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*. Turning on and off the LED are straightforward. For flashing, the LED was to remain flashing until the Twitter user says otherwise. Thus, the flashing was done on a separate thread.

Twython, an open-source API, was used to accomplish the task. The client Pi streamed the Tweets in real-time. After having sent out the GPIO instruction to Pi 2, it sent a Tweet to the Network Application course handle (Twitter account) when it received an acknowledgement.

Section 2 – Team member responsibilities

**Christina**

Christina programmed the server side (RPi 2), which took in the GPIO inputs and controlled the LED accordingly. First, the LED was turned on in the Python code to ensure it was hooked up to the RPi 2 correctly. Once the base code was written out, the client Pi was used to send out plain GPIO instructions to the server Pi to test that it properly received the instructions. It also tested whether or not the acknowledgements were being sent back to the client. Originally, for the flash instruction, the LED merely turned on and off once. After initial server side testing, the flashing was done as specified by the assignment—continuously until a different instruction and on a separate thread. With this change, a parameter was passed into the Thread, the parameter being the GPIO instruction. This caused issues as the instruction was incorrectly processed by the Thread; it interpreted the instructions to have been as many inputs into the Thread as there are characters in the instructions. To get around this issue, a flag was written particularly for the flashing command. The flag was raised whenever the user passed in the LEDFLASH instruction and the Thread was started from that. Once different instruction was parsed to the Pi 2, the flag was set low again.

**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 contain relevant GPIO commands. The team learned the Twitter API and the Twython wrapper interface, and developed an understanding for the functionality and use of the Twitter API.

A second challenge, which presented itself in multiple ways, is the restrictions enforced by Twitter for its 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 the 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, requiring SD card reimaging, proved to be a headache. The team is still working on building a more permanent interface to each Raspberry Pi, by employing static IP’s. This assignment served as a good introduction to network applications, the Twitter API and the Raspberry Pi platform.