# RTLSDR Documentation

## Overview

The compute stick will automatically boot when power is applied. There are two modes to the stick. If a RTLSDR stick is connected to the compute stick, it will start in “flight mode”. If there is no RTLSDR stick connected, it will start in “desktop mode”

Username and password is sdrgps/sdrgps

## Flight Mode

Flight mode will start gnss-sdr to calculate real-time GPS position. It will also dump the raw RF I/Q to disk. Position fixes and I/Q dump will terminate after 10 minutes**. Drive space will allow for two full duration flights/tests and a logrotate utility on boot will ensure the outputs are saved from both attempts.** A third attempt will overflow the storage capacity of the device.

The stick nominally takes about 30 seconds to boot.

## Desktop Mode

If no RTLSDR is detected the compute stick will boot as normal without RF recording or GPS fix computation. This mode is used for bench testing and retrieving results.

Connect to a monitor/TV and usb keyboard. Log in and configure internet and you can then transfer files using scp or a usb stick. Once you configure wifi you no longer need the monitor/keyboard and can simply power on the device and use ssh.

## Internet Access

Right now it is configured for my home wifi. To change wifi you will need to use iwconfig. The command looks something like this for WEP

sudo iwconfing wlsp0 essid <your network name> key s:<asci password>

If you have a public network omit s:<password>. If you are using WPA/WPA2 you need to use wpa\_supplicant (let me know and I’ll figure out the command)

To enable Ethernet, edit /etc/network/interfaces and uncomment the two lines in the file. Please note: if you boot the stick without having it connected to Ethernet with Ethernet enabled the device will hang for several minutes waiting for dhcp to timeout. In testing this was up to 5 minutes before I could log in, and the gnss-sdr service should be blocked until the network resolves – beware enabling Ethernet and flying the stick! **I would strongly suggest not enabling Ethernet.**

## Debugging

* If a RTLTCP stick is connected and doesn’t automatically start recording it may be the device string is different. Send me an email with the output of lsusb.

# Use

## Flight or Test

Ensure a RTLSDR stick and antenna are attached prior to boot. RF / GPS tracking will commence for the first 10 minutes of operation. Boot takes ~ 1 minute.

**Would suggest booting up at T-3 to ensure a fix on the pad.**

## Offline Testing

Boot without a RTLSDR stick. Files are located in /home/sdrgps

* RF captures are named “samples?.complex”
* GNSS-SDR files are various and have date/time embedded in the filename

You can replay a RF recording by running

gnss-sdr –config=/usr/local/share/sdrgps/file\_complex.conf

After changing line 8 in the file to point to the correct RF recording.

I will email you a SCP server to upload results to.

# Build and Configuration

## Software

Ubuntu 16.04 server

GNURadio, gnss-sdr built from source using pybombs

Using scripts revision controlled here: <https://github.com/hahnpv/rocketgps>

## Archetecture

A service called “rtltcp” lives in /etc/init.d/rtltcp and is started on boot. This service detects the existence of a RTLSDR stick by grepping the output of lsusb. If the device exists, gnss-sdr is started as a service. The configuration file used for gnss-sdr is located at /usr/local/share/sdrgps/receiver\_complex.conf. The file specified a fixed number of samples to be recorded/processed which terminates execution after 10 minutes. The stick nominally takes about 30 seconds to boot.

All output is to /home/sdrgps

You can rm /home/sdrgps/\*.\* after testing safely. Do not delete the .gnuradio / .volk / .volk\_gnsssdr folders!

# Future Work

The main shortcoming of the current architecture is that it saves gr\_complex samples which are 64 bits (32 bits each for I/Q). The radio only samples 8 bits. Therefore we can reduce the file size by 75% by saving as interleaved bytes. This isn’t straightforward because the Osmocom SDR block (input source from radio) internally converts to gr\_complex. Three options

* Modify the Osmocom block
* Add type conversion and file out to gnss-sdr
* Start a rtl\_tcp server and have gnss-sdr listen along with a separate script that dumps RF to disk in bytes

The second option requires recompiling gnss-sdr since the flowgraph is built in c++.

Third option requires an intermediate script because rtl\_tcp only allows one listener. I found a script that claimed to do this however I could never get it to track properly.