Christina Nguyen ([cpn716@vt.edu](mailto:cpn716@vt.edu))

Nathaniel Hughes ([njh2986@vt.edu](mailto:njh2986@vt.edu))

ECE 4564, Assignment 3

Fall 2015

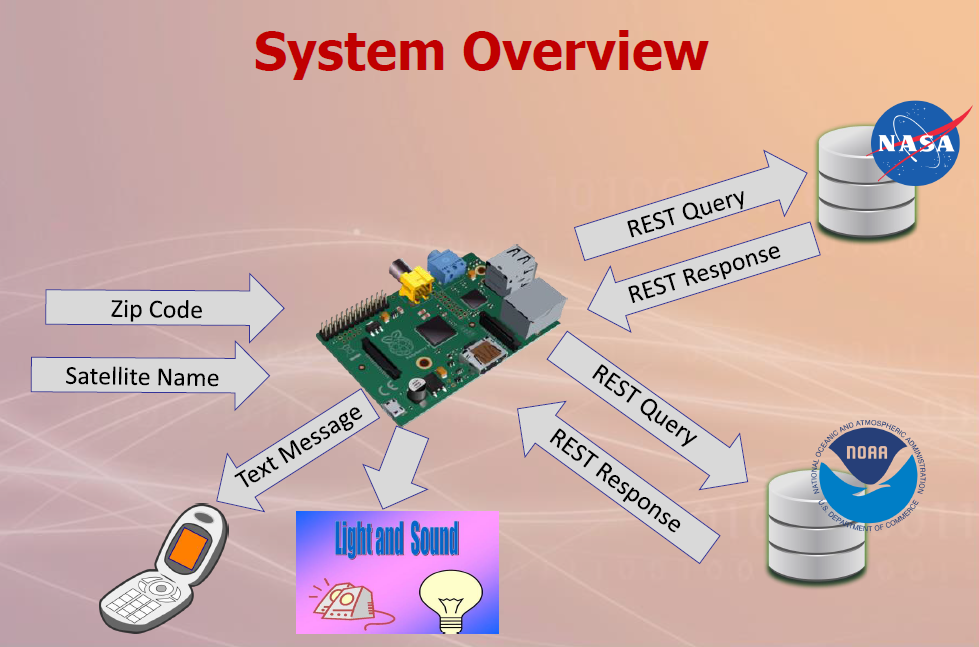
Section 1 – Objectives

The purpose of Assignment 3 is to use a Raspberry Pi as an Event Gateway, which would notify users of when particular satellite viewings are. The user would pass in arguments into the command prompt: zip code and a satellite’s NORAD ID. With these arguments, the Pi would make RESTful queries to a weather API and compare the received forecast information with satellite data, obtained through *ephem*, to determine if the said satellite is viewable in the provided zip code. The next five viewings for a satellite should be printed out, as well as its TLE and the weather forecast for the next 16 days. When a viewable date approaches, the user should be notified 15 minutes in advance via SMS, sound, and a flashing LED.

To provide a break-down of each step, the Pi would first read in the zip code and NORAD ID arguments. With those arguments, it makes queries to obtain satellite passes in the sky, over the provided zip code. The Pi then makes queries to the *OpenWeatherMap* API and processes the 16-day forecast it receives. It processes the forecasts by comparing it with the satellite information it received to determine if it is viewable. The conditions that make it viewable in the requested area are:

1. Dark sky (an hour before sunrise and an hour after sunset).
2. The sun’s height is between 10 and 25 degrees before the horizon.
3. The satellite is illuminated. This is assumed due to the other conditions.
4. The satellite’s elevation angle is at least 25 degrees above the horizon.
5. The weather API indicates that the sky is clear.

After bypassing the challenge of integrating various API’s in the code, an algorithm was developed to process the information, using the listed criteria. Based on the criteria, when a viewing event is upcoming, notifications are invoked on separate threads.



Section 2 – Team Member Responsibilities

Nate processed the weather and satellite data, writing the algorithm to determine viewable dates. Thus, he did some conversions and manipulations of JSON objects and printed them accordingly. He parsed the TLE data of the satellite and called various functions and API’s to get viewings. For satellite data, he used *ephem* to call functions like *Observer()* and extracted necessary information needed for the project such as latitude and longitude of the viewing area. He also called functions like *Sun()* to obtain its elevation. Rise and set times were also obtained for a satellite. All satellite viewings obtained were stored in a list. The same was done for weather data. The weather API was simpler to deal with. Its API was called through a Python URL library. Its URL was passed in with the latitude, longitude, and API key from Christina’s *OpenWeatherMap* API key. An array of Booleans were stored to keep track of when the sky is clear for viewing within the next 16 days.

Christina implemented the notifications. It was made modular so that it could be invoked when needed. The notifications were originally written as a single function. Once its functionality was tested and confirmed, it was split into threads. The LED flash and audio functions were split into their own threads, so that they could run concurrently, while the program still listened for upcoming viewings. The notifications would stop after 10 minutes, with the exception of the SMS, which sends a single text message to the phone. Upon setting up its functionality, a Twilio API key also had to be obtained to use its SMS sevices. Pygame was used for the audio.

Section 3 – Conclusions

Two challenges that stood out for this project were learning many various API’s and developing the algorithm to process the data in the Event Gateway. Tackling the assignment without any astronomy background also provided some challenges. For Computer Engineers who are fond of low-level programming, learning API’s proved to be an obstacle, but necessary in the academic growth.