DRONE NETWORKING

In this project we essentially work on a larger perspective aiming at combining the ground networks & aerial networks. We aim at implementing the concept of flying base stations (LTE networks) attached to the drones which would be seen as a common feature in the future. It is with aim that we start this venture & we are having several steps to achieve this. Currently our test bed consists of a robot car to which a drone would be attached & another drone in the air communicating with the aforementioned drone on the moving car. We essentially plan to establish communication between the drones to synchronize their movements.

Getting Started

These instructions will give an idea of the project up and running on your machine for development and testing.

Prerequisites

Please have the drone with the cables & a PC alongside to get started with the process

Installing

Hardware:

Connecting Cables:

USB adapter is used for plugging multiple USB devices like keyboard and mouse into the Intel aero board and a micro HDMI to HDMI adapter is used to connect the board to the monitor.

Power:

You can either use the wall socket power adapter or the LiPo battery. If you are working for long hours keeping the system ON on your desk, it is advisable to use the wall socket power adapter.

But remember that if you want to test the motors, connect the LiPo battery to the drone and not the wall socket power adapter since it doesn't provide you the required power to spin the motors.

Flashing:

You will first flash the operating system called Linux Yocto on the board. You will then flash the BIOS and firmware for the flight controller. These are really important to do because the version that is coming with the drone from the factory is old and you would need the new version with all new features and bug fixes

- First you will download an ISO file

  (downloadcenter.intel.com page to download the .iso linux image)

  and create a bootable USB disk from this ISO file on your development station.

Steps to create a bootable disk:

a. Insert the removable/USB disk to the windows machine

b. Launch Etcher (a software which you need to download)

https://www.techspot.com/downloads/6931-etcher.html

c. Select the .iso file, the USB drive and click on the "Flash" button

d. Wait for the image to be written and verified

- Once the ISO image is written, shutdown the Intel aero, wait for 5 seconds and restart it, press ESC enter the BIOS and select boot from USB key. When booting from USB key, select "install". This will reinstall Yocto from scratch.

Flashing the BIOS:

Download the latest BIOS from Intel download center

(https://downloadcenter.intel.com/download/27833/Intel-Aero-Platform-for-UAVs-Installation-Files?v=t)

and copy onto the Aero disk space.

- To install, first remove the previous version (v1.00.13) and then install the latest (v1.00.16).
- To do this use the CLI & enter the commands below

```
rpm -ev aero-bios-01.00.13-r1.corei7_64
rpm -i aero-bios-01.00.16-r1.corei7_64.rpm
aero-bios-update
```

Then reboot, the UEFI BIOS will detect the files and update.

On reboot check the BIOS version:

```
aero-get-version.py
```

You should see a message like this (or newer):

```
BIOS_VERSION = Aero-01.00.13
OS_VERSION = Poky Aero (Intel Aero linux distro) v1.6.0 (pyro)
AIRMAP_VERSION = 1.8
FPGA_VERSION = 0xc2
Aero FC Firmware Version = 1.6.5
```

Flashing the FPGA

Flash the new FPGA firmware:

```
cd /etc/fpga/
jam -aprogram aero-rtf.jam
```

Flashing the Flight Controller (RTF only)

To update the flight controller, use the following command:

```
cd /etc/aerofc/px4/
aerofc-update.sh nuttx-aerofc-v1-default.px4
```

In parallel of the Intel provided PX4 version, you can install the Ardupilot version: (Make sure you know the difference between PX4 and Ardupilot before flashing)

```
cd /etc/aerofc/ardupilot/
aerofc-update.sh arducopter-aerofc-v1.px4
```

Check the BIOS-FPGA-FC-OS version:

```
aero-get-version.py
```

You should see a message like this (or newer):

```
BIOS_VERSION = Aero-01.00.13
OS_VERSION = Poky Aero (Intel Aero linux distro) v1.6.0 (pyro)
AIRMAP_VERSION = 1.8
FPGA_VERSION = 0xc2
Aero FC Firmware Version = 1.6.5
```

Calibration:

When you receive the RTF Drone from the factory, it is already calibrated. But after you flash the flight controller, the calibration settings are not valid anymore. You need to recalibrate.

You'll need a computer or tablet running QGroundControl (QGC) and connected to the Intel Aero access point (called "Aero-*", password "1234567890").

Open up QGroundControl and ensure the GPS and Battery are active “colored black” and have basic functionality in the top panel:

Please refer this link for calibration:

https://github.com/intel-aero/meta-intel-aero/wiki/02-Initial-Setup#calibration

Once these steps are followed, **the drone is now ready to take off.**

Installing Ubuntu on Intel Aero:

Once you upgrade Linux Yocto and flash the BIOS, FPGA, Flight Controller, you can now install Ubuntu which will replace Yocto and keep the BIOS, FPGA and Flight Controller.

- Download Ubuntu 16.04.3 x64 Desktop. Note: do not try a more recent version of Ubuntu.
- Create a bootable disk (refer to the Ubuntu documentation)
- Plug Intel Aero to the wall power supply, the USB-OTG adapter, bootable USB key, hub, keyboard and mouse. Power on
- Type ESC to enter the BIOS
- Select boot manager, select your USB key and press Enter
- Install Ubuntu as you would on a computer

Intel Aero Repository:

Connect Intel Aero to the network with internet access (e.g. UB Secure) and run the following commands in the terminal to add Intel aero repo.

```
echo 'deb https://download.01.org/aero/deb xenial main' | sudo tee /etc/apt/sources.list.d/intel-aero.list

wget -qO - https://download.01.org/aero/deb/intel-aero-deb.key | sudo apt-key add -

sudo apt-get update

sudo apt-get upgrade

sudo apt-get -y install gstreamer-1.0 libgstreamer-plugins-base1.0-dev libgstrtspserver-1.0-dev gstreamer1.0-vaapi gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-libav ffmpeg v4l-utils python-pip

sudo pip install pymavlink

sudo apt-get -y install aero-system

sudo reboot
```

Ground Control Station:

QGroundControl (QGC) is required to calibrate the Flight Controller and useful to pilot the drone. There are other Ground Control Stations as well like Mission Planner, but QGC is more compatible with Intel Aero Drone.

In order to connect QGC to Intel Aero, run the following command in the terminal.

sudo mkdir /etc/mavlink-router/config.d

sudo gedit /etc/mavlink-router/config.d/qgc.conf

And fill it with:

```
[UdpEndpoint wifi]
Mode = Normal
Address = 192.168.1.147
```

Note: The above IP address should be your laptop's IP.

Restart the router by running the command:

sudo systemctl restart mavlink-router

Launch QGC on your laptop. It should receive automatically the telemetry feed from the drone.

BIOS

To flash the BIOS that's part of the Aero repo, type

```
sudo aero-bios-update
sudo reboot
```

FPGA

sudo jam -aprogram /etc/fpga/aero-rtf.jam

Flight Controller

```
cd /etc/aerofc/px4/
sudo aerofc-update.sh nuttx-aerofc-v1-default.px4
```

Intel RealSense SDK (for Intel RealSense R200 Camera)

```
cd

sudo apt-get -y install git libusb-1.0-0-dev pkg-config libgtk-3-dev libglfw3-dev cmake

git clone -b legacy --single-branch https://github.com/IntelRealSense/librealsense.git

cd librealsense

mkdir build && cd build
```

```
cmake ../ -DBUILD_EXAMPLES=true -DBUILD_GRAPHICAL_EXAMPLES=true
make

sudo make install
```

Intel Camera Streaming Daemon

Camera streaming daemon gets installed with aero-system. To verify, check:

```
systemctl status csd
```

You should see something like:

```
csd.service - Camera Streaming Daemon
Loaded: loaded (/lib/systemd/system/csd.service; disabled; vendor preset: enabled)
Active: active (running) since Wed 2017-11-08 17:28:00 PST; 6min ago
Main PID: 14616 (csd)
CGroup: /system.slice/csd.service
└─14616 /usr/bin/csd

Nov 08 17:28:00 frelon systemd[1]: Started Camera Streaming Daemon.
Nov 08 17:28:00 frelon csd[14616]: Could not open conf file '/etc/csd/main.conf' (No such file or directory)
Nov 08 17:28:01 frelon csd[14616]: Failed to connect to Mir: Failed to connect to server socket: No such file or directory
Nov 08 17:28:01 frelon csd[14616]: Unable to init server: Could not connect: Connection refused
Nov 08 17:28:01 frelon csd[14616]: (gst-plugin-scanner:14618): Clutter-CRITICAL **: Unable to initialize Clutter: Could not initialize
Nov 08 17:28:01 frelon csd[14616]: (gst-plugin-scanner:14618): Clutter-Gst-CRITICAL **: Unable to initialize Clutter
Nov 08 17:28:02 frelon csd[14616]: AVAHI START
```

Kernel version

Check the kernel version with *uname -a*, you should see:

```
Linux frelon 4.4.76-aero-1.2 #1 SMP PREEMPT Mon Nov 6 19:42:57 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

It should have aero in the kernel name.

Number of video devices

To check if the camera drivers are correctly installed, list the video devices with

```
ls /dev/video*
```

You should see a list like /dev/video0 /dev/video1 /dev/video2 ... up to 13. If less than 13, it's a problem (If you see only 3 video devices, it means you booted the wrong kernel).
The order of devices is also important. Check the details with

```
sudo v4l2-ctl --list-devices
```
You should see a long listing, including:

```
Intel RealSense 3D Camera R200 (usb-0000:00:14.0-4):
/dev/video11
/dev/video12
/dev/video13
```
R200 should have the devices 11, 12, 13

MAVLink router

Check if the system is listening for MAVLink messages on port 5760 thanks to MAVLink router (for the Ready To Fly Drone only):

```
netstat -l|grep 5760
```

You should see:

```
tcp        0      0 *:5760                  *:*                     LISTEN
```

RealSense

Test: run cpp-enumerate-devices to see if your camera is detected. On the Ready To Fly Drone, you should see:

```
Device 0 - Intel RealSense R200:
Serial number: 2481009843
Firmware version: 1.0.71.06
USB Port ID: 2-4
Camera info:
DEVICE_NAME        :        Intel RealSense R200
DEVICE_SERIAL_NUMBER:             2481009843
CAMERA_FIRMWARE_VERSION:     1.0.71.06
CAMERA_TYPE        :        PRQ-Ready
OEM_ID             :   OEM None
ISP_FW_VERSION     :        0x0
CONTENT_VERSION    :     12
MODULE_VERSION     :     4.2.5.0
IMAGER_MODEL_NUMBER :            31
CALIBRATION_DATE   :     2014-07-04 08:18:35 UTC
EMITTER_TYPE       :        Laser Driver 3
FOCUS_VALUE        :        0
LENS_TYPE          :             Newmax 58.9 x 45.9 degs in VGA
3RD_LENS_TYPE      :        Newmax 71.7 x 44.2 degs in 1080p
LENS_COATING__TYPE  :   Visible-light block / IR pass 43 nm width
3RD_LENS_COATING_TYPE:           IR coating
NOMINAL_BASELINE   :     70 mm
3RD_NOMINAL_BASELINE:            58 mm
Supported options:                min        max        step  default
```

```
COLOR_BACKLIGHT_COMPENSATION          : 0    ... 4          1     1
COLOR_BRIGHTNESS                      : 0    ... 255        1     56
...
```

To see the output of a camera, use the SDK tools. Examples:

- cpp-tutorial-1-depth (command line)
- cpp-capture (graphical).

PX4 Optical Flow Sensor:

For the drone to be more stable while flying, we have installed a PX4 Sensor, which has got an onboard sonar input and a high resolution camera to calculate the distance from the ground.

- In order to use the PX4Flow board, just connect it with I2C.
- Update the firmware on PX4Flow using *QGroundControl* (in the top left menu, click on CONFIG, then on Firmware Upgrade)
- Connect PX4Flow I2C to the Pixhawk I2C.
- The module will be detected on boot.

Follow the steps given under "**Configuration: Image Quality and Output**" from this link **https://docs.px4.io/en/sensor/px4flow.html**

Running the tests

Before you begin programming, you need to install certain packages.

sudo apt-get install python-pip python-opencv python-opencv-apps python-zbar zbar-tools vim-python-jedi vim-python-jedi python-editor eric idle vim-nox-py2

pip install Cython numpy

pip install pyrealsense

pip install dronekit

pip install MAVProxy

Here's the small python code used to arm the motors for 5 seconds:

Note: Make sure you unplug the propellers first.

```
#!/usr/bin/python
from dronekit import connect, VehicleMode, LocationGlobalRelative
import time

vehicle = connect('tcp:127.0.0.1:5760', wait_ready=False)
print "Arming motors:"
vehicle.mode    = VehicleMode("GUIDED")
vehicle.armed   = True
while not vehicle.armed:
print "  Waiting for arming to be finished"
time.sleep(1)
print "Keeping motors armed for 5s"
time.sleep(5)
print "Disarming"
vehicle.armed   = False
```

In this code,

i. dronekit - allows you to control PX4 and ArduPilot using the Python programming language
ii. connect - helps establish a connection to the flight controller.
iii. vehicleMode - decides the state of the vehicle.

Python Program for Indoor: Refer dronekit_indoor.py

The program has two functions.

i. The function "hover at" works such a way that you need to pass the height and duration as arguments and the drone takes off, reaches the given height, waits for few seconds and finally makes a landing.

ii. The function "line" works in a way that you need to pass the height, length and duration as arguments and the drone takes off, reaches the given height, waits for few seconds and moves forward for the given length and comes back to its original position and makes a landing.

Simulation and Visualization on QGC:

The following procedure will help you setup the environment in your Linux Virtual machine and create a simulator and then test your own code as if you are testing it on the drone real time.

Install the below packages on your linux virtual machine.

1. pip install mavproxy

2. pip install dronekit

3. pip install dronekit-sitl

4. Download and open QGC on your linux machine

5. Open a new terminal and run the below command.
dronekit-sitl copter --home=42.99,-78.78,0,0

Here, copter is your vehicle, 42.99,-78.78 are the latitudes and longitudes (that's the Davis hall co-ordinates), 0 being the height from the ground and the next 0 being the angle at which the drone points. (like 0 - north, 180 - south)

6. Open another terminal and run

mavproxy.py --master tcp:127.0.0.1:5760 --out:127.0.0.1:14551 --out udp:(ip address):14550

7. Go to QGC and under application settings, click on Comm Links. Click on Add and you can type in any name, and Type should be UDP, Listening port: 14550 and Target Hosts: 127.0.0.1:14550. Click OK and then click on Connect.

8. Open a new terminal and run

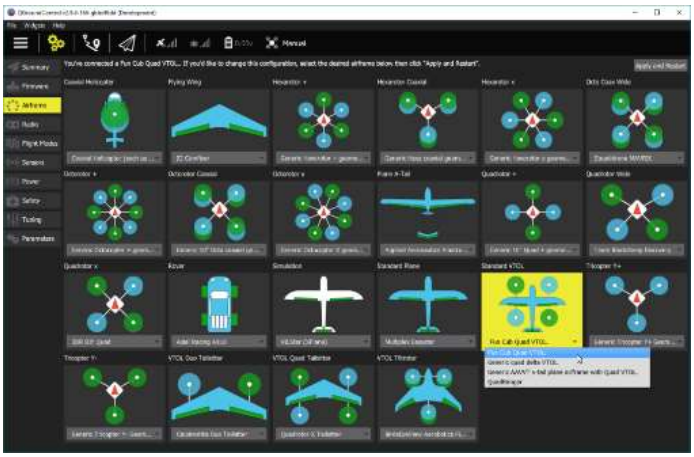

Figure 1-4. QGround Control

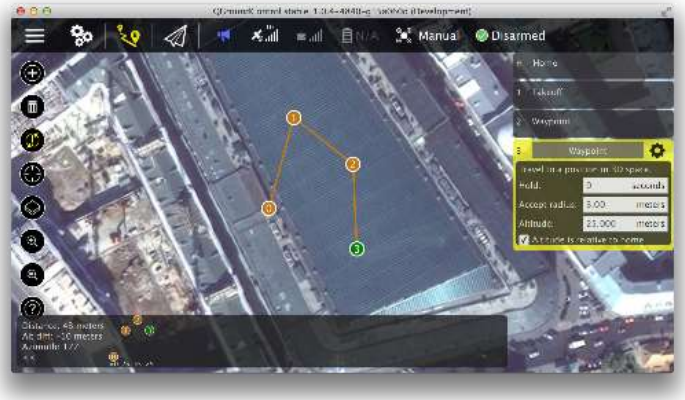

Figure 2

Figure 3

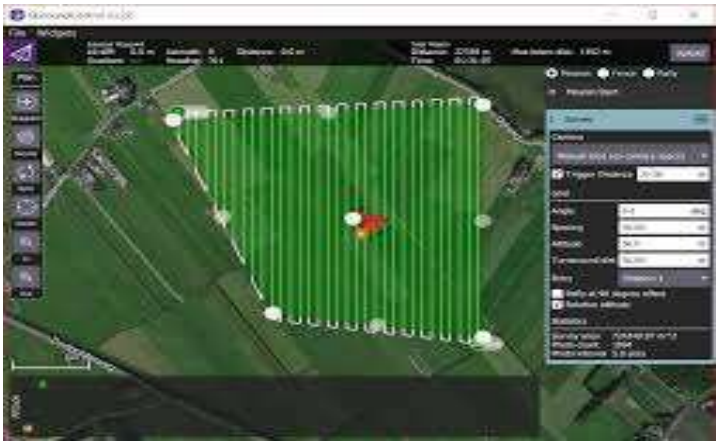

Figure 4

python sample.py --connect udp:127.0.0.1:14551

where sample.py contains your code. You should now see the drone in QGC working as per your code.

Here's a sample code which you can use to control the drone using the arrow keys on your keyboard. As per program, the drone has to reach an altitude of 10 meters. Once it gains that altitude, you can now move the drone in any direction which moves at a speed of 5 m/s. Irrespective of the current location, if you want the drone to make a safe land from where it took off, just press "r" on your keyboard. You can now see the drone coming back to its initial position.

Refer Code: Drone_keyboard.py

Built With

- Ubuntu Linux - The framework used
- Q Ground Control

Acknowledgments

https://github.com/intel-aero/meta-intel-aero/wiki/01-About-Intel-Aero