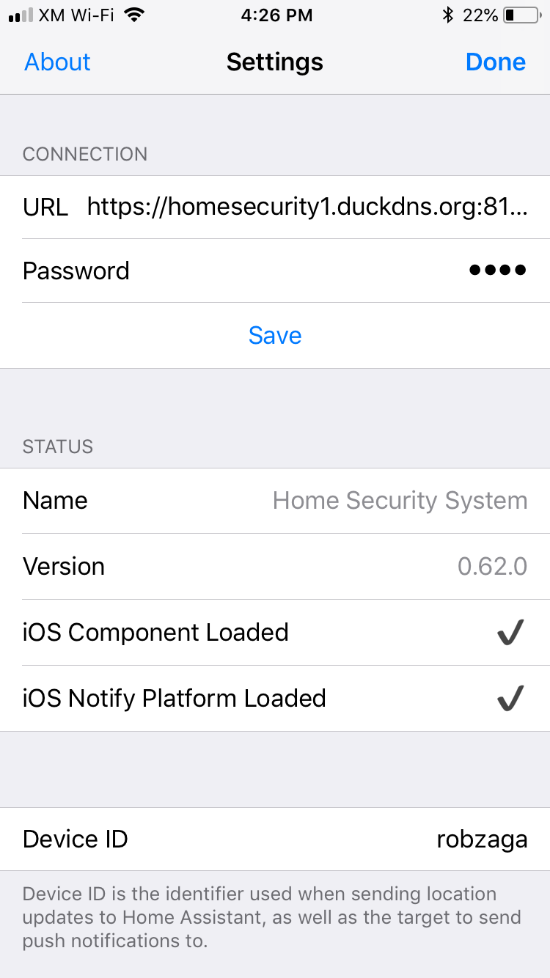
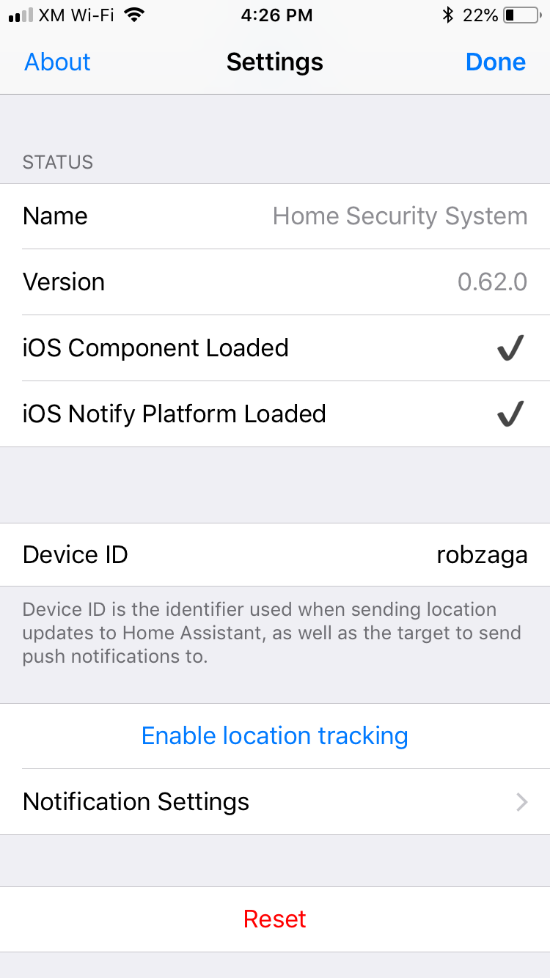
Figure 25: iPhone Home Assistant Settings

Figure 25 shows the settings on the iPhone application, which can be found on the App Store for free. The URL is the website the Home Assistant can be found at, the password is the password set and makes it easier to access Home Assistant without putting that information in every time. The notification settings is how the push notifications are enabled and should be pressed followed by pressing the “Update push settings” button.

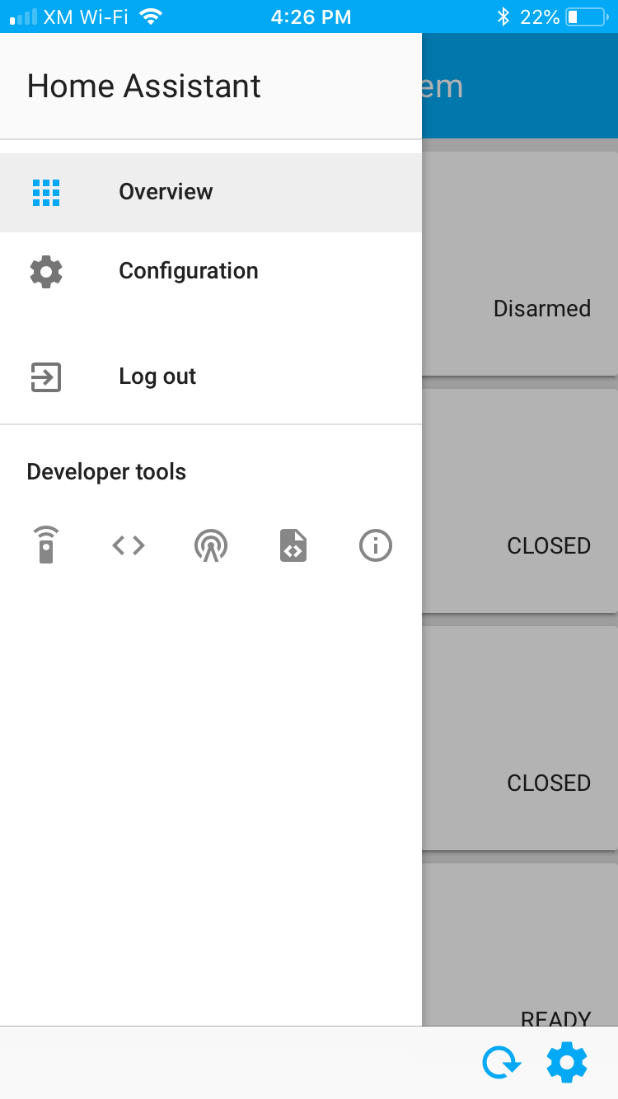
Figure 26: Home Assistant Menu

Figure 26 shows the menu of the Home Assistant. The “Overview” is the front display, the “Configuration” is how to access the Home Assistant settings and include the Z-Wave network settings, the automation settings, etc. “Log out” is how to log out of the Home Assistant. The “Developer tools” are useful to see what types of devices are being used and what services can be called. The first tool is used for calling services. An example of this would be to turn off the Home Assistant or restart it. The second tool is used for the states of the devices and sets the representation of an entity. The third tool is for Events and fires events. The fourth tool is templates and renders templates. The previous two tools weren’t used much at all. The first two tools were used quite often. The fifth and final tool is info and has detail about the Home Assistant. When there is a problem, the Home Assistant might not compile but when it does, there are messages in the overview. The details of those messages can be found in the info with more details of where the problem is and why there are problems.

Figure 27: Door Lock, Camera, and Smart Switch

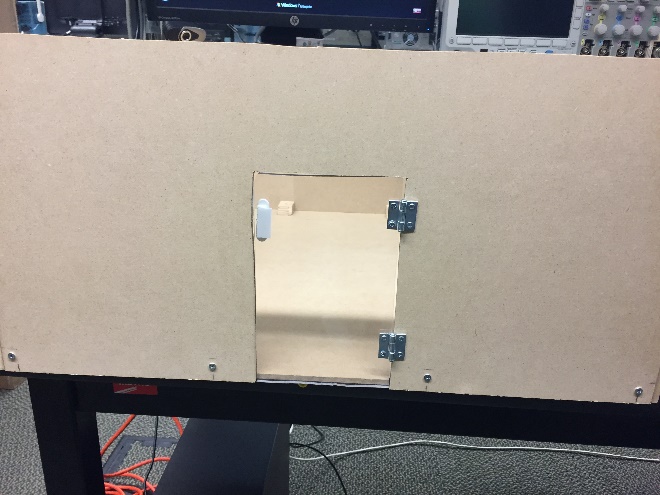
Figure 27 shows the Door Lock displayed and installed on a piece of wood. The Smart Switch is plugged into an extension cord at the time. The Webcam is installed in the corner and has a USB cord connected to the Raspberry Pi.

Figure 28: Door with Contact Sensor

Figure 28 shows the door, which is plastic, connected with the Contact Sensors.

Figure 29: Window with Contact Sensor and Motion Sensor

Figure 29 shows the sliding window with the Contact Sensor attached to it. The Motion Sensor has a range of 120 degrees; the Motion Sensor has a temperature sensor as well.

Figure 30: Speaker, Z-Stick, and Raspberry Pi

Figure 30 shows the speaker attached to the Raspberry Pi through the Aux-Cord. The Z-Stick is connected through USB to the Raspberry Pi. Ideally, the Raspberry Pi would be in a stable spot and inside another box where no one can tamper with it.

**Code explanation**

The Home Assistant files, like discussed earlier, can be accessed by typing “sudo su -s /bin/bash homeassistant” and “homeassistant@raspberrypi:/home/pi$ cd /home/homeassistant/.homeassistant” into a terminal; this is how to get to where the files are located. To see a list of files, type “ls” into the terminal and to edit a file, type “nano” and then the file name. The first file that will be discussed is the configuration file. To get to this type “nano configuration.yaml” into the current terminal. The code that is in this design follows:

homeassistant:

name: Home Security System

unit\_system: imperial

time\_zone: America/Denver

The “homeassistant” is there to say that this particular Home Assistant system consists of the following. The Name of the system is given by “name: Home Security System” and will be displayed in the main display. The unit system is for preference and is used when talking about distance and temperature. The Time Zone is the current Time Zone. The Home Assistant uses the country first followed by a slash and then the region. For UVU, it is “America/Denver” [51].

customize: !include customize.yaml

group: !include groups.yaml

automation: !include automations.yaml

To include files to use in the Home Assistant, it is necessary to explain what the file will be used for and then which file to include. To customize, “customize: !include customize.yaml”; to group, “group: !include groups.yaml”; for automations, “automation: !include automations.yaml”.

sensor:

- platform: template

sensors:

front\_door:

friendly\_name: Front Door

entity\_id:

- binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor

value\_template: >-

{%- if is\_state(“binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor”, “on”) -%}

OPEN

{%- else -%}

CLOSED

{%- endif -%}

icon\_template: >-

{%- if is\_state(“binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor”, “on”) -%}

mdi:door-open

{%- else -%}

mdi:door-closed

{%- endif -%}

Templates are used to customize sensors in specified circumstances. In this design, the default is “on” and “off” for each sensor. To make it more user friendly it is better for it to say “OPEN” or “CLOSED” and so on. By typing “front\_door:” it is now specified that this particular sensor will now be named “front\_door”. The “friendly\_name: Front Door” is the name that will be on the display. To connect the name “front\_door” to the sensor the “entity\_id:” is required. The Contact Sensor that is hooked up to the front door is called “binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor” and can be found on the Home Assistant by going to <http://ipaddressofpi:8123>. The “value\_template: >-“ provides the Door to change from “on” and “off” to now display “OPEN” and “CLOSED”. To do this, an if-else statement is used and uses the sensor name and a conditional statement where “is\_state” is where if the state of the particular sensor is a particular state. In this case, if the contact sensor is on then the value is displayed “OPEN”; else it is “CLOSED”. The “icon\_template: >-“ is meant to do the same as above but instead of changing the value, the icon will be switched depending on the state of the sensor. The icons can be found at <https://materialdesignicons.com/> and are the only icons that can be used. To implement, use “mdi:” and then the name of the material design icon [51].

window:

friendly\_name: Window

entity\_id:

- binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3

value\_template: >-

{%- if is\_state("binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3", "on") -%}

OPEN

{%- else -%}

CLOSED

{%- endif -%}

icon\_template: >

{% if is\_state('binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3', 'on') %}

mdi:mdi:window-open

{% else %}

mdi:mdi:window-closed

{% endif %}

Almost the same thing is going to happen with the Contact Sensor that is used for the window apart from the sensor name and the icons. Note that the icons, for an unknow reason, needed an additional use of “mdi:” for them to work.

motion:

friendly\_name: Motion

entity\_id:

- sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar

value\_template: >-

{%- if is\_state("sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar", "8") -%}

DETECTED

{%- else -%}

READY

{%- endif -%}

The PIR Motion Sensor is similar to the above sensors with the exception of the detection. The Motion Sensor comes with a few different detections but for this design, the “burglar” detection is used with a value of 8 meaning there is a detection. When there is no detection, the burglar value is at a 0. When the motion is at a 3 the motion is being opened at any time but is not used in this project. The values will be changed to “READY” meaning it is ready to detect and “DETECTED” meaning there is motion detected.

alarm\_control\_panel:

- platform: manual

name: Alarm Home

code: 1111

pending\_time: 0

trigger\_time: 0

armed\_away:

trigger\_time: 0

pending\_time: 15

armed\_home:

pending\_time: 0

trigger\_time: 0

The “alarm\_control\_panel:” is the alarm system that is used to manually set the alarm stay, away, or disarm. Manually setting the alarm is helpful for the notifications to work correctly as described earlier; if the user is at home, it wouldn’t make sense for the motion detector to be armed unless in a remote room while having the door and window sensors armed. The “platform” is there to show that it is a manual setting. The name is going to be the display name and the code is the code that is used for arming and disarming. The “pending time” is there for the user to arm the home and gives time for that user to leave the home before the motion sensor is armed. All other time delays are 0 because it is not important to delay disarming or arming while at home [51].

frontend:

themes:

day:

primary-color: yellow

night:

primary-color: blue

The “frontend:” is for the display where the themes change according to whether it is day or night and is just to allow for a user-friendly display.

config:

The “config:” is to enable the user the ability to edit the features associated with the Home Assistant. These features include the Automation, Z-Wave Network, etc.

http:

base\_url: https://URL:8123

api\_password: password

ssl\_certificate: location of the certificate

ssl\_key: location of the certificate

This is the section where the port forwarding on the modem is used in the Home Assistant. The “base\_url:” section is where the Duck DNS website is used. The api\_password is the specific password that is chosen for the Home Assistant and makes it more secure. Without the password, no one can access the Home Assistant. The “ssl\_certificate:” and the “ssl\_key:” are used for encryption and make the Home Assistant even more secure while putting the Home Assistant out on the web. Putting that specific information out on the web is dangerous. If it is not secure, anyone could have access to the door lock and the sensors to turn them off.

zwave:

usb\_path: /dev/ttyACM0

network\_key: “0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00”

As described earlier, the “zwave: usb\_path:” is there to interact with the Z-Wave controller and is called the Z-Stick. There can be no interactions with the Z-Wave protocols and the devices without this controller. The “network\_key” is a random assortment of hex number bits for the Door Lock, which is a secure node.

discovery:

ignore:

- roku

- samsung\_tv

- netgear\_router

- apple\_tv

Having “discovery:” here makes the Home Assistant discovering anything connected to the same network that is connected to the Raspberry Pi.

Ignoring devices is helpful when the Home Assistant is discovering all devices because there are devices that will show up that the user will not want to use. To figure out what the name of the device the user doesn’t want to include, the user must go online to the Home Assistant website because each device is given a different name.

ios:

push:

categories:

- name: pushbuttons

identifier: 'pushbuttons'

actions:

- identifier: 'IGNORE'

title: 'Ignore'

activationMode: ‘foreground’

authenticationRequired: yes

- identifier: 'SOUNDALARM'

title: 'Sound the Alarm!'

activationMode: ‘foreground’

authenticationRequired: yes

The “ios:” is where the push notifications can be configured into the Home Assistant with the iPhone. The “push:” signifies that it is push notifications that is wanted and the “categories:” are to be referenced in other parts of the program and in this design is referenced in the automation section. “pushbuttons” is the first and only category that is used and has two identifiers, “IGNORE” and “SOUNDALARM” which have titles associated with them that the user will see in their notifications. These both have authentication required so when it appears in the application, it won’t just enable anything without the user placing their fingerprint or their password to authenticate that it was them that issued an action [51].

notify:

- name: Rob\_Alert

platform: smtp

server: smtp.gmail.com

port: 587

sender: rob.bunzaga@gmail.com

starttls: 1

username: rob.bunzaga@gmail.com

password: RobDanielUA

recipient: rob.bunzaga@gmail.com

The “notify:” section is for the email notifications. This is where the password and the email is stored. Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission and is a form of encoding the message for security purposes. The sender is the email being used and the recipient is who gets the email. The username and password is associated with the sender email.

Next is the customize.yaml file. To access this file, type “nano customize.yaml” into the terminal.

binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor:

friendly\_name: Door Sensor

binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3:

friendly\_name: Window Sensor

sensor.front\_door:

icon: mdi:door

sensor.window:

icon: mdi:mdi:window-closed

sensor.aeotec\_zw096\_smart\_switch\_6\_voltage:

icon: mdi:flash-circle

friendly\_name: Smart Switch Voltage

The customize.yaml file is where the sensors are given a “friendly\_name” for the display and, if not given anywhere else, the sensors are given an icon. Here, each sensor is mentioned that is to be changed and below that are the changes made.

To access the Group file, type “nano groups.yaml” into the homeassistant terminal.

default\_view:

view: yes

icon: mdi:home

entities:

- group.Alarm

- group.Door

- group.Window

- group.Motion

- group.SmartSwitch

- group.DoorLock

The beginning of the groups.yaml file has “default\_view” which is the main page in the Home Assistant display. The icon is at the top of the page and is of the home. The “entities:” is what groups to include; these groups are found below and include the sensors in each group. The reason to do this is to make the groups of sensors and switches better visible on the display.

Alarm:

name: Alarm

entities:

- alarm\_control\_panel.alarm\_home

Door:

name: Door Sensor

entities:

- sensor.front\_door

Window:

name: Window Sensor

entities:

- sensor.window

DoorLock:

name: Door Lock

entities:

- lock.kwikset\_touchpad\_electronic\_deadbolt\_locked

SmartSwitch:

name: Smart Switch

entities:

- switch.aeotec\_zw096\_smart\_switch\_6\_switch

- sensor.aeotec\_zw096\_smart\_switch\_6\_voltage

Motion:

name: Motion Detector

entities:

- sensor.motion

- sensor.vision\_zp3102\_pir\_motion\_sensor\_temperature

Each group is mentioned above with the name the Home Assistant recognizes it as, for example “Door:” and is recognized as “group.Door”. The “name:” is the display name on the Home Assistant and is what is seen when using the app or the website. The “entities:” is what sensors are included in the group. Some of these sensors have been given a friendly name and are used and the rest are the Home Assistant names that were given when attached [51].

To access the Automations file, type “nano automations.yaml” into the terminal.

- id: action\_push\_message

alias: Door armed home notification

initial\_state: 'on'

trigger:

- entity\_id: sensor.front\_door

from: CLOSED

platform: state

to: OPEN

action:

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed stay

service: notify.ios\_daniels\_iphone

- data:

message: door is open. Armed Stay

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed stay

service: notify.ios\_robzaga

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_home

The first automation is related to the front door and has a condition of whether the alarm is “armed\_home” or not; this is located at the bottom. The initial state of the automation is set to ‘on’ and is triggered by the contact sensor connected to the front door and if it is opened. The action is, once again, connected to the front door and has a “push” notification associated with it. The category is “pushbuttons” which is mentioned in the configuration.yaml file. This is associated with the below code. This notification goes to three locations, “ios\_daniels\_iphone”, “ios\_robzaga” and has an email message sent to “rob\_alert”. The badge number is what matches on the iPhone. This reason to have this automation is to let the user know when the door is opened while at home and the system is armed [51].

- id: push\_notify\_action

alias: 'Push Notify Action'

initial\_state: 'on'

trigger:

- platform: event

event\_type: ios.notification\_action\_fired

event\_data:

actionName: IGNORE

action:

service: homeassistant.restart

The configuration.yaml file has two identifiers under the category “pushbuttons” and includes the above code. The “id:” is for reference and Alias is what will be displayed in the Home Assistant configuration settings. The trigger is whether the button is pressed on the notification on the iPhone and will act as “ios.notification\_action\_fired”. This means that that particular button was pressed and will do the following action. What is set currently is the Home Assistant will restart if the ignore is pressed.

- id: push\_notify\_action\_two

alias: 'Push Notify Action Two'

initial\_state: 'on'

trigger:

- platform: event

event\_type: ios.notification\_action\_fired

event\_data:

actionName: SOUNDALARM

action:

service: homeassistant.restart

The same code is used on the “SOUNDALARM” button being pressed, which is the second button that could be pressed.

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed away

service: notify.ios\_daniels\_iphone

- data:

message: window is open, Armed away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed away

service: notify.ios\_robzaga

alias: Window armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.window

from: CLOSED

platform: state

to: OPEN

This code segment is in relation to the Contact Sensor with the window and is used for when the system is “armed\_away”. This is to let the user know the window has been opened while the user is away.

- action:

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed away

service: notify.ios\_daniels\_iphone

- data:

message: door is open. Armed away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open

service: notify.ios\_robzaga

alias: Door armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.front\_door

from: CLOSED

platform: state

to: OPEN

The code segment above is for the front door when the alarm is “armed\_away” and is used to let the user know when the door is opened while they are away from the home. This is very similar to other sections of code and, to avoid redundancy, will not be explained.

- action:

- data:

data:

action\_data:

entity\_id: sensor.motion

push:

badge: 0

category: pushbuttons

message: Motion was detected. Armed away

service: notify.ios\_daniels\_iphone

- data:

message: Motion is detected. Armed Away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.motion

push:

badge: 0

category: pushbuttons

message: Motion was detected. Armed away

service: notify.ios\_robzaga

alias: Motion armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar

from: '0'

platform: state

to: '8'

The code segment above is similar to other code segments but is associated with the motion sensor. A big difference with this is the trigger. The motion sensor has different states than the contact sensors. The trigger will be when the motion sensor goes from state 0 to state 8. The motion sensor is only armed when the user arms the home as “armed\_away” because it wouldn’t make sense to arm the home with a motion sensor while still at home. The motion sensor does not have a “armed\_home” like the contact sensors.

- action:

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed\_home

service: notify.ios\_daniels\_iphone

- data:

message: window is open. Armed home

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed\_home

service: notify.ios\_robzaga

alias: Window armed\_home notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_home

trigger:

- entity\_id: sensor.window

from: CLOSED

platform: state

to: OPEN

This code segment is in relation to the contact sensor connected to the window. This is only triggered when the state is “OPEN” instead of “CLOSED” and when the alarm is set to “armed\_home”. Everything else has been mentioned in the first automation code segment.

The next file is the known\_devices.yaml file and has two devices set up: the two iPhones with their names. “Robzaga:” and “daniels\_iphone:” are the reference names that will be used in the files.

robzaga:

hide\_if\_away: false

name: robzaga

track: true

daniels\_iphone:

name: daniels\_iphone

track: true

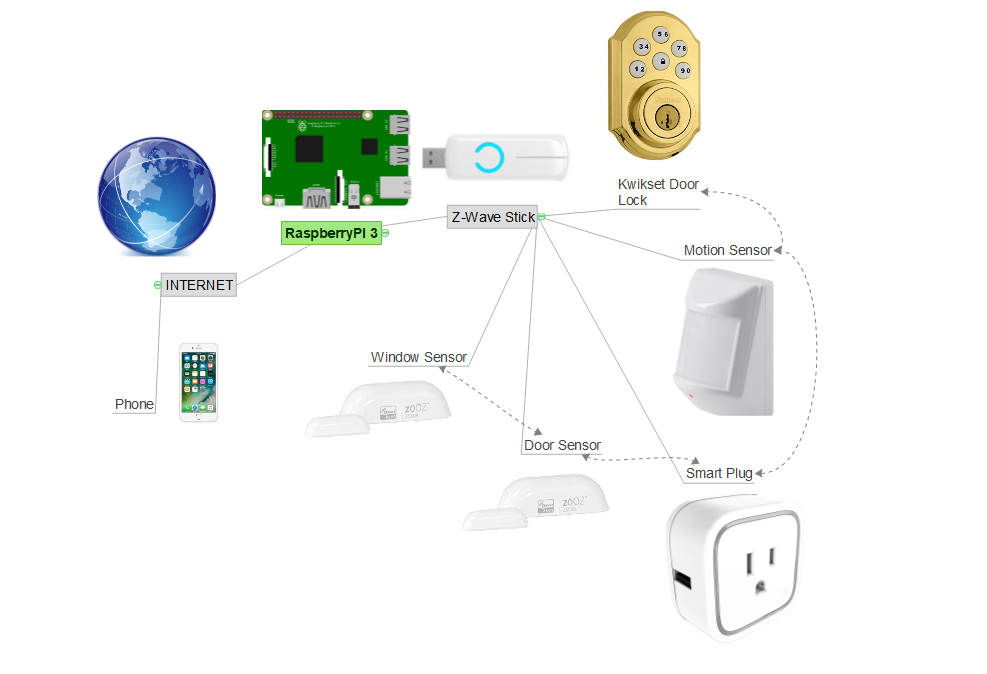
**Description of Design**

Figure 31: Schematic of Wireless Devices

Figure 31 is a representation of the wireless devices connected to the Z-Wave Stick and how the Raspberry Pi uses Wi-Fi to interact with the system and the iPhone. All of the devices have been added by the protocols listed above to the Z-Wave Stick.

****Figure 32: Schematic of Wired Devices

Figure 32 shows what components and devices are connected by a physical wire. The 808 Canz speaker is connected through USB to the Raspberry Pi for battery recharge and is connected by a 3.5mm audio cable. The camera is connected through USB as well as the Z-Wave Stick.

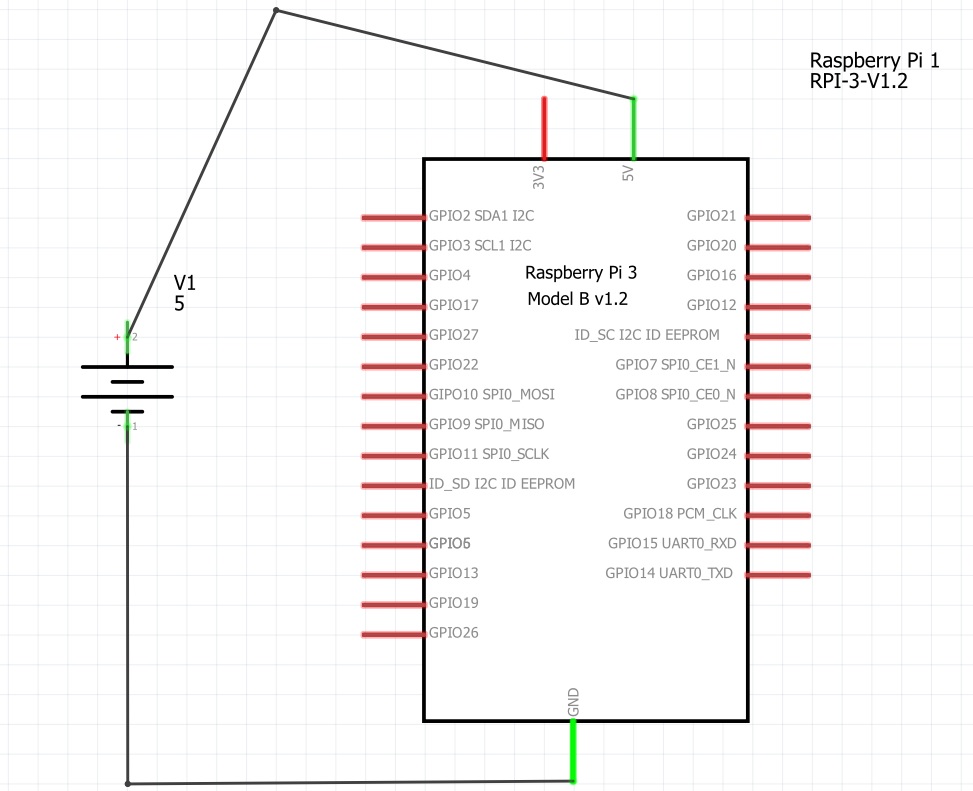
****Figure 33: Circuit Design of Raspberry Pi

Figure 33 is the circuit design of the Raspberry Pi. None of the GPIO pins are used but has 5 Volts going in to power and has a GND.

|  |  |  |
| --- | --- | --- |
| **Device** | **Cost** | **Paid** |
| Canz 808 bluetooth speaker | $19.99 | $0 |
| Brass Z-wave Kwikset Door lock | $100 - $200 | $0 |
| Raspberry Pi with Protective Case | $69.99 | $69.99 |
| Aeotec Smart Switch 6, remote control z-wave plus smart plug | $49.99 | $49.99 |
| 2 Zooz Z-wave plus D/W contact Sensors ZSE08 | $19.95 | $39.90 |
| Monoprice Z-wave plus PIR Motion detector | $24.99 | $24.99 |
| Cimkiz A871 USB Webcam with built in MIC | $11.85 | $11.85 |
| Aeotec Z-stick Gen 5, Z-wave plus USB to create gateway | $44.65 | $44.65 |
| Wood and accessories for model home | $30 | $30 |
| **Total** | **$397 - $497** | **$271.37** |

Figure 34: Cost of Project [61]

Figure 34 shows the prices of each device and component of the project. This project was worked on through research and also hardware application and implementation. The total amount of time that it took to finish this project was 201 hours.

**Measurement Methods**

The home security and automation is only a success if the devices and WSN can be accessed outside the home. This was hard to implement into the network because it not much was known about this process. Testing the DuckDNS process wasn’t hard and either worked or didn’t work. The obvious way to test is to turn off Wi-Fi on mobile device and connect to the Home Assistant. Everything works outside of the home network and functions properly.

**Project Constraints**

This project has a good amount of project constraints which some are talked about in the future work section below. The ability to record video from the Web Camera to access later is a huge constraint. Most of the sensors don’t allow for reading the microchips either because they are closed for good or they are blacked out with a marker or something similar. Sensors being made by another person is a constraint because it makes it so it’s not clear what’s exactly going on with the sensor. This project uses Home Assistant as the software; a constraint is that a custom software hasn’t been made. The camera is not connected to the main app and could possibly, with a lot of implementation, be a part of it. A big constraint, that is part of any security system, is that there needs to be power connected to the main hub. When there is no power, no sensors will be working with the hub and no security is working. Money is almost always a constraint but the price of the devices and sensors are more than a lot of others because of what they can do. Battery life is a constraint because the amount of batter life is limited per sensor and device. The Z-Stick has a rechargeable battery powered by the Raspberry Pi. Lastly, the distance to the Raspberry Pi to the sensors is a constraint because the longer the distance the less functionality of the sensors.

**Problems Encountered/What was Learned**

When the project was first started, there was a problem with a Raspberry Pi and was caused when there was an attempt to control the Raspberry Pi remotely and damaged the working condition of the Raspberry Pi. To fix this, the Micro SD card had to be reimaged with the original Operating System. This fixed the problem so the project could continue.

The Door Lock couldn’t connect for a long period of time giving other hardware more of a priority. The process of connecting the Door lock to the Z-Wave controller was hard to find but eventually, with the help of others connecting their devices and sharing their process, the Door Lock connected in a Secure way. This is because the Home Automation needed a network key and then could be connected in the Secure way, as described earlier.

During the implementation of the design of this project there was trouble with implementing notifications. Trying to find the necessary procedure for emailing and sending a notification to the iPhone seemed impossible on the Home Assistant website. Luckily, there have been others that have used the Home Assistant and have posted their success on the Home Assistant Blog. It is critical to have the code placed correctly in the YAML language and so it didn’t work for a couple weeks. A few hours were spent on trying to figure this out and finally could be part of the design; the solution was more complex than what was imagined.

Creating the Actionable Notifications was harder than the email because it includes an action with the notification on the iPhone app. This is because it sends a message with the notification and has an action associated with it. Two of the actions in the design are to “Sound the Alarm” and “Ignore” which are associated with the door and window sensors respectively. It took a bunch of hours of researching what others have done to figure this out because the information wasn’t clear enough on the Home Assistant website. What was learned from this process was that it’s important to research what is already out there and what has been done to help the project become a success. Many hours can be spent on research and is worth the time.

Accessing the Home Assistant away from home was a longer process and was harder to figure out at first. The project was mostly worked on in the Computer Engineering Lab at UVU and had access to the Wi-Fi. The reason the Home Assistant couldn’t be accessed away from the network is finding out the modem would have to be configured for port forwarding. To do this, a personal network would need to be used. After finding this, the process was found and implemented to allow access to the Home Assistant remotely and completely away.

**Student Experience**

The Senior Design Project was something that I had been dreading for some time so prior to coming into the class I had some ideas. None of them were realistic with the time frame; some were too easy and one would have taken a lot longer than the semester. I’m happy with working with a Wireless Sensor Network and integrating a Raspberry Pi into a project. Certain aspects were harder than others but the time I put into this project makes me proud of what this project has resulted in. The idea of making a Z-Wave network intrigued me because it’s what a lot of the “big” companies are using for their home security and home automation and is very current with their technology. The devices are expensive and I think you get what you paid for. The project made me stretch, especially with the amount of detail needed for this paper, and helped me to learn all the necessary parts of a project. I’m glad that I could be part of this.

**Conclusions/Recommendations**

After all that has been implemented and designed, this project’s finish is better than what was ideal in the project’s beginning. The results of everything working well and being able to control a lot of items while away from home is a success. The display is user friendly and contains only the elements that are important to this project. To do it all over again, it is recommended to consider making an app at the very beginning and not worry about the devices until later. Everything else was great and there was great success in this project.

**Future Work**

What needs to be added to this project and design is the ability to store video to an external hard drive; doing this will allow for video playback. If there is a break-in, the user, for now, can only view the live video without the ability of saving the video to show police later. That’s the evidence that someone broke in and could possibly have the identity of the person who broke in. Another way to do more future work on this project is to make homemade sensors. Transceivers and receivers would be required to do this and would be cheaper in the end. Making the Video Camera(s) more secure and able to be outside would benefit the user a lot more so there is peace at home. It would require a wireless signal to transport the video from the camera to the hub wirelessly. Lastly, working to make a web application and mobile application would benefit because then the display would be more custom and user friendly, especially because it would be specifically for the Z-Wave protocol.

**References**

[1] M. Assaf, R. Mootoo, S. Das, E. Petriu, V. Groza, and S. Bixwas, “Sensor Based Home Automation and Security System”, February 5, 2018

[2] H. Huang, S. Xiao, X. Meng, and Y. Xiong, “A Remote Home Security System Based on Wireless Sensor Network and GSM Technology”, February 5, 2018

[3] “Safer. Smarter. Z-Wave”, Z-Wave, Silicon Laboratories, <http://www.z-wave.com/>, February 5, 2018

[4] A. Kamain and Z. Zakaria, “A Review on ZigBee Security Enhancement in Smart Home Environment,” February 6, 2018

[5] M. Knight, “How Safe is Z-Wave?”, February 5, 2018

[6] K. Romer and F Mattern, “The Design Space of Wireless Sensor Networks”, February 4, 2018

[7] T. Hargreaves, C. Wilson, and R. Hauxwell-Baldwin, “Learning to Live in a Smart Home”, February 10, 2018

[8] W. Burr, “Selecting the Advanced Encryption Standard”, February 15, 2018

[9] Wikipedia, “IEEE 802.15.4”, February 20, 2018

[10] R. Krejci, O. Hujnak, and M. Svepes, “Security Survey of the IoT Wireless Protocols”, March 1, 2018

[11] C. Williams, “ZigBee Green Power”, March 16, 2018

[12] D. Chen, M. Wang, “A Home Security Network for Remote Monitoring Application”, February 5, 2018

[13] “Our History”, Z-Wave Alliance, Z-Wave Alliance, <https://z-wavealliance.org/z-wave_alliance_history/>, February 5, 2018

[14] “So you've got Smart Home questions?”, Silicon Laboratories, [www.Z-Wave.com/faq](http://www.z-wave.com/faq), February 5, 2018

[15] “Z-Wave Plus Certification”, Z-Wave Alliance, <https://z-wavealliance.org/z-wave_plus_certification/>, February 5, 2018

[16] “Z-Wave Specs”, Silicon Laboratories, <http://www.z-wave.com/search-product-manuals?query=z-wave+specs>, March 3, 2018

[17] “National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information”, National Institute of Standards and Technology, <http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf>, March 1, 2018

[18] “Cryptographic nonce”, Wikipedia <https://en.wikipedia.org/wiki/Cryptographic_nonce>, March 20, 2018

[19] Z-Wave, “Z-Wave Protocol Overview”, March 2, 2018

[20] M. Yassein, W. Mardini, A. Khalil, “Smart Homes Automation using Z-Wave Protocol”, February 5, 2018

[21] “Elliptic Curve Diffie Hellman”, Wikipedia, <https://wiki.openssl.org/index.php/Elliptic_Curve_Diffie_Hellman>, April 12, 2018

[22] “QR code”, Wikipedia, <https://en.wikipedia.org/wiki/QR_code>, April 12, 2018

[23] C. Badenhop, S. Graham, B. Ramsey, B. Mullins, L. Mailloux, “The Z-Wave routing protocol and its security implications”, April 13, 2018

[24] “BCM2837”, Raspberry Pi, <https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md>, April 2, 2018

[25] “Random-access memory”, Wikipedia, <https://en.wikipedia.org/wiki/Random-access_memory>, April 2, 2018

[26] “Ethernet”, TechTerms, <https://techterms.com/definition/ethernet>, April 2, 2018

[27] “microSD”, Wikipedia, https://simple.wikipedia.org/wiki/MicroSD, April 2, 2018

[28] “General-purpose input/output”, Wikipedia, <https://en.wikipedia.org/wiki/General-purpose_input/output>, April 15, 2018

[29] “HDMI”, HDMI, <https://www.hdmi.org/learningcenter/kb.aspx#1>, April 16, 2018

[30] “USB”, Wikipedia, <https://en.wikipedia.org/wiki/USB>, April 2, 2018

[31] “RASPBERRY PI 3 IS OUT NOW! SPECS, BENCHMARKS & MORE”, Raspberry Pi, <https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/>, April 2, 2018

[32] “Installing Operating System Images”, Raspberry Pi, <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>, April 15, 2018

[33] “Virtual Network Computing”, Wikipedia, <https://en.wikipedia.org/wiki/Virtual_Network_Computing>, April 15, 2018

[34] “VNC”, Raspberry Pi, <https://www.raspberrypi.org/documentation/remote-access/vnc/README.md>, April15, 2018

[35] “Developers”, Home Assistant, <https://www.home-assistant.io/developers/>, April 15, 2018

[36] “YAML”, Home Assistant, <https://www.home-assistant.io/docs/configuration/yaml/>, April 16, 2018

[37] “YAML 1.2”, YAML, <https://Yaml.org>, April 12, 2018

[38] “YAML Specification Index”, YAML, <http://www.yaml.org/spec/>, April 12, 2018

[39] “YAML”, Wikipedia, <https://en.wikipedia.org/wiki/YAML#cite_note-1.0-1>, April 12, 2018

[40] “Introducing JSON”, JSON, <https://www.json.org/>, April 12, 2018

[41] “Manual Installation on a Raspberry Pi”, Home Assistant, <https://www.home-assistant.io/docs/installation/raspberry-pi/>, April 15, 2018

[42] “Z-Wave”, Home Assistant, <https://www.home-assistant.io/docs/z-wave/installation>, April 15, 2018

[43] “The Smartest House Your Z-Wave Store”, The Smartest Home, <https://www.thesmartesthouse.com/products/aeon-labs-z-stick-gen5-z-wave-plus>, February 19, 2018

[44] “SD3502”, Keil, <http://www.keil.com/dd/docs/datashts/sigmadesigns/sd3502_bro.pdf>, April 19, 2018

[45] “Zooz Z-Wave Plus Contact Sensor ZSE08”, Zooz, <http://www.getzooz.com/zooz-zse08-contact-sensor.html>, February 19, 2018

[46] “Reed switches”, Chris Woodford, April 22, 2018 <http://www.explainthatstuff.com/howreedswitcheswork.html>, April 2, 2018

[47] “Z-Wave Plus PIR Motion Detector with Temperature Sensor”, Monoprice, <https://www.monoprice.com/product?p_id=15271>, February 19, 2018

[48] “PIC16F631”, Microchip, <http://ww1.microchip.com/downloads/en/DeviceDoc/40001262F.pdf>, April 19, 2018

[49] “Smart Switch 6 Technical Specifications”, Aeotec, <https://aeotec.com/z-wave-plug-in-switch/1477-technical-specifications.html>, April 12, 2018

[50] “SmartCode Electronic Deadbolt”, Kwikset, <https://www.kwikset.com/products/details/electronic-locks/909-l03-smt.aspx>, February 19, 2018

[51] “Awaken Your Home”, Home Assistant, <https://www.home-assistant.io/>, April 1, 2018

[52] “Z-Wave Devices – Adding and Removing”, Home Assistant, <https://www.home-assistant.io/docs/z-wave/adding>, April 12, 2018

[53] “Cimkiz A871 Web Camera,USB Webcam,Web cam Desktop camera With Built-in MIC for Video Calling and Recording on Skype/ FaceTime / YouTube / Hangouts / On Air”, Cimkiz, <http://cimkiz.com/product/product-93-669.html>, February 19, 2018

[54] “SP880”, 808 Audio, <http://www.808audio.com/portable-bluetooth-speakers/everyday-speakers/?sku=SP880>, April 26, 2018

[55] “Duck DNS”, Duck DNS, <https://www.duckdns.org/why.jsp>, April 12, 2018

[56] <https://www.duckdns.org/install.jsp>, April 12, 2018

[57] “install”, Duck DNS, <http://info.ssl.com/article.aspx?id=10241>, April 12, 2018

[58] “Remote Access with TLS/SSL via Let's Encrypt”, Home Assistant, <https://www.home-assistant.io/docs/ecosystem/certificates/lets_encrypt/>, April 15, 2018

[59] “The world’s best video conferences are built on Jitsi.”, Jitsi, <https://jitsi.org/what-is-jitsi/>, April 12, 2018

[60] “Jitsi Meet”, Jitsi, <https://jitsi.org/jitsi-meet/>, April 12, 2018

[61] “Amazon”, Amazon, <https://www.amazon.com/>, April 21, 2018

**Appendices**

**Appendix A: Original Proposal**

Wireless home security Network

Daniel Anderson

Roberto Unzaga

Afsaneh Minaie

Spring 2018

Abstract

With the Advancement of newer technology, the need to feel protected and have a more convenient lifestyle is growing. Technology is advancing at an alarming rate. Home security and automation devices has become increasingly popular. There are apps to let the homeowner know if someone has broken in or if they forgot to turn off a light. Many people would like to control automated devices from an app, or know how each sensor is doing. We will be creating a wireless sensor network for inside the home, using Z-wave. We will be connecting sensors in the home for security purposes. There will be a viewable camera and a controllable light module. Using an app, we will be able to view, control, and see the security system.

Description

Our project will let you connect to a Wireless Sensor Network through an app to let you know what’s going on in specific locations. To do this we will need a Raspberry Pi 3 Model B to connect to the various sensors we will have in a model home we will build. Each sensor will be connected using the Z-wave protocol to the Raspberry Pi. This will include a usb drive that will receive the Z-wave signals to the Raspberry Pi.

The Z-wave sensors we will have in our network are a motion detector to detect intruders in a certain location of the home, 2 magnetic sensors to detect if a window and door is opened and closed. In addition to the sensors we will have a camera to view a room and an outlet to be used for the light, that is to be left on. This outlet will be controlled so a person can turn it on and off by a push of a button on the app. The camera will be accessible by the user from the app. This will allow them to hear, see, and speak to anyone in the home. Home security would not be complete without a controllable door lock. We will be adding this to the front door.

As mentioned earlier, we will build a model home to secure all the components that will be included in the design of our WSN. This will be a single room with a motion detector in the corner. There will be a door and a window that opens. We will have a camera in another corner of the room. There will be a light connected to an outlet that will be controlled from an app. Also, we will have a door lock on the front door, to be controlled in person or by the app.

Components

1 Raspberry Pi 3 Model B

1 Brass Z-wave Kwikset Door lock

2 Zooz Z-wave plus D/W contact Sensors ZSE08

1 Monoprice Z-wave plus PIR Motion detector

1 Aeotec Smart Switch 6, remote control z-wave plus smart plug

1 Aeotec Z-stick Gen 5, Z-wave plus USB to create gateway

1 Cimkiz A871 USB Webcam with built in MIC

Wood

Cardboard (for mockup)

Glue and or small screws to secure the devices to the wall

Small Hinge for the door

Deliverables

We will be providing the source code for the raspberry pi, for the connection of the sensors and the camera, as well as the app. There will be a raspberry pi with several sensors connected to it. The sensors will be inside of a model home for display. We will be meeting with the professor to discuss our progress each week on Tuesday at 11:30am each week. We will provide a journal that we record our work done each week. We will also provide a written weekly report. At the end of the semester, we will provide a typed final project report and oral presentation.

Work Load

Daniel

Connection, code and testing for the Wireless door, wireless window, and Door lock. Camera testing. Creating and implementing the app for connectivity. Building the little model home.

Rob

Connection, code and testing for the wireless motion, controllable outlet plug, and Camera connection. Help connecting the devices to the app. Supplying wood and securing the devices.

Schedule

Week 1: Finding an idea for a project, researching on how to complete it and submitting a proposal.

Week 2: Consultation with Advisor about project and further research for resources. Order components. Start on research and writing design project final report.

Week 3: Finish obtaining the necessary equipment and learn how each part works. Begin coding for the raspberry pi and start to connect.

Week 4: Connect the door and window sensors individually for demonstration and implement into raspberry pi.

Week 5: Testing the door and window. Connect the motion sensor and door lock for demonstration and implement into raspberry pi.

Week 6: Testing the motion and door lock with the other sensors. Connect the smart plug and start to begin coding for the camera.

Week 7: Test the smart plug with the other sensors. Continue the connection and code for the camera. Start creation of the app.

Week 8: Creating the app and model home. Testing camera.

Week 9: Connect the sensors to app and model home.

Week 10: Testing the sensors in the model home. Reprogramming any bugs.

Week 11: Testing the sensors with the app. Reprogramming any bugs.

Week 12: Begin writing the research report. Look for research sources.

Week 13: Finish writing the research report and create the oral presentation.

Week 14: Review and submit the written project report to the advisor.

Week 15: 45-min presentation of the project.

**Appendix B: Source Code**

------configuration.yaml------

homeassistant:

# Name of the location where Home Assistant is running

name: UVU HOME

# metric for Metric, imperial for Imperial

unit\_system: imperial

# Pick yours from here: http://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones

time\_zone: America/Denver

# Customization file

customize: !include customize.yaml

temperature\_unit: F

# Customization file for name changes and such

customize: !include customize.yaml

# customization file for the groups and tabs

group: !include groups.yaml

#including the automation file to the main configuration file

#this includes the notifications

automation: !include automations.yaml

#A template for the front door sensor to show if open or closed

sensor:

- platform: template

sensors:

front\_door:

friendly\_name: Front Door

entity\_id:

- binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor

value\_template: >-

{%- if is\_state("binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor", "on") -%}

OPEN

{%- else -%}

CLOSED

{%- endif -%}

icon\_template: >

{% if is\_state('binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor', "on") %}

mdi:door-open

{% else %}

mdi:door-closed

{% endif %}

window:

friendly\_name: Window

entity\_id:

- binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3

value\_template: >-

{%- if is\_state("binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3", "on") -%}

OPEN

{%- else -%}

CLOSED

{%- endif -%}

icon\_template: >

{% if is\_state('binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3', 'on') %}

mdi:mdi:window-open

{% else %}

mdi:mdi:window-closed

{% endif %}

motion:

friendly\_name: Motion

entity\_id:

- sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar

value\_template: >-

{%- if is\_state("sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar", "8") -%}

DETECTED

{%- else -%}

READY

{%- endif -%}

#The alarm system. To manually set the alarm stay, away, or disarm

#used for the notifications to work right

alarm\_control\_panel:

- platform: manual

name: Alarm Home

code: 1111

pending\_time: 0

trigger\_time: 0

armed\_away:

trigger\_time: 0

pending\_time: 15

armed\_home:

pending\_time: 0

trigger\_time: 0

# Enable frontend and creates new color themes

frontend:

themes:

day:

primary-color: green

night:

primary-color: red

# Enables configuration UI

config:

http:

base\_url: https://URL:8123

api\_password: password

ssl\_certificate: location of the certificate

ssl\_key: location of the certificate

# Discover some devices automatically

discovery:

#for certain devices to ignore when discovering all

ignore:

- roku

- samsung\_tv

- netgear\_router

- apple\_tv

zwave:

usb\_path: /dev/ttyACM0

notify:

- name: Rob\_Alert

platform: smtp

server: smtp.gmail.com

port: 587

sender: rob.bunzaga@gmail.com

starttls: 1

username: rob.bunzaga@gmail.com

password: RobDanielUA

recipient: rob.bunzaga@gmail.com

ios:

push:

categories:

- name: pushbuttons

identifier: 'pushbuttons'

actions:

- identifier: 'IGNORE'

title: 'Ignore'

activationMode: 'foreground'

authenticationRequired: yes

- identifier: 'SOUNDALARM'

title: 'Sound the Alarm!'

activationMode: 'foreground'

authenticationRequired: yes

-----customize.yaml-----

binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor:

friendly\_name: Door Sensor

binary\_sensor.zooz\_zse08\_contact\_sensor\_sensor\_3:

friendly\_name: Window Sensor

sensor.front\_door:

icon: mdi:door

sensor.window:

icon: mdi:mdi:window-closed

sensor.aeotec\_zw096\_smart\_switch\_6\_voltage:

icon: mdi:flash-circle

friendly\_name: Smart Switch Voltage

-----groups.yaml-----

#This is the main display for the home automation and alarm system

default\_view:

view: yes

icon: mdi:home

#what groups to show in the Home assistant

entities:

- group.Alarm

- group.Door

- group.Window

- group.Motion

- group.SmartSwitch

- group.DoorLock

#groups for the arming and disarming

Alarm:

name: Alarm

entities:

- alarm\_control\_panel.alarm\_home

#the front door group

Door:

name: Door Sensor

entities:

- sensor.front\_door

#the window group

Window:

name: Window Sensor

entities:

- sensor.window

#the door lock group

DoorLock:

name: Door Lock

entities:

- lock.kwikset\_touchpad\_electronic\_deadbolt\_locked

#the smart switch group

SmartSwitch:

name: Smart Switch

entities:

- switch.aeotec\_zw096\_smart\_switch\_6\_switch

- sensor.aeotec\_zw096\_smart\_switch\_6\_voltage

#the motion sensor group

Motion:

name: Motion Detector

entities:

- sensor.motion

- sensor.vision\_zp3102\_pir\_motion\_sensor\_temperature

----automations.yaml-----

- id: action\_push\_message

alias: Door armed home notification

initial\_state: 'on'

trigger:

- entity\_id: sensor.front\_door

from: CLOSED

platform: state

to: OPEN

action:

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed stay

service: notify.ios\_daniels\_iphone

- data:

message: door is open. Armed Stay

service: notify.rob\_alert

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed stay

service: notify.ios\_robzaga

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_home

####What happens when you push Ignore

- id: push\_notify\_action

alias: 'Push Notify Action'

initial\_state: 'on'

trigger:

- platform: event

event\_type: ios.notification\_action\_fired

event\_data:

actionName: IGNORE

action:

service: homeassistant.restart

#####What happens when you push Sound the Alarm

- id: push\_notify\_action\_two

alias: 'Push Notify Action Two'

initial\_state: 'on'

trigger:

- platform: event

event\_type: ios.notification\_action\_fired

event\_data:

actionName: SOUNDALARM

action:

service: homeassistant.restart

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed away

service: notify.ios\_daniels\_iphone

- data:

message: window is open, Armed away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed away

service: notify.ios\_robzaga

alias: Window armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.window

from: CLOSED

platform: state

to: OPEN

- action:

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open, armed away

service: notify.ios\_daniels\_iphone

- data:

message: door is open. Armed away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.front\_door

push:

badge: 0

category: pushbuttons

message: door is open

service: notify.ios\_robzaga

alias: Door armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.front\_door

from: CLOSED

platform: state

to: OPEN

- action:

- data:

data:

action\_data:

entity\_id: sensor.motion

push:

badge: 0

category: pushbuttons

message: Motion was detected. Armed away

service: notify.ios\_daniels\_iphone

- data:

message: Motion is detected. Armed Away

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.motion

push:

badge: 0

category: pushbuttons

message: Motion was detected. Armed away

service: notify.ios\_robzaga

alias: Motion armed away notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_away

trigger:

- entity\_id: sensor.vision\_zp3102\_pir\_motion\_sensor\_burglar

from: '0'

platform: state

to: '8'

- action:

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed\_home

service: notify.ios\_daniels\_iphone

- data:

message: window is open. Armed home

service: notify.dan\_alert

- data:

data:

action\_data:

entity\_id: sensor.window

push:

badge: 0

category: pushbuttons

message: window is open, armed\_home

service: notify.ios\_robzaga

alias: Window armed\_home notification

condition:

- condition: state

entity\_id: alarm\_control\_panel.alarm\_home

state: armed\_home

trigger:

- entity\_id: sensor.window

from: CLOSED

platform: state

to: OPEN

------known\_devices.yaml------

robzaga:

hide\_if\_away: false

name: robzaga

track: true

daniels\_iphone:

name: daniels\_iphone

track: true

**Appendix C: Slideshow**