Introduction

For the final project in EEC 172 I decided to create a drone that could respond to accelerometer commands from the CC3200. In order to accomplish this task, I fed data from the accelerometer and sent it to AWS, where the drone was listening in order to figure out how to fly.

Parts List

- AWS Account
- TI CC3200 Board
- Raspberry Pi 3
- Pixhawk Flight Controller
- Drone Parts
    - Frame
    - Propellors (Get extras)
    - Power Distribution Board
    - ESCs
    - Brushless DC Motors
    - RC Batteries

How I Did It

There were 3 main components to the actual project, the drone assembly, the CC3200 programming, and programming the raspberry pi. I will not cover drone assembly in this guide, as there are too many possible configurations. The only assumption is you have already built a drone with a pixhawk flight controller, and that it can fly.

Programming the CC3200

Programming the CC3200 was simply a matter of adapting the code from Lab 5. We had already created a thing in AWS IoT, attached and downloaded the certificates to the CC3200 board. Since we had already implemented the creation of POST and GET calls on the board, I simply needed to make new messages to update the thing shadow for my purposes. I connected the accelerometer on the board through I2C (making sure to connect jumpers J2 and J3). I then read in the accelerometer values and determined which way the board was being tilted (taking code adapted from Lab 2). I then created a specific POST message that posted the board's position to AWS. The positions were fairly primitive as I was just trying to do a proof of concept. The messages were forward, backward, idle, left and right. In a more developed implementation I might include more potential positions like the diagonals, or directly feed in accelerometer inputs to the AWS shadow.

Programming and Connecting the Raspberry Pi

Before you can program the Raspberry Pi, I’m going to do a quick overview of what it's actually being used for in terms of the drone. The pixhawk flight controller is essentially just an STM32 with the necessary sensors in order to do flight control for a drone. Under the hood is some pretty advanced systems programming that is running a lot of complex flight control algorithms. While the board does have a POSIX based operating system, it is still very difficult to program due to the potential of messing with the flight control algorithms being run on the flight controller. This is where the Raspberry Pi comes in. We can run more high level algorithms and send commands to the flight controller. This essentially allows us to create more advanced high level software that can send flight commands to the flight controller without exposing any of the lower level flight control algorithms. The next section will show how to connect the Raspberry Pi, and what software was used to set it up.

FlytOS

In order to send flight commands we use an extra layer of abstraction known as ROS. However, ROS is exceptionally difficult to setup and install, and also has a lot of difficulty in actually sending commands to the Pixhawk. This is where FlytOS comes in. FlytOS is an additional layer of abstraction on top of ROS, which handles the installation and backend of ROS. In order to install FlytOS, simply install the provided image on their website.
http://docs.flytbase.com/docs/FlytOS/GettingStarted/RaspiGuide.html#rpi-guide

The link above details how to flash your Pi with the FlytOS image, and then also wire the raspberry pi to the Pixhawk. After setting everything up I had to do a quick flight test to see if Offboard mode works. The pixhawk always makes you start in manual flight control when armed. In order to fly in offboard mode, we need to program a switch on our transmitter that switches the flight mode. This can be done using a program called QGroundControl. After testing that offboard mode works (warning: the drone will take off when the mode is switched), I went to program the pi.

AWS for the Pi

As in Lab 5, we need to make it so that the raspberry pi can actually communicate with AWS IoT. This was a particularly frustrating challenge for me as I was struggling to get two devices to interface to the same thing shadow. I simply created a new security certificate with all permissions and attached it to the same CC3200_Teja that I created in Lab 5. I then downloaded the certificates to the Raspberry Pi. After that I made a python script that used the FlytBase API and paho MQTT in order to listen to the AWS Thing shadow. Based on the position message read from AWS, I set the velocity of the drone to a certain value on the X/Y plane in order to get the drone to move left, right, forward and backward.

Challenges

There were numerous challenges I faced with this project. The most significant issues stemming from hardware. The wifi chip on my Pi got burnt out so I needed to SSH into it through an ethernet cable. The serial communication on one of the Pixhawks also stopped working, but fortunately there was a backup board. In terms of software, everything was pretty smooth other than the fact that I was stuck for about 4 hours getting the Rpi to connect to AWS, because I didn't set the proper permissions.

Future Goals/Conclusion

This project actually started the backend to a lot of potential avenues to develop on. The first would be to create my own low level flight control on the CC3200 and send some relevant messages to AWS while flying. The second is to expand the accelerometer to control to 3 dimensions and also be able to move the board up and down on the z axis to change the drone's height. Finally, because we setup the AWS backend and got the drone to communicate with AWS, I could very easily make an Alexa controlled drone by making a skill and then activating a lambda function that update the thing shadow. Overall, this project taught me how to communicate with a pixhawk and send flight commands to it. I also learned about the MQTT communication protocol for more advanced IoT.