Report Date: 11/11/2022
To: ematson@purdue.edu, ahsmith@purdue.edu, lee3450@purdue.edu
From: Team Coyote(Sensors & Network)

- Hyemin Lim (freemini2@cau.ac.kr)
- Nayoun Kim (202010766@live.wsu.ac.kr)
- Jaehui Boo (32192075@dankook.ac.kr)
- Hyeongjun Kim (aa980305@cu.ac.kr)

Summary

After deciding the project's new architecture, there was a challenge. For Raspberry Pi to communicate with LoRaWAN, a LoRa HAT that supports LoRaWAN functionality was needed. However, LoRa HAT available at KSW only supports LoRa point to point communication and existing LoRaWAN supported HAT is way over the team's budget. That's why we suggested an alternative solution, which is connecting Raspberry Pi and ESP 32 that can communicate through LoRaWAN. Raspberry Pi will do the classification of Coyote sound and sends signal containing only the boolean result to ESP 32. Then ESP 32 will send a frame to The Things Network when it receives a signal from Raspberry Pi. [1]

To create a connection between Raspberry Pi and ESP 32, the team tried to use different serial communication methods. First one is a protocol called UART(Universal Asynchronous Receiver and Transmitter), which uses RX(receiver) and TX(transmitter) to conduct a two-way communication between two devices. It is widely used protocol so the team decided to try it first. And I2C(Inter-Integrated Circuit) was also considered as alternative, since it was popular method for serial communication other than UART. [2]

The problem was that both methods needed cables and breadboard, and things the team got was not working well. For example, during I2C connection, I2C bus couldn't detect a device's address. But after changing the cable and breadboard, it popped up in the I2C bus. And it disappeared shortly. Therefore the team assumed the connection through wiring cables is unstable.

Establishing serial communication through USB port is considered as alternative. Raspberry Pi can write or read arduino serial port so the team implemented it in the ESP 32. Then the serial communication code and packet sending code on ESP 32 was combined, making a signal from raspberry pi into a trigger of sending packet.

The Unity environment has been set up and necessary installations have been completed to implement visualization of the project. To show the farm, the code succeeded in bringing the surrounding environment to the model ID registered on google maps in 3D.[4][5] For the convenience of the user, the implementation of the touch-related map moving code has also been completed. After building the application in Unity, the data transmission, and reception between the nodeJS server and the Unity client code were completed. During the process, the push notification code also worked well.[6] Now, trying connection between unity and mobile with a camera.[3]

The method to detect the coyote is triangulation. However, triangulation is using 3 triangles to detect the thing using TDOA(Time Difference of Arrival). Besides using triangles, it is more accurate using hyperbola. Therefore, it is done using Python to draw three hyperbola and find intersection between two hyperbola. The intersection will be the location of the thing.[7]

Fig. 1. LoRa frames sent by ESP 32, which is triggered by Raspberry Pi

What Coyote Team completed this week:

- Sent a signal from Raspberry Pi to ESP 32 through USB Serial connection
- Sent a frame from ESP 32 to LoRaWAN Gateway using a signal from Raspberry Pi as a trigger
- Worked on Localization algorithm
- Installed Unity 2021.3.9f1 version, plugins, asset, SDK, unity package file for visualization
- Registered google map API key, SDK for Unity
- Succeeded in load the registered coordinate surrounding environment and buildings in 3D via google map API for Unity.
- Wrote a code about camera object moving(enlarges, reduces, move) in 3D map when touching a phone screen.
- Tested data transmission and reception between nodeJS server and Unity client
- Completed the test by push notification code in Unitys
- Succeeded drawing 3 hyperbola and finding intersection between 2 hyperbola using Python.

Things to do by next week

- Conduct a experiment on Localization code
- Make Raspberry Pi Camera to work
- Work on visualization platform
- Create a code about push notification with Unity
- Research about nodeJS for Unity

Problems or challenges:

- Functional ideas using camera sensors are needed without machine learning model.

References

[1] automaticaddison. “Two Way Communication Between Raspberry Pi and Arduino” Automatic Addison. Jul, 6, 2020. [Online]. Available: https://automaticaddison.com/2-way-communication-between-raspberry-pi-and-arduino/

[2] Hopkins. J. “Understanding the difference between UART and USB” Totalphase. Accessed: Nov, 11, 2022. [Online]. Available: https://www.totalphase.com/blog/2022/01/understanding-differences-between-uart-and-usb/

[3] Press Start. “Unity - Mobile Panning with a Perspective Camera” Youtube. Nov, 9, 2018. [Online]. Available: https://www.youtube.com/watch?v=4_HUlAFlxwU

[4] Google. “Maps SDK for Unity Key Concepts” Googles map platform. Accessed: Nov, 9, 2022. [Online]. Available: https://developers.google.com/maps/documentation/gaming/concepts_musk

[5] Kristopher C. “A Guide to Using Google’s Maps SDK for Unity 3D - Part 1” Medium. Apr, 19, 2021. [Online]. Available: https://medium.com/everdevs-community/a-guide-to-using-googles-new-maps-sdk-for-unity-3d-ed1d68b2305e

[6] Ahmed. S. “WebSocket Client & Server (Unity & NodeJS)” Medium. Dec 6, 2020. [Online]. Available: https://medium.com/unity-nodejs/websocket-client-server-unity-nodejs-e33604c6a006

[7] SymPy Development Team. (2022). Accessed: Aug. 22, 2022. [Online]. Available: https://docs.sympy.org/latest/tutorials/intro-tutorial/index.html