**Hummingboard Radio Collect Instructions**

**1. Unit Descriptions:**

Unit 0: ATSC Unit

hostname: cubox-00-i

username: aanderson

password: changeme

description: older unit requiring WiFi dongle

ssh example:

$ ssh aanderson@cubox-00-i.local

Unit 1: LTE Unit

hostname: cubox-01-i

username: aanderson

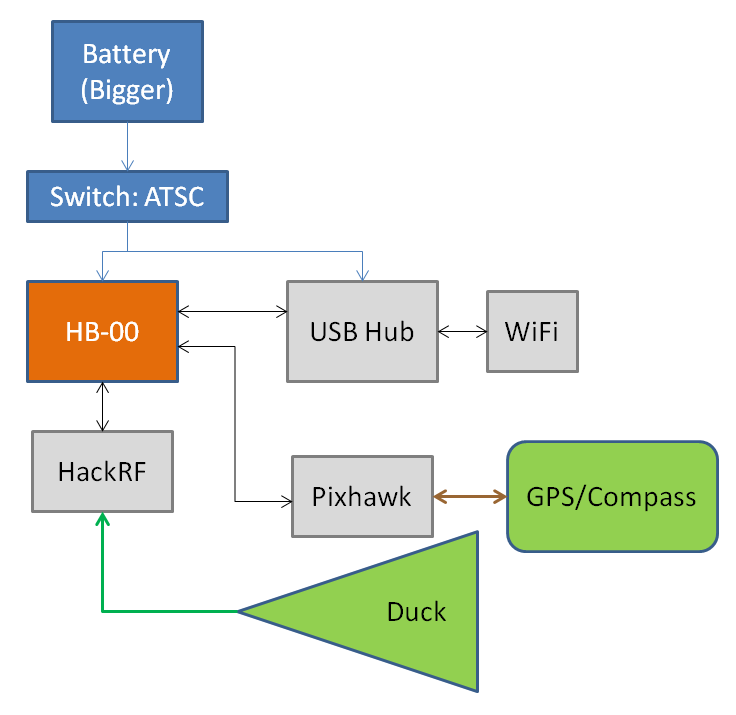
password: changeme

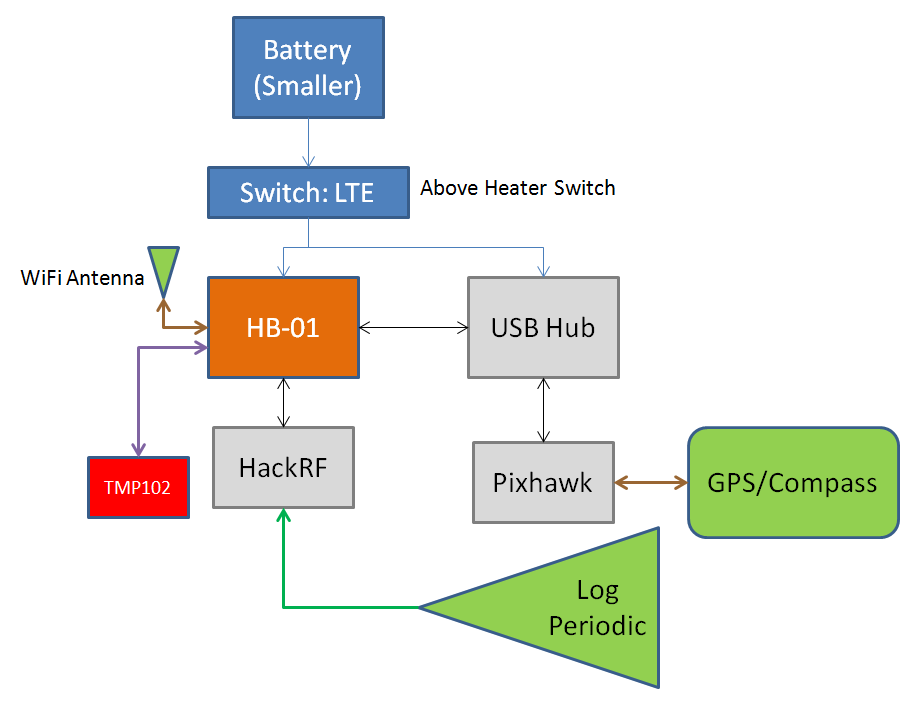
description: newer unit with integrated WiFi

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

/etc/rc.local -> startup script for automatic logging

**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 boot up is complete, the recording starts automatically after another minute.

4. Launch script takes ~30 seconds.

3. After >4 minutes SSH into the devices.

**6. Manually Launching a Collect:**

MANUAL DIRECTIONS ONLY! IGNORE UNLESS ATTEMPTING MANUAL COLLECT! (Collects are automatic)

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:

For ATSC:

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-proto-XXXXXX.proto <name of local file.proto>

For LTE:

local\_machine$ scp aanderson@cubox-01-i.local:/mnt/data/scanner-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/scanner-proto-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

hb\_rad\_col\_proc\_atsc.m -> Coming soon! used to process ATSC files in MATLAB

**8. Troubleshooting**

The data recording is now automatic. The boards will start recording ~3-4 minutes after the switch is flipped. To check health of the recording ssh in and check:

1. $ top -> look for mavros, hackrf\_sdr, sdr\_recording etc. in the list of running programs

-> If you need to kill the processes for any reason use pkill:

$ pkill 12345 -> where 12345 is the number associated with the different running programs

2. To check that mavros is running correctly (the software running INS and GPS collection) run:

$ rostopic echo /mavros/state -> you should see the pixhawk state printed out at ~ 1 Hz intervals

if you only see one state printed out and it hangs, the pixhawk is dead and the cable will need to be re-seated.

-> You can try to just power cycle it and check if this fixd the problem

-> Alternatively kill the running processes, re-seat the cable, and use the manual recording

3. To check if the recording is generally performing alright check if the data files are growing:

$ cd /mnt/data -> to navigate to where the data is being populated

$ watch ls -lh -> and watch the size of the files to make sure the right ones are growing

-> if the data isn't growing that's a bad sign, maybe try to reboot or check permissions on /mnt/data