Introduction

Home automation has become very popular among homeowners and is an important smart system driven by Internet of Things. 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. The purpose of this project is to design and integrate a Raspberry Pi as a “smart home” device and see how a small use case can be applied to larger home automated networks.

Platform Selection

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

WiFi

WiFi is an increasingly popular option with the ecosystem products such as Google Home or Amazon Alexa. This is a convenient for an end user that prefers to stay within those ecosystems and control Google Home or Alexa compatible devices from a mobile app. Unfortunately, this system is not as open source friendly and customizable as the other two platforms.

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.

ZWave (ZWave Plus)

ZWave is also an open standard run but 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.

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, 4 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.

Hardware

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 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.

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.

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.

ZWave Controller Transceiver

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

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.

The second option for the ZWave controller transceiver was a USB 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.

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.

Software

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 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 to be sent via HTTP GET requests.

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.

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.

Project Stages

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 binary LED diode on and off. 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 update, the pins were labeled correctly.

The second challenge manifested in not understanding the design of the breadboard correctly. The breadboard included in the Raspberry Pi kit was an “extended” breadboard and was actually 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.

Light Sensor

The objective of this stage involved setting up the light sensor/capacitor and reading the 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 provided less challenging than other stages.

ZWay, ZWave USB Dongle, and Switch Setup

The objective of this stage involved setting up a simple 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 USB dongle, and the inclusion process for the ZWave Plug-in Switch Outlet. This stage did not involve any Python programming.

Setting up the ZWay software on the Raspberry Pi was relatively easy as you could install it using thru the typical “apt-get install” functionality.

However, plugging in the USB dongle wasn’t nearly as automatic. We needed to verify it was present using “lsusb” and then specifying that the device was device “ttyACM0” in the "Z Wave Network Access Page". The 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 US frequency of 908 MHz.

The addition and inclusion of the Leviton Plug-In Outlet 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 one device. A simple query online indicated that this was because a license for ZWay is required when using a USB dongle. After the license was purchased for approximately $30, ZWave immediately recognized the Leviton Plug-In Outlet and it worked seamlessly.

Testing Leviton Plug-In Outlet Activation/Deactivation with Python Code

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

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. Also, we did not want specific outlier or erroneous readings of the light sensor