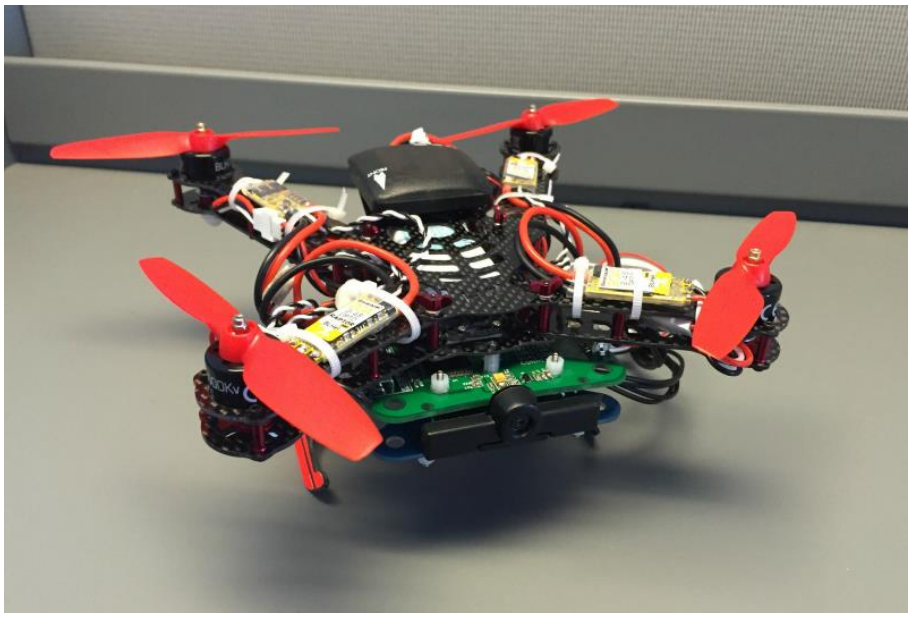
Build a Drone within 2 Hours

(updated 11/15/2016)

Chong Huang

This document is used to build a drone based on Qualcomm Snapdragon Flight board and **our designed flight control board**. Distribution of this document is strictly prohibited without our permission.



Outline

1. Components
2. Assembling
3. Programming

1. Component

Qualcomm Snapdragon™ Flight Kit (Link)
Flight Control Board (Contact us)
Blade Brushless Motor Reverse 200Qx Rotors (Link)
Blade 200QX Brushless Motor (Link)
Blade Red Propellers 200QX (Link)
Battery (Link)
Quadcopter Frame Kit (Link)
Wi-Fi antenna (Link)
GPS with Compass Kit (Link)
Android Device

2. Assembling

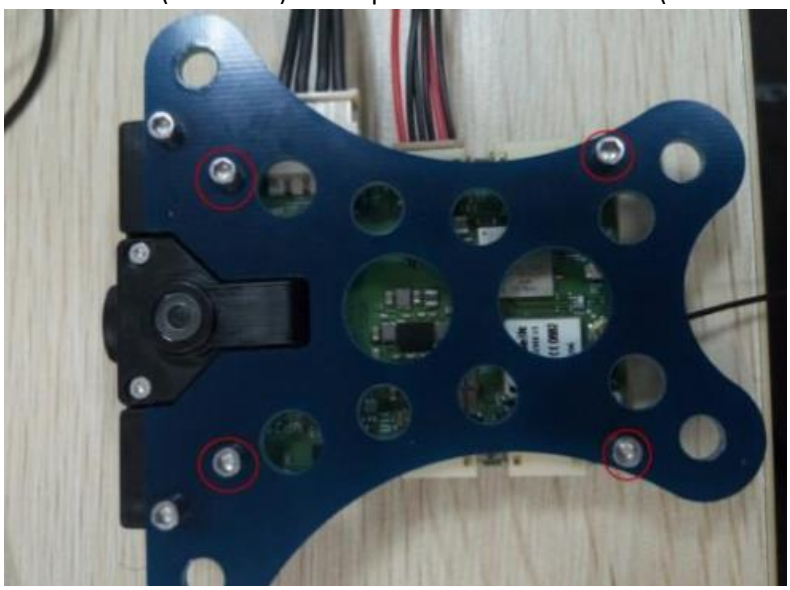
2.1. Frame Assembling

We recommend 200 Size Quadcopter Frame Kit (Link) as the frame, the corresponding instruction can be seen in here (This will require a Intrinsyc login). Of course, you can select other frame on Amazon based on your favor.

2.2. Flight Control Board Assembling

The flight control board is embedded between Snapdragon Flight Board and Frame.

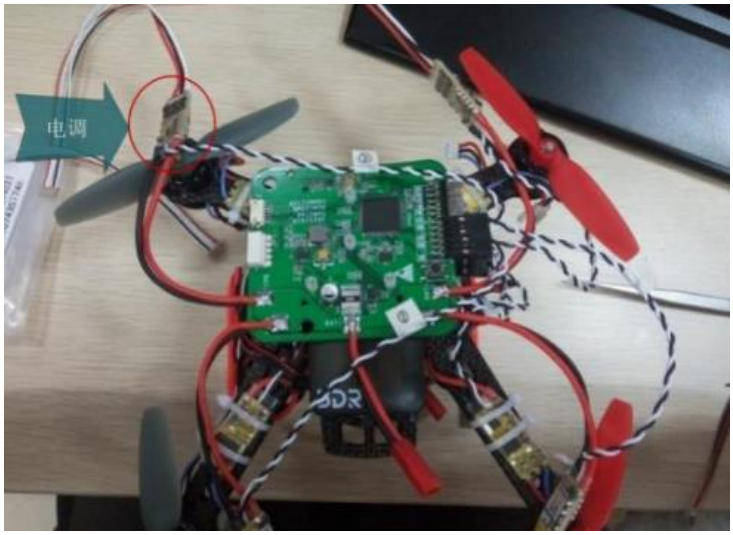
Step 1: Spin out four screws (red circle) and replace them with screws (M2*20mm).



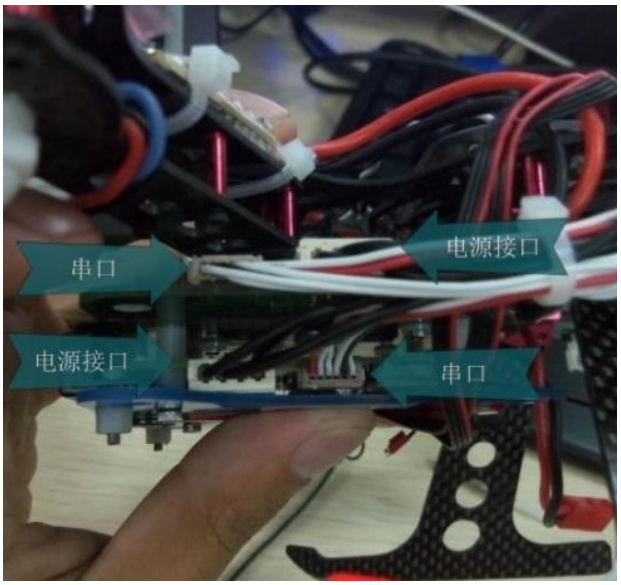
Step 2: Add the double-pass nylon tube between flight control board and Snapdragon Flight board.

Step 3: Fix the flight control board by using the nuts fetched out.

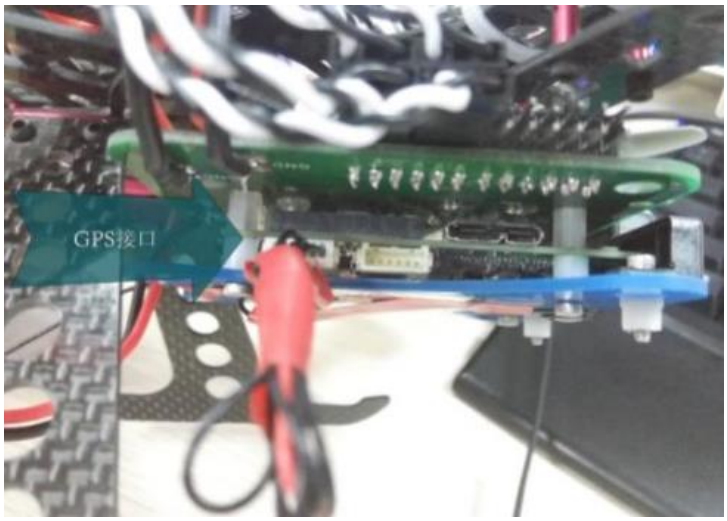
Step 4: Connect the motors with flight control board as follows:



Step 5: Connect the serial ports and power ports of flight control board and Snapdragon Flight board.



Step 6: Mount the GPS module on the frame and connect with the Snapdragon Flight board. Make sure that the arrow on GPS module point to front.



Step 7: Fix the wires on the drone and keep them from twisted by motors.

3. Programming

The programming tutorial is based on Ubuntu 14.04 and ROS Indigo.

3.1. Linux Environment Configuration

Step 1: Permission Setup

```
sudo usermod -a -G dialout $USER
```

Step 2: Toolchain Installation

```
sudo apt-get remove gcc-arm-none-eabi gdb-arm-none-eabi binutils-arm-none-eabi
sudo add-apt-repository ppa:team-gcc-arm-embedded/ppa
sudo apt-get update
sudo apt-get install python-serial openocd \
    flex bison libncurses5-dev autoconf texinfo build-essential \
    libftdi-dev libtool zlib1g-dev \
    python-empy gcc-arm-embedded -y
```

```
sudo apt-get install android-tools-adb android-tools-fastboot fakechroot fakeroot
unzip xz-utils wget python python-empy -y

 unzip xz-utils wget python python-empy -y

git clone https://github.com/ATLFlight/cross_toolchain.git
```

Download the Hexagon SDK 3.0 from here (This will require a QDN login), and move the following files in the download folder of the cross toolchain as follows:

```
mv ~/Downloads/hexagon-sdk-v3-linux.bin cross_toolchain/downloads
```

Install the toolchain and SDK like this:

```
cd cross_toolchain
```

```
./installv3.sh

cd ..
```

After the installation, append the following to your ~/.bashrc

```
export HEXAGON_SDK_ROOT="${HOME}/Qualcomm/Hexagon_SDK/3.0"

export HEXAGON_TOOLS_ROOT="${HOME}/Qualcomm/HEXAGON_Tools/7.2.12/Tools"

export PATH="${HEXAGON_SDK_ROOT}/gcc-linaro-4.9-2014.11-x86_64_arm-linux-

gnueabihf_linux/bin:$PATH"
```

Load the new configuration:

```
source ~/.bashrc
```

Step 3: Sysroot Installation

A sysroot is required to provide the libraries and header files needed to cross compile applications for the Snapdragon Flight applications processor. The qrlSDK sysroot provies the required header files and libraries for the camera, GPU, etc. Download the file Flight_3.1.1_qrlSDK.zip and save it in cross_toolchain/download/.

```
cd cross_toolchain

unset HEXAGON_ARM_SYSROOT

./qrlinux_sysroot.sh
```

Append the following to your ~/.bashrc

```
export HEXAGON_ARM_SYSROOT=${HOME}/Qualcomm/qrlinux_v3.1.1_sysroot
```

Load the new configuration:

```
source ~/.bashrc
```

3.2. Update ADSP firmware

Part of the PX4 stack is running on the ADSP (the DSP side of the Snapdragon 8074). The underlying operating system QURT needs to be updated separately. If anything goes wrong during the ADSP firmware update, your Snapdragon can get bricked! Follow the steps below carefully which should prevent bricking in most cases.

Step 1: Prevent bricking

To prevent the system from hanging on boot because of anything wrong with the ADSP firmware, do the following changes before updating:

Edit the file directly on Snapdragon over adb shell.

```
vim /usr/local/qr-linux/q6-admin.sh
```

Comment out the while loops causing boot to hang:

```
# Wait for adsp.mdt to show up
#while [ ! -s /lib/firmware/adsp.mdt ]; do
#  sleep 0.1
#done
```

and

```
# Don't leave until ADSP is up
#while [ "`cat /sys/kernel/debug/msm_subsys/adsp`" != "2" ]; do
#  sleep 0.1
#done
```

Step 2: Push the latest ADSP firmware files

Download the file Flight_3.1.1a_qcom_flight_controller_hexagon_sdk_add_on.zip from Intrinsyc.

Add copy them on to the Snapdragon:

```
unzip Flight_3.1.1a_qcom_flight_controller_hexagon_sdk_add_on.zip
cd images/8074-eagle/normal/adsp_proc/obj/qdsp6v5_ReleaseG/LA/system/etc/firmware
adb push . /lib/firmware
```

Then do a graceful reboot, so that the firmware gets applied:

```
adb reboot
```

3.3. Building the PX4 software.

Open a terminal and start in the home directory.

Step 1: Download

```
mkdir -p ~/src
cd ~/src
git clone https://github.com/PX4/Firmware.git
cd Firmware
git submodule update --init --recursive
cd ..
```

Step 2: Build and upload it

```
cd Firmware
make eagle_default
make eagle_default upload
adb push ROMFS/px4fmu_common/mixers/quad_x.main.mix  /usr/share/data/adsp
```

After this step, you will copy (and overwrite) the two config files mainapp.config and px4.config to the device. Those files are stored under /usr/share/data/adsp/px4.config and /home/linaro/mainapp.config respectively if you want to edit the startup scripts directly on your vehicle.

Step 3: Run it.

Enter the drone via micro_usb or ssh.

```
./px4 mainapp.config
```

3.4. Receive camera data in ROS

Assume that you have successfully installed ROS and snap_cam,

Step 1: Start video stream on drone side

```
cd catkin_ws
export ROS_MASTER_URI=http://linaro-developer:11311/
source devel/setup.bash
roslaunch snap_cam highres.launch
```

Step 2: Start video receiver on the PC side

```
cd catkin_ws
export ROS_MASTER_URI=http://192.168.1.1:11311/
source devel/setup.bash
rosrun image_view image_view image:=/snap_cam_highres_publisher/image
```