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 identifies and processes relevant Tweets and parse 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 was an API used to accomplish tasks within the assignment. The client Pi streamed the Tweets in real-time. Part of the goal of the assignment was to filter pertinent Tweets to send to RPi 2 so that it could execute the commands (the GPIO instructions). 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 from RPi 2.

Ultimately, the assignment prepared students to program network interfaces in Python and work with APIs, which included the Twitter API and the socket API. In order to make the Pi’s communicate with one another over the network, students were to work in teams. Thus, collaboration and teamwork became another learning aspect for the course.

Section 2 – Team Member Responsibilities

Nathaniel primarily worked on the RPi 1, streaming and filtering out unrelated Tweets. His Pi had set up a socket to communicate with the IP specified in the Tweets. From the Tweet, the GPIO instruction was extracted and sent to the RPi 2. Once sent, the RPi 1 was to wait for an acknowledgement from the RPi 2. Upon successful receipt, RPi 2’s acknowledgement was sent to the VTNetApps handle. Various design decisions were being made about properly parsing the Tweets, since the Pi was streaming all Tweets in real-time. Many test instructions were Tweeted out throughout the design process.

Christina programmed the server side (RPi 2), which took in the GPIO inputs and controlled the LED accordingly. The client Pi was also responsible for sending acknowledgements back to the client Pi. 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. Much about threading was learned.

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, developing an understanding for the functionality and use of the Twitter API.

A second challenge, which presented itself in multiple ways, was 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 that 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.