Riley Cooper : cooperra@vt.edu
Sarah Kharimah : sarahkh@vt.edu
Zachary Yee : zjwy99@vt.edu
Anthony Clifton : antclift@vt.edu
ECE 4564, Assignment 4

Section 1: Objectives

The objective of this assignment is to implement a client/server model with multiple clients communicating with a dedicated server, RESTful interactions using CoAP, and a GUI component. The clients use the REST interface to build a 10 block wide by 2 block tall by 1 block deep wall in the Minecraft world. They get the current location with a GET request, and if it is their turn, they POST a new location for a block. This project uses three clients. The requests need to be made over COAP. The server indicates which client's turn it is with different color LED's.

Section 2: Team Responsibilities

Sarah Kharimah: Token algorithms
Sarah implemented the tokens portion of the code to allow the server to differentiate which client to listen to for POST requests. Client A responds to a token value of 0, B to a token value of 1, and C to a token value of 2.

Zachary Yee: Minecraft server and API
For this part of the project, Zachary had to use the MInecraft API to get the player position and set blocks at a given position. The server received the position coordinates from the clients and then using the Minecraft API, the server set a block at the position. The server determined if the wall was finished if total number of POST requests given to the server was 10. The counter reset and the process started again except one position higher in the y-axis to give the wall some height. After the second time the number of POST requests equals 10, the server tells all the clients the wall is complete.

Riley Cooper: COAP communication
Riley designed the RESTful COAP communication between the minecraft server Raspberry Pi and the client Raspberry Pi's. This part of the project required the clients to make GET requests to the server which obtained a location and token once we integrated the parts. The server needed to respond with a tuple that sent the information. The clients also needed to make POST requests based on the location and token, so Riley designed the interface to make the post request. He used the aiocoap library and based the code off of the aiocoap examples on the website for the library. The code creates a class for storing location information, which also includes the token and any other necessary information. It also uses asyncio to handle requests as events and respond to these events.

Anthony Clifton: Integration and troubleshooting

Anthony conducted all of the troubleshooting and implementation of final client and server codes. This process included setting up the server to use the GPIO and a connected LED to indicated the current player’s token value. The were slight issues with the clients not posting correctly and some changes had to be made to the hosting values. Once the troubleshooting was finished, the notification that the wall was complete and the killing of the clients was added to the client code.

Section 3: Conclusion

We had difficulties with upgrading Python to version 3.5. The original instructions worked on a normal Debian Linux VM where Riley designed the COAP code, but the same instructions failed on the Raspberry Pi with little in the way of help on the internet. The main issue was upgrading pip. We resolved this issue with the second set of installation instructions sent out.

Integrating the Minecraft API with COAP was really simple. The only issue that came up was converting between types when communicating between the Minecraft API and COAP. We determined that one main reason to upgrade was the interface with aiocoap using asyncio required the update. Python 3.4 used a different method to set up the GET and POST requests and responses.

We also ran into an issue with copying and pasting code between the clients. Anthony mentioned the issue with POST, which occurred because of a line of code that we miscopied between the client programs. We eventually found the mistake and managed to correct it for tests.

We found the minecraft code fairly intuitive. We could easily figure out which calls performed which actions. Once we established how to use that code, we began to integrate it into the server code. It was also interesting to see our code affect the virtual world in minecraft through the GUI.