

Teletics Inc.

w\*intercom System

OpenWRT Development



OVERVIEW

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Rev** | **By** | **Description** |
| 31 Jan 2018 | 1.0 | DJS | Draft Release |
| 20 Mar 2018 | 1.2 | Jim | Added section 4.2 for RB411 board development |
| 08 May 2018 1 | 1.3 | Jim | Added Section 4.3 for ZipLine-RB911\_Lite5 board development |
| 08 Jun 2018 | 1.4 | Jim | Added Section 4.4 for Feature Server 4F4E board development |
| 14 Jun 2018 | 1.5 | Dennis | Added Section 5 to hold and keep track of tests (WIP) |
| 01 July 2018 | 1.6 | Jim | Updated section 4.2 for RB411 board Firmware upgrading |
| 05 Dec 2018 | 1.7 | Jim | Added Section 4.3.3 for ZipLine upgrading utility |
| 12 Dec 2018 | 1.8 | Jim | Updated ZipLine Openwrt |
| 21 Jun 2019 | 1.9 | Jim | Added Factory and system upgrading for RB411 on Section 4.2.2/3 |
| 20 Mar 2020 | 1.10 | Jim | Added New board support for RB912 on Section 4.4 |
| 15 May 2020 | 1.11 | Jim | Added another flash chip support for RB912 on Section 4.4.2 |
| 02 June 2020 | 1.12 | Jim | Update development for RB912 board on section 4.4 |
|  |  |  |  |

Contents

[1 Introduction 1](#_Toc42027248)

[1.1 Objective 1](#_Toc42027249)

[1.2 Background 1](#_Toc42027250)

[2 System Overview 1](#_Toc42027251)

[2.1 Server 1](#_Toc42027252)

[2.1.1 Feature Server 1E 1](#_Toc42027253)

[2.1.2 Feature Server 4F1E 2](#_Toc42027254)

[2.1.3 Feature Server 4F4E 2](#_Toc42027255)

[2.1.4 Feature Server 4E 2](#_Toc42027256)

[2.2 Station 3](#_Toc42027257)

[2.2.1 Boards 3](#_Toc42027258)

[3 Cloud System 3](#_Toc42027259)

[3.1 Client 4](#_Toc42027260)

[3.2 Cloud Server 4](#_Toc42027261)

[3.2.1 Message Definition 4](#_Toc42027262)

[3.2.2 Authentication 4](#_Toc42027263)

[4 OpenWRT Development 5](#_Toc42027264)

[4.1 Prerequisites 5](#_Toc42027265)

[4.1.1 git 5](#_Toc42027266)

[4.1.2 dnsmasq 5](#_Toc42027267)

[4.1.3 HTTP server 5](#_Toc42027268)

[4.1.4 Other modules needed for build Openwrt 5](#_Toc42027269)

[4.2 RB-411 board 5](#_Toc42027270)

[4.2.1 Build images 5](#_Toc42027271)

[4.2.2 Factory Firmware image upgrading 6](#_Toc42027272)

[4.2.3 System upgrading 7](#_Toc42027273)

[4.2.4 Feature test 7](#_Toc42027274)

[4.3 ZipLine RB911\_AC Board 8](#_Toc42027275)

[4.3.1 Build image 8](#_Toc42027276)

[4.3.2 Upgrade Image 8](#_Toc42027277)

[4.3.3 Upgrade Image by Utility 9](#_Toc42027278)

[4.4 RB912UAG-5HPnD Board 9](#_Toc42027279)

[4.4.1 Build image for ram init 9](#_Toc42027280)

[4.4.2 Build image for w25x05 flash chip 9](#_Toc42027281)

[4.4.3 Build image for gd25d05 10](#_Toc42027282)

[4.4.4 Upgrade Image 10](#_Toc42027283)

[4.4.5 Upgrade Image by Utility 10](#_Toc42027284)

[4.4.6 Sportsman’s Corner Campground 10](#_Toc42027285)

[4.5 Feature Server 4F4E board 11](#_Toc42027286)

[4.5.1 Build image 11](#_Toc42027287)

[4.5.2 Upgrade Image 11](#_Toc42027288)

[4.5.3 Features Test: 12](#_Toc42027289)

[5 Testing 12](#_Toc42027290)

[5.1 Feature Server 12](#_Toc42027291)

[5.1.1 Networking (4F4E) 12](#_Toc42027292)

[5.1.2 Asterisk Testing 12](#_Toc42027293)

[5.1.3 Python Testing 12](#_Toc42027294)

[5.2 Radio 12](#_Toc42027295)

[5.2.1 Networking Overview 13](#_Toc42027296)

[5.2.2 SSH Testing 14](#_Toc42027297)

# Introduction

Teletics is working on a review and formalization of the OpenWRT based Server and Station for the w\*intercom wireless intercom system.

This is a living document and will be updated regularly during the development project.

## Objective

The objective of this document is to provide information for the development of OpenWRT firmware in the Server and Station of the w\*intercom system and integration with the Harness server.

## Background

The w\*intercom product has evolving over the past several years. It is now time to review, accept and document the w\*intercom system hardware and software.

# System Overview

The w\*intercom System is a standalone system providing local communications around, and to a site.

There are 2 basic components of the system, a Server and a Station.



Figure - Typical w\*intercom System block diagram

## Server

There are several versions of the Server.

### Feature Server 1E

The Feature Server 1E is a small table top embedded computer that provides call management for the w\*intercom System.

The Feature Server has one RS-232 port for diagnostics and one Ethernet port for network communication.



#### Boards

CPU PC Engines Alix 3D2 Geode Compact Flash

http://pcengines.ch/alix3d2.htm

### Feature Server 4F1E

The Feature Server 4F1E is a rack mount embedded computer that provides call management for the w\*intercom System.

The Feature Server 4F1E has one RS-232 port for diagnostics, one Ethernet port for network communication and four FXO ports for POTS line connections.



#### Boards

CPU PC Engines Alix 3D2 Geode Compact Flash

http://pcengines.ch/alix3d2.htm

FXO OpenVox A400M

http://openvox.cn/products/telephony-cards/analog-cards/133-a400-series

### Feature Server 4F4E

The Feature Server 4F4E is a rack mount embedded computer that provides call management for the w\*intercom System.

The Feature Server 4F4E has one RS-232 port for diagnostics, four Ethernet ports for network communication and four FXO ports for POTS line connections.



#### Boards

CPU Soekris 5501 Geode Compact Flash

http://soekris.com/media/manuals/net4801\_manual.pdf

http://soekris.com/media/manuals/net5501\_mech.pdf

FXO Atcom AX400P

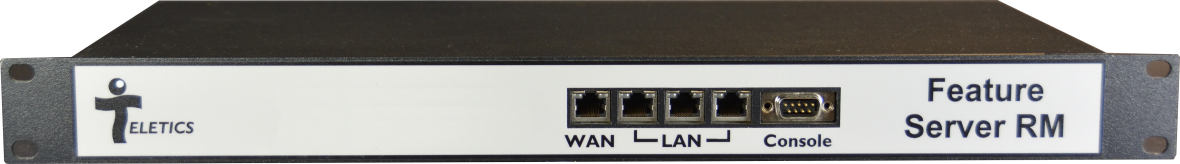
http://www.atcom.cn/uploadfile/2014/1219/cards/ax400p/AX400P&AXE400P%20Datasheet.pdf

http://www.atcom.cn/uploadfile/2014/1219/cards/ax400p/AX-400P-Ubuntu%208.10-dahdi-User%20Manual-V1.0-EN.pdf

### Feature Server 4E

The Feature Server 4E is a rack mount embedded computer that provides call management for the w\*intercom System.

The Feature Server 4E has one RS-232 port for diagnostics and four Ethernet ports for network communication.



#### Boards

CPU Soekris 5501 Geode Compact Flash

http://soekris.com/media/manuals/net4801\_manual.pdf

http://soekris.com/media/manuals/net5501\_mech.pdf

## Station

A Station consists of a radio, ATA and power board.

Up to 20 Stations can be used in a wintercom System.

### Boards

#### Radio (old)

CPU Microtik RB411

https://mikrotik.com/product/RB411

Radio DBii Pro F-50

Serial RS-232

#### Radio (new 2018)

Microtik RB912AG-5HPND single board radio

https://mikrotik.com/product/RB912UAG-5HPnD

MiMo radio with one channel disabled

#### ATA (old)

Grand Stream HT502 Analog Telephone Adapter

http://www.grandstream.com/products/gateways-and-atas/analog-telephone-adaptors/product/handytone-502

#### ATA (new 2018)

Grand Stream HT812 Analog Telephone Adapter

http://www.grandstream.com/products/gateways-and-atas/analog-telephone-adaptors/product/ht812

# Cloud System

In the Cloud system, remote monitoring and support is provided through the Cloud Server (Harness). Harness communicates with the Feature Server (FS) to provide historical and semi-real-time information.



Figure - Cloud System block diagram

## Client

A client is anyone that accesses Harness through the internet. This includes:

* Teletics Admin
* Teletics Site Management
* Customer Site Management

## Cloud Server

### Message Definition

Requests will be received by Harness from two sources:

* Client
* Feature Server

Harness will manage all time related scheduling

#### Client

The client will request pages from Harness through a traditional web interface.

#### Feature Server

In a perfect world, Harnessthe CS would talk directly to the FS. As the FS is behind a firewall in almost all instances, we will use the FS to initiate all sessions with a next message response.

The FS sends a request with the instructions from the last CS response along with the corresponding data.

### Authentication

There will be several levels of Harness access. These are …

1. Developer
2. Admin
3. Customer
4. Site

#### Developer

Developer access allows the modification the website and associated files.

#### Admin

Admin access allows the creation and management of customers and related parameters.

#### Customer

Customer access allows the request and display of site related parameters.

#### Site

Site access is provided by the existing Teletics software tools.

# OpenWRT Development

## Prerequisites

### git

git will be used for updating and checking out the source code from git repository

sudo apt install git

### dnsmasq

dnsmasq will be dhcp server and tftp server for RB-411 upgrading.

sudo apt install dnsmasq

### HTTP server

sudo apt-get install apache2

### Other modules needed for build Openwrt

sudo apt-get install git-core build-essential libssl-dev libncurses5-dev unzip gawk zlib1g-dev

## RB-411 board

The source code for RB411 is based on Openwrt git repository commit fb1be20d63f08a07c63bcd5ffb26d010935dfa61.

### Build images

**Step1:** git clone [git@git.assembla.com](mailto:git@git.assembla.com):teletics/teletics-openwrt.3.git

or

git clone https://git.assembla.com/teletics/teletics-openwrt.3.git

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_radio\_RB411 .config

**Step4:** make menuconfig (don’t need change anything, just save and close)

**Step5:** make V=99

**Step6:** cp config\_ramImage .config

**Step7:** make menuconfig (don’t need change anything, just save and close)

**Step8:** make V=99

After image build is done, the images can be found at ./eletics-openwrt.3/bin/targets/ar71xx/mikrotik

* radio-v2.0.0-ar71xx-mikrotik-vmlinux-lzma.elf for factory kernel upgrading
* radio-v2.0.0-ar71xx-mikrotik-rootfs.ubifs for factory root file system upgrading
* openwrt-ar71xx-mikrotik-vmlinux-initramfs-lzma.elf for factory root file system upgrading

### Factory Firmware image upgrading

* cp openwrt-ar71xx-mikrotik-vmlinux-initramfs-lzma.elf upgrade-rb411/.
* In order to avoid root file system crash, kernel upgrading will erase root file system. So the kernel upgrading should be the first step.
* The firmware which includes system upgrading features only can be graded by Factory image upgrading

#### Factory Kernel upgrading

On Host PC

* cd teletics-openwrt.3/upgrade-rb411
* ./upgrade-rb411-kernel.sh

On console port of feature server 4F4E

* restart RB411 and boot with set boot device to “e – boot over Ethernet”

Select boot device:

\* e - boot over Ethernet

  n - boot from NAND, if fail then Ethernet

  1 - boot Ethernet once, then NAND

  o - boot from NAND only

  b - boot chosen device

  f - boot Flash Configure Mode

  3 - boot Flash Configure Mode once, then NAND

* Wait till OpenWrt reboot
* Restore RB411 boot device to “n - boot from NAND, if fail then Ethernet”

#### Factory root file system upgrading

The image layout in RB411 board is

dev: size erasesize name

mtd0: 0000b000 00001000 "routerboot"

mtd1: 00001000 00001000 "hard\_config"

mtd2: 00002000 00001000 "bios"

mtd3: 00001000 00001000 "soft\_config"

mtd4: 00040000 00020000 "booter"

mtd5: 003c0000 00020000 "kernel"

mtd6: 1fc00000 00020000 "ubi"

**On Host PC**

* cd teletics-openwrt.3/upgrade-rb411
* ./upgrade-rb411-roofs.sh

**On console port of feature server 4F4E**

* restart RB411 and boot with set boot device to “e – boot over Ethernet”

Select boot device:

\* e - boot over Ethernet

  n - boot from NAND, if fail then Ethernet

  1 - boot Ethernet once, then NAND

  o - boot from NAND only

  b - boot chosen device

  f - boot Flash Configure Mode

  3 - boot Flash Configure Mode once, then NAND

* Wait till OpenWrt reboot
* Restore RB411 boot device to “n - boot from NAND, if fail then Ethernet”

### System upgrading

On Host PC

* System upgrading image located at ./bin/targets/ar71xx/mikrotik/
* System upgrading filename: Teletics-RB411-sysupgrade + FW Version + build Time (UTC) + build Date + “.bin”

ex: Teletics-RB411-sysupgrade-v2.0.0.1-173708-2019-06-21.bin

* cp Teletics-RB411-sysupgrade-v2.0.0.1-173708-2019-06-21.bin /var/www/html

On RB411 console port

* sysupgrade\_rb411 http://169.254.4.200/Teletics-RB411-sysupgrade-v2.0.0.1-173708-2019-06-21.bin

### Feature test

#### VLAN test

added vlan tag 200 to both end of Ubuntu and RB411

eth0 192.168.6.0/255.255.0.0

eth0.200 192.169.200.0/255.255.0.0

ping from Ubuntu

In RB411 openwrt command line:

tcpdump -i eth0 -e

Observe the output:

5:17:29.127748 00:1d:72:b9:ab:3c (oui Unknown) > 4c:5e:0c:c5:10:94 (oui Unknown), **ethertype 802.1Q (0x8100), length 102: vlan 200**, p 0, ethertype IPv4, 192.169.200.100 > 194

05:17:29.128116 4c:5e:0c:c5:10:94 (oui Unknown) > 00:1d:72:b9:ab:3c (oui Unknown), **ethertype 802.1Q (0x8100), length 102: vlan 200**, p 0, ethertype IPv4, 192.169.200.10 > 1924

#### Mesh Test

* Set two RB411 with mesh mode
* Set same Key and SSID in both end.
* Test bandwidth

server side:

iperf -s -p 2000

client side:

iperf -c serverIP -p 2000 -i 2 -t 30

Observe the output for bandwidth

------------------------------------------------------------

Server listening on TCP port 2000

TCP window size: 85.3 KByte (default)

------------------------------------------------------------

[ 4] local 192.169.200.10 port 2000 connected with 192.169.200.100 port 44424

[ ID] Interval Transfer Bandwidth

[ 4] 0.0-10.0 sec 24 Mbits/sec

## ZipLine RB911\_AC Board

### Build image

**Step1:** git clone -b ZipLine-rb911\_ac [git@git.assembla.com:teletics/teletics-openwrt.3.git](mailto:git@git.assembla.com:teletics/teletics-openwrt.3.git)

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_ram\_RB911ac .config

**Step4:** make menuconfig

**Step5** save and Exit from menuconfig

**Step6:** make V=99

**Step7:** cp config\_RB911ac .config

**Step8:** make menuconfig

**Step9:** make V=99

### Upgrade Image

There are two files can be found at teletics-openwrt.3/bin/targets/ar71xx/mikrotik after OpenWrt build successfully.

* openwrt-ar71xx-mikrotik-nand-large-ac-squashfs-sysupgrade.bin
* openwrt-ar71xx-mikrotik-vmlinux-initramfs-lzma.elf

**On Host PC:**

**cp openwrt-ar71xx-mikrotik-vmlinux-initramfs-lzma.elf ../upgrade-rb911**

**cd ../upgrade-rb911**

**./upgrade-rb911.sh**

**On RB911 board:**

* Press the button on RB-911\_AC board and plug in the power supply.
* Release button until all LED are off

### Upgrade Image by Utility

Upgrading utility should have DHCP server, tftp server and http server

Step 1: Start DHCP server in utility

Step 2: Set dhcp boot as RAM init image which built at section 4.3.1

Step 3: copy latest image to http server directory

Step 4: SSH to ZipLine board

IP address of ZipLine AP : 169.254.4.11

IP address of ZipLine Remote : 169.254.2.11

Step 5: Type “mtd erase kernel” in SSH console

Step 6: Type “reboot” in SSH console

After successfully upgrade, The IP address will 169.254.2.10

Step 7: SSH to 169.254.2.10

Type

“cd /root”

“./ETH\_program\_as\_master\_trunk” for ZipLine AP

“./ETH\_program\_as remote\_trunk” for ZipLien Remote

“reboot”

## RB912UAG-5HPnD Board

### Build image for ram init

**Step1:** git clone -b rb912 git@git.assembla.com:teletics/teletics-openwrt.3.gitgit@git.assembla.com:teletics/teletics-openwrt.3.git

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_ram\_RB912 .config

**Step4:** make menuconfig

**Step5:** save and Exit from menuconfig

**Step6:** make V=99

### Build image for w25x05 flash chip

**Step1:** git clone -b rb912 git@git.assembla.com:teletics/teletics-openwrt.3.gitgit@git.assembla.com:teletics/teletics-openwrt.3.git

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_RB912 .config

**Step4:** rm –rf target/linux/ar71xx/patches-4.9/960-routerboot.patch

**Step5:** make menuconfig

**Step6:** save and Exit from menuconfig

**Step7:** make V=99

**Step8:** cp bin/target/ar71xx/mikrotik/openwrt-ar71-mikrotik-nand-large-ac-squashfs-sysupgrade.bin upgrade-rb912/.

### Build image for gd25d05

**Step1:** git clone -b rb912 git@git.assembla.com:teletics/teletics-openwrt.3.gitgit@git.assembla.com:teletics/teletics-openwrt.3.git

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_RB912 .config

**Step4:** make menuconfig

**Step5:** save and Exit from menuconfig

**Step6:** make V=99

### Upgrade Image

**On Host PC:**

**Warning: Since there are two types of NOR flash is used on RB912, gd25d05 and w25x05. We do not know which flash was used before we program OpenWrt. The upgrade script will auto detect and upgrade different image for different boards.**

**Step 1: cd ../teletics-openwrt.3/upgrade-rb912**

**Step 2:** **./upgrade-rb912.sh (this script will get latest built image and upgrade it to target board )**

**On RB912 board:**

* Press the button on RB912 board and plug in the power supply.
* Release button until all LED are off

### Upgrade Image by Utility

Upgrading utility should have DHCP server, tftp server and http server

Step 1: Start DHCP server in utility

Step 2: Set dhcp boot as RAM init image

Step 3: copy latest image which built on 4.4.1 to http server directory

Step 4: SSH to RB912 board

Step 5: Type “mtd erase kernel” in SSH console

Step 6: Type “reboot” in SSH console

After successfully upgrade, The IP address will 169.254.2.10

Step 7: SSH to 169.254.2.10

Type

“cd /root”

“./set-camp0-campground-office” for camp0 office. This node need connect Internet as a gateway

or

“./set-camp71-site-c” for other node

“reboot”

### Sportsman’s Corner Campground

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Radio** | **SSID** | **Password** | **Num User** | **IP Addr** | **Comment** |
| Radio “0” Campground office | Campers\_0 | camping1 | 20 | 169.254.0.1 | Gateway,  Connect to Internet |
| SCC\_0 | owner246 | 5 | 169.254.1.1 |
| Radio “71” Owners Site “C” | Campers\_71 | camping1 | 20 | 169.254.71.1 | Client |
| SCC\_71 | owner246 | 5 | 169.254.72.1 |
| Radio “44” Washroom near site 44 | Campers\_44 | camping1 | 20 | 169.254.44.1 | Client |
| SCC\_44 | owner246 | 5 | 169.254.44.1 |
| Radio “83” Washroom near site 83 | Campers\_83 | camping1 | 20 | 169.254.83.1 | Client |
| SCC\_83 | owner246 | 5 | 169.254.84.1 |
| Radio “28” Installation near site 28 | Campers\_28 | camping1 | 20 | 169.254.28.1 | Client |
| SCC\_28 | owner246 | 5 | 169.254.29.1 |

## Feature Server 4F4E board

### Build image

**Step1:** git clone -b fs-4F4E [git@git.assembla.com:teletics/teletics-openwrt.3.git](mailto:git@git.assembla.com:teletics/teletics-openwrt.3.git)

**Step2:** cd teletics-openwrt.3

**Step3:** cp config\_ram\_fs-4F4E .config

**Step4:** make menuconfig

**Step5** save and Exit from menuconfig

**Step6:** make V=99, then wait until built done

**Step7**: cp config\_fs-4F4E .config

**Step8:** make menuconfig

**Step9** save and Exit from menuconfig

**Step10:** make V=99

### Upgrade Image

There are two files can be found at teletics-openwrt.3/bin/targets/x86/geode after OpenWrt build successfully.

* openwrt-x86-geode-ramfs.bzImage
* openwrt-x86-geode-combined-squashfs.img.gz

Before upgrade image make sure the dnsmasq and httpd already installed on the host machine

**On Host PC**

* cp openwrt-x86-geode-ramfs.bzImage teletics-openwrt.3/upgrade\_fs-4F4E/.
* cp openwrt-x86-geode-combined-squashfs.img.gz /var/www/. (The folder for web server)
* cd teletics-openwrt.3/upgrade\_fs-4F4E
* ./upgrade-fs4F4E.sh

**On console port of feature server 4F4E**

* set console baudrate to 19200
* Restart feature server and press “ctrl + p ” to enter comBios of the feature server
* Enter “boot F0”

Feature server will boot up by ram file system and will auto update to new image “openwrt-x86-geode-combined-squashfs.img.gz”

### Features Test:

* Port0 and Port1 will start with DHCP client. When connect with DHCP server, it will get IP address.
* Port2 and Port3 will run with DHCP server and will assign IP address to client from “169.254.0.1” to “169.254.0.251”

# Testing

## Feature Server

### Networking (4F4E)



Numbered according to how the ports correspond to their interface names in OpenWRT’s settings.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Port | Intended Functionality | IP Address | Subnet Mask | Gateway | DNS |
| eth0 | DHCP Server | 169.254.0.1 to 169.254.0.251 | 255.255.0.0 | 169.254.5.200 | 169.254.200 |
| eth1 | DHCP Server | 169.254.0.1 to 169.254.0.251 | 255.255.0.0 | 169.254.5.200 | 169.254.200 |
| eth2 | DHCP Client | 10.0.0.X | 255.255.255.0 | 10.0.0.254 | 199.185.220.254 and 198.80.55.5 |
| eth3 | DHCP Client | 10.0.0.X | 255.255.255.0 | 10.0.0.254 | 199.185.220.254 and 198.80.55.5 |

Do note that the DHCP Client results will vary depending on the environment that it is tested in. The data in this table is what we would expect to be provided in the usual environment for the Teletics office.

### Asterisk Testing

### Python Testing

## Radio

### Networking Overview

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Connection Type | Static IP | Ping | Basic SSH | iperf | Extended SSH | VLAN |
| Wired |  |  |  |  |  |  |
| Wireless |  |  |  |  |  |  |

Our radios communicate over a mesh, so we wanted to ensure that we tested on stations directly connected, as well as remote stations connected through the mesh.

### SSH Testing

Opening a basic SSH session and sending commands works fine. However, when we have a long SSH session where we repeatedly send commands, the session is forcibly closed. There is also another case where if we try to send too large of a command at once, the radio outright refuses us. We are currently using dropbear and believe that it may be what is causing the issue. We intend to try OpenSSH to see if that resolves some of these issues.

|  |  |  |
| --- | --- | --- |
| Dropbear | | |
| Test Application | Status | Comments |
| TUtil Signal Screen |  | Time can vary, but generally ~8 hrs.  Generally receive **connection forcibly closed by remote host**. |
| Standalone SSH 3.0 |  | We strayed from TUtil to remove overhead, and avoided using WeOnlyDo. We login and logout each time. |
| Putty |  | An ash script in the radio loops commands in a putty session |
| Plink |  | A batch script loops opening the SSH session, sending commands then closing the session before going again. |
| Standalone SSH 4.0 |  | Similar to 3.0 but does not login and logout, just continuously sends the commands using the same session.  The SSH application window simply hangs. |
| Plink 20x |  | Similar to Plink but rather than sending the two commands just once, we sent them 20 times. |
| Plink 30x |  | I don’t quite remember what error we got here. |
| Plink 40x |  | Errors with **connection forcibly closed by remote host**. |