Riley Cooper : cooperra@vt.edu
Sarah Kharimah : sarahkh@vt.edu
Zachary Yee : zjwy99@vt.edu
Anthony Clifton : antclift@vt.edu
ECE 4564, Assignment 3

Section 1: Objectives

The objective of this assignment is to further our understanding of accessing and retrieving information with RESTful interfaces, including Space-Track.org and Open Weather Map databases, utilizing third-party Python libraries, including PyEphem, and generating user notification through Python and Raspberry Pi 3.

Section 2: Team Responsibilities

Section 2.1 Weather: Zachary Yee:

For this part in the assignment, we had to use the Open Weather Map api to get the forecast data. In order to get the forecast, all we had to do was insert a ZIP code and all the relevant data was outputted as a JSON. The longitude, latitude, and cloudiness was extracted from the JSON. Using this data and data taken from the satellite, we can determine if the satellite is a viewable event for a given ZIP code.

Section 2.2 Finding Satellites: Sarah Kharimah:

The next part of the assignment is to obtain the specified satellite's TLE (Two-line Element), which contains orbital elements. To achieve this, we gained access to Space-Track.org API and database and built a query that looks for the orbital elements of the NORAD ID of a specific satellite. By specifying the desired date and the satellite's NORAD ID, using Python Requests, we are able to send a RESTful POST request, containing the query, to the database and receive TLE values as a response.

Section 2.3 Calculating Visible Satellites: Anthony Clifton:

Calculation of the visible satellite times was done through the use of PyEphem with the data provided by the Weather and NORAD modules. The weather module was used to find the latitude and longitude through a user inputted zip code. The zip code was then used along the user inputted NORAD ID code to receive the TLE data. The TLE data was analyzed by pyEphem which returned the times that the satellite was visible. The date was then checked with the 15 day forecast provided by the weather module and only those events on days with <20% cloudiness were added to the list of visible events. The function would continue to search through the events until either 5 visible events were found or the list had extended beyond 15 days. The list of visible events would later be printed and used for the notification module.

Section 2.4 User Notification: Riley Cooper:

This part of the assignment requires three types of notification at fifteen minutes before a satellite will appear. First, it uses the Twilio API to send a text message to the user. The text message contains all of the information from the python datetime object. Next, it starts flashing an LED and playing a beeping sound to notify the user. As soon as the calculator determines when the satellite will become visible, it can call this function, and the effects won't start until fifteen minutes before the event. Instead, it shall sleep until the correct time.

We produce the sound with the aplay system command. We used the subprocess module with the Popen command to create a new process to play the beep and not wait for it to complete before sleeping for one second. That makes sure the light blinks at the proper interval. The beep doesn't last exactly one second, and so throws off the timing.

Section 3: Conclusions

Python has a very convenient way to calculate the time and differences in time. A timedelta object can be added to a datetime object to create another datetime object at the new time.

At the completion of this assignment, we were able to utilize Python Requests, RESTful interfaces, third-party Python libraries, and Raspberry Pi GPIO. It was a valuable experience because we were able to apply what we have learned in class and obtained data that can be used in real-life scenarios. For example, weather forecast segment on morning news can definitely use the data obtained from a satellite to predict the weather.