Hummingboard Radio Collect Instructions

1. Unit Descriptions:

Unit 0:
hostname: cubox-00-i
username: aanderson
password: changeme
description: older unit requiring WiFi dongle

ssh example:

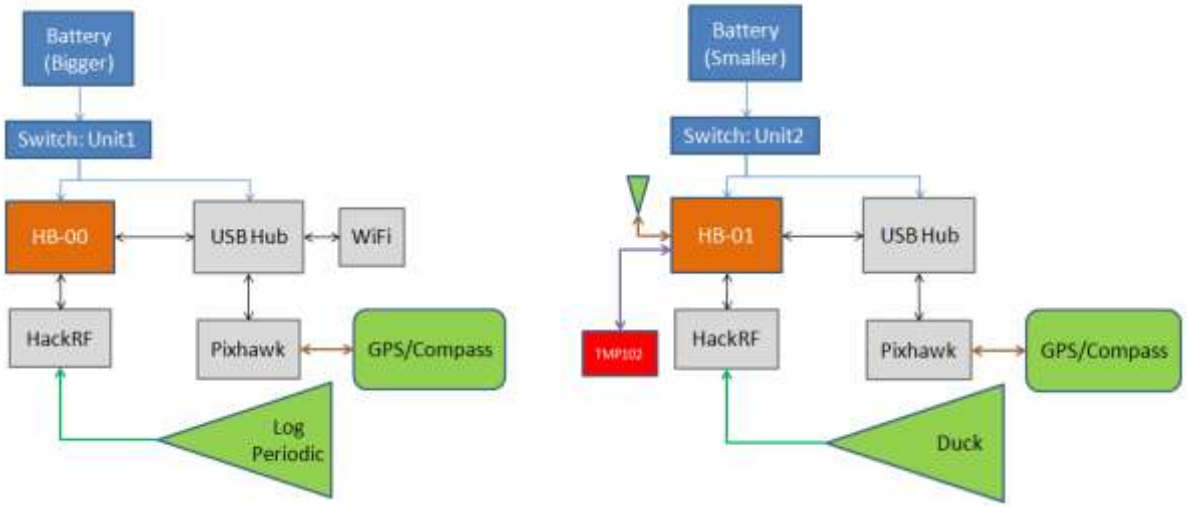
$ ssh aanderson@cubox-00-i.local

Unit 1:
hostname: cubox-01-i
username: aanderson
password: changeme
description: newer unit with integrated WiFi

ssh example:

$ ssh aanderson@cubox-01-i.local

2. Connectivity Diagram:



3. Connecting to the hummingboards:

Currently both boards are configured to connect to a WiFi network with the following credentials:

ssid="cubox_network"
key_mgmt=WPA-PSK
psk="analog_password"

In order to connect to them, you'll need to set your computer to act as a mobile hotspot using with that SSID, password (psk), and security set to WPA-PSK. This is easily done through Ubuntu (and MACs I think).

4. Hummingboard File System Setup:

Both units are equipped with 128GB SSDs. Each unit is configured identically with:
/dev/sda1 -> 10GB partition devoted to swap, automatically mounted at boot up
/dev/sda2 -> remainder of 128GB SSD devoted to data storage
/dev/sda2 -> /mnt/data (mounted to this directory automatically on boot, collected data is written here)

/home/aanderson/ros-sdr -> contains all files relevant for data collect
/home/aanderson/ros-sdr/proto -> contains makefiles for protobuffer code and basic data processing python script: sdr_data_recs.py
/home/aanderson/ros-sdr/catkin_ws -> ROS workspace
/home/aanderson/ros-sdr/catkin_ws/ -> ROS workspace
/home/aanderson/ros-sdr/catkin_ws/src/ros_sdr -> Contains all files relevant to data collect
/home/aanderson/ros-sdr/catkin_ws/src/ros_sdr/src -> has source files relevant to data collect
/home/aanderson/ros-sdr/catkin_ws/src/ros_sdr/scripts -> has data collect tuning scripts, atsc_tuner.py
/home/aanderson/ros-sdr/catkin_ws/launch -> contains bash script to launch collect: balloon_launch.sh

5. Power up Procedure:

1. Flip switch on the side of the box for both units. The batteries should detect load and turn on.
2. Boot up takes ~2 minutes.
3. After 2 minutes SSH into the devices.

6. Launching a Collect:

In the directory:
/home/aanderson/ros-sdr/catkin_ws/launch

run:
$ ./balloon_launch.sh

this will execute the full collect based on this script:

```
#!/bin/bash
source ../devel/setup.bash # Source the setup file before run
roslaunch mavros px4.launch & # launch the mavros script to start roscore and run the pixhawk
sleep 10s # sleep for 10 seconds to let the mavros script startup
rosrun temp_mon temp_mon_node & # run the temperature monitoring node if it's attached
sleep 2s # sleep for 2 seconds to let the temperature monitoring come up
rosrun ros_sdr hackrf_sdr & # turn on the hackrf
sleep 5s # wait 5 seconds while the hackrf comes up and is set in default mode
roslaunch ros_sdr sdr_rec.launch & # launch the recorder, launch file connects relevant topics
```

```
sleep 5s # sleep for 5 seconds while the recorder comes up
rosrun ros_sdr atsc_tuner.py & # start the actual tuner and trigger the start of recordings
```

The sleep times are likely conservative, but recording will start 10+2+5+5=22 seconds this script is launched.

This script is currently configured for an atsc collect. A new recording script will need to be generated, put in the ros_sdr/scripts directory, and the highlighted line in the balloon_launch.sh script will need to be updated!

Once a launch is triggered, simply close the terminal connection. The & call at the end of each line in the bash script means independent threads will be generated and run so even a ctrl+c shouldn't kill the run.

7. Transferring Data Off:

An example command to transfer data is:

local_machine$ scp aanderson@cubox-01-i.local:/mnt/data/atsc-iq-2015-12-XX-XX-XX-XX-part000X.hackrf_data <name of local file.bin>
local_machine$ scp aanderson@cubox-01-i.local:/mnt/data/atsc-iq-XXXXXX.proto <name of local file.proto>

For post processing:
sdr_data_recs.py - can be used to process proto buffer meta data and extract desired IQ runs
FFT_conv_code -> Comming soon! used to process ATSC files in MATLAB

8. Troubleshooting

1. If an error message comes up during the launch sequence relating to resource being busy on pixhawk, power cycle the pixhawk and look for /dev/ttyACM0 to show up.

2. If an error is reported that the recording file cannot be created, look at /mnt/data, check to make sure the directory exists, use command:
$ mount
to confirm /dev/sda2 is mounted properly. If not attempt to mount with:
mount /dev/sda2 /mnt/data
Then confirm permission on /mnt/data, if necessary execute:
chmod 777 /mnt/data

3. If the hackrf is not found try:
$ hackrf_info
if this doesn't report a device found, look in /dev for the hackrf. If it's there, check permissions and modify them if necessary if it's not, try power cycling the hackrf and starting the script again.