ECE 4564 - Group 20

Ethan Brooks ethanb98@vt.edu

Chris Honaker chris98@vt.edu

Kishan Parikh kjp2376@vt.edu

Assignment 2

Wishing Well

In this project, our job is to set up two Raspberry Pi's, one as a capture pi, and one as a consumer, using the direct exchange of RabbitMQ. Our RabbitMQ has 4 different exchanges: Library, Squires, Goodwin, and Commands. Library has 3 queues: Noise, Seating, and Wishes. Squires has 3 queues: Food, Meetings, and Rooms. Goodwin has 2 queues: Classrooms and Auditoriums. Commands exchange has 1 queue: SendtoCapt. MongoDB also has a similar layout to the RabbitMQ. There are 3 warehouses in MongoDB: Library, Squires, and Goodwin. These warehouses are further split into databases. Library has 3 databases: Noise, Seating, and Wishes. Squires has 3 databases: Food, Meetings, and Rooms. Goodwin has 2 databases: Classrooms and Auditorium. Our capture pi has a few functionalities: controls the GPIO LEDs, captures the tweets, stores the messages in the MongoDB database, displays the checkpoints to the screen at the appropriate times, and sends the messages to the repository pi.

The RabbitMQ portion of the code serves as a way for the two Raspberry Pis to communicate with one another. Our RabbitMQ code has 4 different exchanges: Library, Squires, Goodwin, and Commands. Library has 3 queues: Noise, Seating, and Wishes. Squires has 3 queues: Food, Meetings, and Rooms. Goodwin has 2 queues: Classrooms and Auditorium. Commands exchange has 1 queue: SendtoCapt. Library, Squires, and Goodwin exchanges serve as a way for people to tweet messages associated with the 3 places on campus with a specific message that goes with one of the queues. The Commands exchange is an exchange the sends messages from the repository pi to the capture pi in the SendtoCapt queue. We need this extra exchange so that we have a way to send checkpoints from the repository pi to the capture pi to display on the screen. All of the exchanges were direct exchanges in RabbitMQ. Also, all of the queues we were given a routing key that is the same as the name of the queue. This allows every queue to have a unique key so that every message goes to the right place.

Twitter utilizes the Twitter API Stream to look at all tweets sent from a given user, and combs through them to find tweets about specific things. In our case, we are searching for the following hashtag: #ECE4564T20. Any tweet with that hashtag will be grabbed and scanned, and should contain the following format: #ECE4564T20 request:location+command "string sentence". First, the hashtag is removed from the tweet so that the rest of the tweet is only about components wanted. The next thing looked at is whether it is a producer or consumer request, denoted by 'p:' and 'c:' respectively. If a producer (p:) request is found, the location is scanned, which is found between the ':' and '+', and the command is found from after the '+' to the 'whitespace'. After

ECE 4564 - Group 20

Ethan Brooks ethanb98@vt.edu
Chris Honaker chris98@vt.edu
Kishan Parikh kjp2376@vt.edu

the location and command are found, the MongoDB server, talked about below, and the RabbitMQ server, talked about above, can correctly read in the string message, between the quotes, and store it in the correct queue. On the other hand, if a consumer (c:) request is found, the location and command are read in the same way as for a producer, but a consumer has no string message. Instead, it will ask the necessary queue for the oldest data from that given location and command. For instance, if someone tweets #ECE4564T20 p:Library+Noise “4th floor is loud today”, and someone else tweets #ECE4564T20 c:Library+Noise, they will receive a message that says 4th floor is very loud today.

The RGB LED that we received is directly connected to the board using the Raspberry Pi’s GPIO pins. The RGB LED is connected to red, ground, green, blue pins, and each of the color pins has a resistor attached to it to ensure current is not high for the Raspberry Pi to handle. On the Pi, the program created multiple functions to turn on and turn off the 3 different LED lights. Throughout the code, while waiting for a command, the light will have all three LEDs on to create a white light. Once a producer request is received, the green and blue LED lights turn off, leaving only the red LED on. Once a consumer request is received, the blue and red LEDs turn off, to leave only the green LED light on. After a few seconds of being on, the producer and consumer lights will revert back to the white LED light as it awaits more requests .

MongoDB is used to store commands sent from twitter. These commands are parsed and stored in their correct warehouses and databases. The warehouses are called Library, Squires, and Goodwin. The Library has 3 databases: Noise, Seating, and Wishes. Squires has 3 databases: Food, Meetings, and Rooms. Goodwin has 2 databases: Classroom and Auditorium. The commands are also parsed into the document format provided in the project slides and then stored into their respective database. The mongoDB instance is not started by the python code, but instead it is run as a service on the pi. This allows the server to automatically be started when the pi turns on.

At each of these stages and more, checkpoints are output by both the capture pi and repository pi, as specified in the grading rubric. Each checkpoint lists the stage the capture pi and repository pi are at, such as if the tweet is captured or if the GPIO LEDs is updated. The checkpoints acted as good flags of code progress as we ran and decoded the project.

Our project was completed and met all the requirements set for it in the rubric. We communicated wirelessly from Eduroam. The Twitter API keys were written in the correct file. Both files can be written the correct way from the command line with the correct arguments.

ECE 4564 - Group 20

Ethan Brooks	Chris Honaker	Kishan Parikh
ethanb98@vt.edu	chris98@vt.edu	kjp2376@vt.edu

The Twitter portion of the code does not need to be restarted every time to work, and the code is able to extract all the correct information from the tweets. All the commands are able to be stored in the MongoDB database, and the command status is displayed by the GPIO LEDs. The RabbitMQ portion of the code was written as a direct exchange with the ability to consume and produce messages from one pi to another. Lastly, all of the correct checkpoints are displayed to the screen by the capture pi at the correct times.