Home Automation with Raspberry Pi

Kevin Kuo
Department of Computer & Information Sciences
College of Science & Mathematics
Towson University
Kkuo1@students.towson.edu

Advisor: Wei Yu Ph.D.
Department of Computer & Information Sciences
College of Science & Mathematics
Towson University
WYu@towson.edu

ABSTRACT
Smart homes are becoming more popular and prevalent as they seek to improve the quality of life of home owners. There are a variety of ecosystems to choose from currently to include Google Home, Amazon's Alexa, etc. Hardware manufacturers have even designed and marketed "smart home hubs" with the express purpose of integrating various difference competing protocols and single stop smart home solution. The goal of this project is to design and integrate a Raspberry Pi as a "smart home" device and see how a small use case be applied to larger home automated networks.

Categories and Subject Descriptors
J.0 **[Computer Applications]** Computer Applications – *General*

Keywords
Home Automation; Smart Home; Wifi; ZigBee; Raspberry Pi; Z-Wave; Z-Wave Plus; LED; Light Sensor; Python; GPIO; Rolling average; Breadboard; Light Circuits; Python GPIO; Networking; Pandas; Numpy; Leviton; ZWay; Rolling; Black Hole; Key exchange

1. INTRODUCTION
Home automation has become very popular among homeowners and is being driven by Internet of Things environment and culture. Home automation may include automating everyday routine activities such as turning on lights, reading the weather report, or opening the locks on exterior doors. When home automation was in its infancy, its hardware, software, and integration was segmented and did not interoperate well enough for a wide uptake rate. Today, home automation is more affordable and "smart" - with better interoperability between devices, accessories, and protocols. This project will seek to implement home accessories such as lighting using affordable hardware such as a Raspberry Pi based on user and environmental inputs.

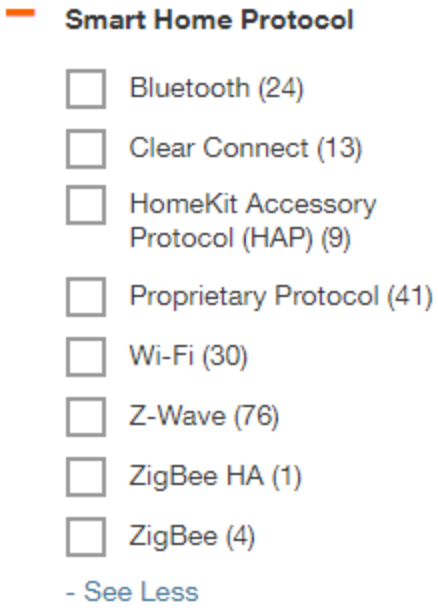
2. PLATFORM SELECTION AND CONFIGURATION

2.1 Protocol Selection
The three main competing protocols for wireless home devices are WiFi, ZigBee, and ZWave (ZWave Plus).

2.1.1 Wifi
WiFi is an increasingly popular option with the ecosystem products such as Google Home or Amazon Alexa. This is convenient for an end user that prefers to stay within those ecosystems and control Google Home or Alexa compatible devices from a mobile app on a smartphone. Unfortunately, this system is not as open source friendly and customizable as the two other popular platforms. Outside of the Google Home and Amazon Alexa ecosystems, there are plenty of smart home integrators that have developed proprietary WiFi protocols. These have the unfortunate effect of locking a homeowner into what may become a less widely supported and updated ecosystem in the future. WiFi dependent devices also tend to consume more electricity than other similarly functioning wireless protocols such as ZigBee or ZWave Plus.

2.1.2 ZigBee
ZigBee is an open standard run by the Zigbee Alliance. ZigBee is designed to be a "mesh" network and operates at the 900Mhz and 2.4Ghz frequency range. ZigBee has a linear range as low as 10 meters. ZigBee is broken into multiple sub-protocols and they do not necessarily interoperate with each other well enough.

2.1.3 ZWave (ZWave Plus)
ZWave is also an open standard run by SiliconLabs . ZWave is designed to be a "mesh" network as well and operates in the 900Mhz range in the United States. ZWave has since been updated to the ZWave Plus protocol which improves battery consumption, distance to 200 feet, streamlines the device inclusion process into the ZWave network, and provides additional device diagnostic capacities.



Home Depot Product Filter By Smart Home Protocol

2.1.4 Selection Decision

The platform selection process was a choice between ZigBee and ZWave. When browsing Home Depot's website's Smart Switch and Dimmer section, there were 76 ZWave options, 5 ZigBee options, and 30 WiFi options. In addition, ZWave had more open source support with source code for web based management applications and controllers available on GitHub.

2.2 Z Wave Background

2.2.1 Pairing

The pairing process, or the process by which devices enter and leave the network, is similar to the way Bluetooth devices are paired. A person wanting to add a device to the network puts both the Z Wave controller and the device in pairing mode. When the device is in pairing mode, the controller recognizes that and allows the device to be paired. This is called the inclusion process and is usually initiated by the press of a button on the device. The exclusion process is the opposite in which a device is removed from the control of a Z Wave controller.

2.2.2 Device Identification

Z Wave devices are typically identified by a 4 digit home ID which is the controller to which the device is paired with. The device ID, or node ID, is a byte value assigned by the controller to the device during the pair process. The controller starts by assigning the value of "1" to the first device paired to the controller. All subsequent devices are assigned a value in ascending order.

2.2.3 Command Classes

Command classes are associated with the application layer protocol of Z Wave which is well defined in the OpenZWave library. Each Z Wave devices supports many command classes but not necessarily all. For example, a binary Z Wave light switch may not support the particular command classes that a dimmer Z Wave light switch would use. Unsupported command classes set to a Z Wave device are often ignored.

2.3 Hardware

2.3.1 Raspberry Pi 3 Model B+

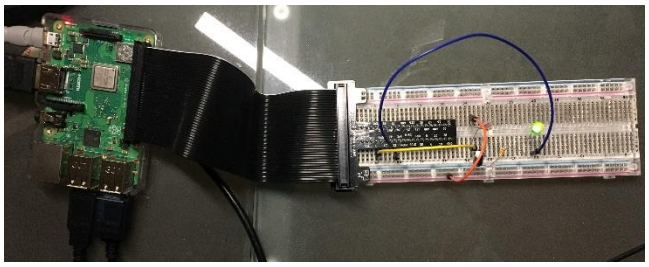
Raspberry Pi 3 Model B+ was the newest available Raspberry at the time. The Raspberry Pi 3 Model B+ improved upon processing power and WiFi connectivity through a more capable network adapter which supported the 5GHz frequency. There was a concern that the open source information regarding the newest Raspberry Pi model would not be as complete and cause problems with configuration. Fortunately, the Raspberry Pi 3 Model B+ was an incremental upgrade and did not pose any significant challenges other than the 'wiringpi' module needing to be updated from version 2.44 to version 2.46 in order to support the GPIO.readall command for the new Raspberry. Once the pins were labeled and routed correctly, the Raspberry Pi 3 Model B+ was completely compatible with the previous Raspberry Pi 3 Model B schematic.

2.3.2 LED

The 5mm LED diode was part of an MCM Raspberry Pi Project Package for an older Raspberry Pi 3 Model B. It was used in the early stages of this project to test the functionality of the GPIO pins as well as the breadboard. It was a simple binary output that could be used to test the Python GPIO library functionality.

2.3.3 Light Sensor

The light sensor that was used was part of an MCM Raspberry Pi Project Package for an older Raspberry Pi 3 Model B. It was re-used as an input sensor in this project in conjunction with a 50V 1uF electrolytic capacitor. The capacitor allows the user to measure the resistance to the light sensor.



Project Raspberry Pi 3 Model B+ and Breadboard Setup with LED and light sensor.

2.3.4 ZWave Controller Transceiver

There were two main options as far as ZWave Controllers for the Raspberry Pi 3 Model B+.

ZWave GPIO Daughter Card

2.3.4.1 ZWave GPIO Daughter Card

The first option was the ZWave GPIO daughter card. This hardware utilizes the GPIO pins of the Raspberry Pi and physically sits on top of the Raspberry Pi. Unfortunately, due to the physical location of the ZWave GPIO daughter card, it would have made the GPIO pins unusable and inaccessible for the light sensor and LED.

ZWave Controller USB Dongle

2.3.4.2 ZWave Controller USB Dongle

The second option for the ZWave controller transceiver was a USB based dongle. This is the smallest Z-Wave USB stick currently on the market and it is using a Silicon Labs Z-Wave Serial API. The USB model requires additional configuration as far as mounting the USB dongle. It also required purchasing a license key to be used with the ZWay management software. Such a license key was not required for the ZWave GPIO daughter card, but would necessitate purchasing a secondary Raspberry Pi just to utilize the GPIO pins. Using a secondary Raspberry Pi introduced the potential for network connectivity issues which depends heavily on the network configuration of the home/office which was out of the scope for this project.

Leviton Plug-In Outlet Switch

2.3.5 ZWave Plug-In Switch

The Leviton Plug-In Outlet with Z-Wave Technology was used. It was designed to be compatible with a Z-Wave enabled network. It can control electronic ballasts, CFLs and LED lights. We have a simple outlet circuit tester to verify the output of the switch.

2.4 Software

2.4.1 Z-Way

Z-Way is a complete cross platform smart home controller software. It is open source and designed to be run on hardware supporting the Raspbian OS, Windows, Ubuntu, etc. It has been tested against 500 physical devices for interoperability and continues to be supported by the ZWave community with ongoing software updates. Z-Way runs in the background of the Raspbian OS, but has a web browser accessible user interface. Z-Way can be access remotely by any other machine on the same network subnet. The most enthusiast friendly portion of the Z-Way software is that it implements Z-Wave controller functions in a closed source implementation, but has a robust Z-Way API to allow customization and commands known as "command classes" to be sent via HTTP GET requests.

2.4.2 Python 2.7.13

This project predominantly used the Python GPIO library to control both the GPIO connected devices as well as pass commands to the open source ZWay management service. While the project could have been written in C and/or C++, Python provided a convenient yet powerful higher level

scripting language to achieve the project object which was to integrate a Raspberry Pi as a home automatic device. If the project required sensors or inputs that are machine timing specific, C and/or C++ may have been the more appropriate system language. However, the commands to the ZWave network would still have been subject to wireless network latency.

2.4.3 Additional Python Libraries

Python libraries pandas and numpy were utilized to implement rolling averages. Rolling averages were useful to "smooth" out light sensor readings which would cause the ZWave switch to turn on and off. The polling and refresh rate of the Raspberry Pi and light sensor could be far too fast for the ZWave network to receive and process commands. Rolling averages reduced the chances of any erroneous or outlier measurements from trigger the ZWave switch alone.

3. RELATED WORK

3.1 Z Wave Network Layer

The specifications of the physical (PHY), Medium Access Control (MAC), and application layer are publicly, but few details for the network layer are public. The network layer is responsible for how the Z Wave protocol uses a static source routing and how it determines the routing path for the home station and nodes. Routes are determined from a centralized routing table and potentially forwarded from one node to another. Currently, an open source implementation of Z Wave routing does not exist. The Z Wave implementation of a controller exists, but the routing logic resides in the firmware of the Z Wave transceiver. This was determined by reverse engineering a static controller library. Several network topologies were part of the experiment and the forwarding and topology mechanisms of the Z Wave routing protocol were reverse engineering used passing and active observations of the Z Wave test setup.

3.2 Security Concerns

3.2.1 Key Exchange Exploit

Fouladi and Ghanoun analyzed the Z Wave proprietary protocol, specifically its encryption, authentication and key exchange protocols. They were able to develop a relatively inexpensive Z Wave packet interception and injection tool that highlighted a particular vulnerability in which they took full control of a Z Wave door lock. They were able to do this only knowing the home and node IDs of the target device which was deduced by observing Z Wave network traffic over a short period of time. This vulnerability was not due to a Z Wave protocol issue, but rather an implementation error in disabling the temporary key after the initial network key exchange during the inclusion process.

3.2.2 Black Hole Attack

Security research into the Z Wave protocol and the relationship between routing nodes and controllers reveals that the Z Wave network can be exploited by a malicious actor. A malicious actor can not only pretend to be other nodes in a particular Z Wave network, but it can also change the routing topology and manipulate data in transit. A Black Hole attack, where frames are maliciously discarded, can be used to prevent devices from receiving command classes from the controller as well as the controller from receiving sensor reports from the devices. The Z Wave protocol is susceptible to integrity based attacks.

3.2.3 Rogue Controllers

With the growing popularity of smart home automation to include Z Wave networks, vendors are introducing many new gateway devices. Unfortunately, the end users are unaware of the lack of security in these gateway devices. Malicious actors can gain unauthorized access to these gateway devices and take control of Z Wave devices which may pose a serious threat to homes, offices, etc.

4. PROJECT STAGES

4.1 GPIO Breadboard and LED

This stage involved the basic GPIO breadboard setup. The objective in this stage was to write simple Python code that would turn a LED diode on and off in a binary fashion. The first challenge was understanding the GPIO pins. The Raspberry Pi 3 Model B+ was new at the time of manufacturer and the 'wiringpi' module needing to be updated from version 2.44 to version 2.46 in order to support the GPIO.readall command for the new Raspberry. Once it was updated, the pins were labeled correctly.

The second challenge was caused by not understanding the design of the breadboard correctly. The breadboard included in the Raspberry Pi kit was actually an "extended" breadboard and, therefore, two separate breadboard circuits. The LED diode did not light up even with functioning code and this was determined by using a voltmeter to measure the voltage and current through the actual GPIO pins on the Raspberry Pi. Moving one of the resistors for the LED diode circuit closer to the adapter resulted in the LED diode lighting up.

4.2 Light Sensor

The objective of this stage involved setting up the light sensor/capacitor and reading the light sensor input. The capacitor allows the user to measure the resistance to the light sensor. The lower the reading from the sensor, the brighter the light. The higher the reading from the sensor, the darker the surroundings were. Once the basic pin layout was understood, this stage was less challenging than other stages. The light sensor was testing by covering it momentarily and observing the change in the light sensor reading.

4.3 ZWave Environment Setup

The objective of this stage involved setting up a limited ZWave environment as it would be commonly used in a household environment. That entails setting up the ZWay software on the Raspberry Pi, mounting and configuring the ZWave Controller USB dongle, and the inclusion process for the Leviton ZWave Plug-in Outlet Switch. This stage did not involve any modification to project source code.

4.3.1 ZWay Software

Setting up the ZWay software on the Raspberry Pi was relatively easy as you could install it using thru the typical "sudo apt-get install" functionality. It's accessible via the web browser using the URL: http://[your_ip]:8083.

4.3.2 ZWave Controller USB Dongle

However, plugging in the ZWave Controller USB dongle wasn't nearly as automatic. We needed to verify the ZWave Controller USB dongle was present in a USB port using "lsusb" and then specifying that the device was device "ttyACM0" in the "Z Wave Network Access Page" of the ZWay software. The ZWave Controller USB dongle also supported many different frequencies in the 900Mhz range; each of the frequencies corresponded to a different region in the world. A script was run to make it transceiver on the United States frequency of 908 MHz.

4.3.3 Leviton ZWave Plug-in Outlet Switch

The addition and inclusion of the Leviton ZWave Plug-in Outlet Switch was the most challenging of this stage. Inclusion kept failing in the beginning and the web user interface did not explain why other than a time out error. ZWay does write to a log file and when we tailed that log file, it indicated that with the USB dongle, the maximum number of devices have been reached which was just one device. A simple query online indicated that this was because a license for ZWay is required when using ZWave Controller USB dongle. After the license was purchased for approximately $30, ZWave immediately recognized the Leviton Plug-In Outlet and the inclusion process worked seamlessly.

4.4 Testing Leviton Plug-In Outlet Activation/Deactivation

The objective of this stage involved writing short Python code to turn on and off the Leviton Plug-In Outlet switch. This involved sending an open URL request to the ZWay software running in the background of the Raspberry Pi to either turn on or off the Leviton Plug-In Outlet switch. Formatting the URL string was the most time consuming part of this stage as the return value for the URL request was not as helpful for syntax errors in this case. It is interesting to note that you do not need to run the ZWay web browser in order to issue ZWay API supported Command Classes. One interesting thing to note is that the manual Python and HTTP requests which changed the status of the switch were reflected in the status of the switch in the web based user interface.

```
pi@raspberrypi:~ $

top - 11:15:16 up 13:07,  1 user,  load average: 0.13, 0.20, 0.13
Tasks: 125 total,   1 running,  78 sleeping,   0 stopped,   1 zombie
%Cpu(s):  2.3 us,  0.8 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   949448 total,   522524 free,   115560 used,   311364 buff/cache
KiB Swap:   102396 total,   102396 free,        0 used.   768880 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  458 root      20   0  143072  48880  24424 S   4.3  5.1   0:50.36 Xorg
 1693 pi        20   0   47392  19460  16804 S   3.9  2.0   0:01.02 lxterminal
  336 root      20   0  115736  33256  16648 S   2.0  3.5  10:21.86 z-way-server
 1705 pi        20   0    8244   3288   2832 R   1.0  0.3   0:00.24 top
  639 pi        20   0  143480  27448  20248 S   0.7  2.9   2:09.83 lxpanel
```

ZWay Running in the Background

```
[2018-12-13 21:41:25.233] [I] [zway] Waiting for job reply: SwitchBinary Get
[2018-12-13 21:41:25.249] [D] [zway] RECEIVED: ( 01 09 00 04 00 02 03 25 03 00 D5 )
[2018-12-13 21:41:25.249] [D] [zway] SENT ACK
[2018-12-13 21:41:25.249] [D] [zway] SETDATA devices.2.data.lastReceived = 0 (0x00000000)
[2018-12-13 21:41:25.249] [D] [zway] Received reply on job (SwitchBinary Get)
[2018-12-13 21:41:25.249] [D] [zway] SETDATA devices.2.instances.0.commandClasses.37.data.level = False
```

ZWave Log File for Turning On and Off Switch

4.5 Moving/Rolling Averages

The objective of this stage involved implementing a rolling window for light sensor readings. The purpose of this was to smooth out the imprecise sensor readings and handle the fast polling/refresh rates for the light sensor. The ZWay network may not be able to activate and deactivate a switch at the same rate as the polling rate of the light sensor. This became more problematic as the ZWave Network devices were located further and further from the ZWave Controller USB Dongle. Also, we did not want specific outlier or erroneous readings of the light sensor to drastically influence the activation and deactivation of the switch. The major challenge of this stage was, ultimately, converting the value of the DataFrame object type from the pandas/numpy object to an integer for comparison.

4.6 Integrating

The objective of this stage involved integrating all the previous stages into the ability to activate a ZWave outlet switch based on the light in an experimental environment. The previous stages made this stage easy as testing/integration occurred along the way. This stage satisfied the project objective.

5. FUTURE WORK/CONSIDERATIONS

The current project is being hosted and version controlled at https://github.com/Polo08816/HomeAutomation. This repository includes source code, documentation, and development notes which may be useful in future development.

5.1 ZWave Sensor Input and GPIO output

This project involved taking a GPIO sensor input and effecting a ZWave device output. Future work should involve taking a ZWave sensor input and effecting a GPIO accessory's output. The ZWay software's Expert User Inferface "interviews" the ZWave device and lists all ZWay API calls that the device supports which makes future development more straightforward and less proprietary. Logic could be written in AJAX, JSON, C/C++, and various other supported languages.

5.2 Cost

Market options for the ZWave ecosystem are substantial. The cost for the Raspberry Pi kit was $80, the ZWave plug in outlet switch was $35, and the ZWave Controller USB dongle was $30. There are more polished products made by reputable vendors such as Samsung, etc. that may fit the needs of a consumer (not prosumer or your typical enthusiast) at a substantially better cost. The strength in using the configuration that was used for this project is customizability.

5.3 Custom Sensors

The strength of this project shines if the user or project manager needs to integrate a custom sensor at the circuit level. The Raspberry Pi's GPIO interface is an extremely powerful and flexible interface that is suited for rapid experimentation.

6. CONCLUSIONS

This project achieved its objective of designing and integrating a Raspberry Pi as a "smart home" device and see how a small use case be applied to larger home automated networks. The particular application here was to implement a light sensor which would turn on and off a lamp plugged into a ZWave plug in outlet switch. There are countless applications for this ecosystem that can be developed in the future to provide more granular control over a smart home system.

7. REFERENCES

[1] Docs.python.org. (2017). About these documents – Python 3.6.4rc1 documentation. [Online] Available at: https://docs.python.org/3/ [Accessed 07-Dec-2017]

[2] RPi.GPIO 0.6.3 : Python Package Index. Pypi.python.org. (2017). [Online] Available at: https://pypi.python.org/pypi/RPi.GPIO. [Accessed: 07-Dec-2017]

[3] Croston, Ben. "raspberry-gpio-python / Wiki / Home". Sourceforge.net. (2017) [Online]. Available at https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/. [Accessed: 07-Dec-2017]

[4] Johansen, N. Z-Wave Networking Basics. Sigma Desings. (2017). [Online] Available at: htt[://zwavepublic.com/sites/default/files/APL13031-2%20-%20-Z-Wave%20Networking%20Basics.pdf. [Accessed: 07-Dec-2017]

[5] "Using light sensor module with Raspberry Pi | UUGear". Uugear.com. (2017). [Online] Available at: http://www.uugear.com/portfolio/using-light-sensor-module-with-raspberry-pi/. [Accessed: 07-Dec-2017]

[6] "On/Off Project – Switch a light on/off using your smart phone – PrivateEeyePi Project". Projects.privateeyepi.com. (2017). [Online] Available at: http://projects.privateeyepi.com/home/on-off-project. [Accessed: 07-Dec-2017]

[7] Z-Wave>Me UZB. Z-Wave.Me. (2017). [Online] Available at: https://z-wave.me/uzb/ [Accessed 17-Oct-2018]

[8] Z-Way Manual: Z-Way Essentials. Z-Wave.Me. (2017). [Online] Available at: https://z-wave.me/essentials [Accessed 17-Oct-2018]

[9] Christopher Badenhop, Scott Graham, Benjamin Ramsey, Barry Mullins, and Logan Mailloux. The Z-Wave routing protocol and its security implications. Computers & Security, Vol. 68. Air Force Institute of Technology. https://www.sciencedirect.com/science/article/pii/S0167404817300792 [Accessed 17-Dec-2018]

[10] Fehrang Fouladi and Sahand Ghanoun. Security Evaluation of the Z-Wave Wireless Protocol. Blackhat USA, 2013. https://pdfs.semanticscholar.org/10e1/21b903366ea81b94ca0c2e61c095cc087695.pdf [Accessed 17-Dec-2018]

[11] Jonathan Fuller and Benjamin Ramsey. Rogue Z-Wave controllers: A persistent attack channel. 2015 IEEE 40th Local Computer Networks Conference Workshops. https://ieeexplore.ieee.org/abstract/document/7365922/authors#authors [Accessed 17-Dec-2018]