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

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

ECE 4564, Assignment 1

Section 1 – Objectives

Assignment 1 aimed to use Python network programming to have two Raspberry Pi’s communicate with each other through sockets. The first Raspberry Pi (RPi 1) was to act as a client that takes in a user’s Tweets and parses it accordingly. The parsed Tweets, which contained instructions, were processed and sent to RPi 2, the server side, to control an LED. The format of the Tweet is as follows:

*@UserAcc #ECE4564\_IPByte1\_IPByte2\_IPByte3\_IP Byte4\_PortNo\_GPIOinstruction*

The GPIO instruction portion could be one of three things: *LEDON*, *LEDOFF*, and *LEDFLASH*.

Twython, an open-source API, was used to accomplish the task.

Section 2 – Team member responsibilities

**Christina**

**Nathaniel**

* Raspberry Pi 1
* Stream tweets; identify and process relevant tweets
* Set up socket to communicate with IP spec’d in tweet
* Communicate GPIO command to Raspberry Pi 2
* Wait for acknowledgement from Raspberry Pi 2 (the server); on reception, send acknowledgement tweet to VTNetApps

Section 3 – Conclusions

Overall, this project went smoothly, but there were a couple of obstacles faced: Effectively making use of the Twitter API proved to be a challenge; a stream of all public tweets was set up, and filters were implemented in order to identify and process the tweets which are relevant GPIO commands. The team learned the Twitter API and the Twython wrapper interface, and developed an understanding for the types of applications the Twitter API could be utilized for.

A second challenge presented itself in multiple ways, related to request restrictions enforced by Twitter for their public API. Command and acknowledgement tweets must be unique, so deletion of tweets was necessary before reusing the same IP address and GPIO command in a tweet. The team learned to adapt to the restrictions enforced by a platform in order to produce intended results, even if under different design.

A third major challenge was concurrency on the side of the server – ensuring the program was still accepting communications on its open socket while controlling the hardware, i.e. flashing the LED. A separate thread was created to flash the LED, so that additional commands could be received and processed in parallel. The team learned the value of threading in network communication – that always listening is absolutely essential.

This assignment was a good experience and offered a lot to learn in the area of network programming, specifically concurrency and socket communication in Python. Hardware malfunctions, causing SD card reimaging, proved to be a headache. And the team is still working on building a more permanent interface to each Raspberry Pi by employing static IP’s.