HamSCI

TAPR

PERSONAL SPACE WEATHER SYSTEM

LOCAL HOST

DETAILED DESIGN SPECIFICATIONS

Version Number: 0.1

Version Date: Nov. 15, 2019

VERSION HISTORY

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
0.1	W. Engelke	*11/15/2019*			Added Odroid installation & config details; Added LH – DE communication concept, Appendix A; Changed planned communication between DE and SBC to UDP; Added system architecture, with explanation of major processes; Added instructions for creating/adapting GBURadio OOT Modules; Added details on FT8 monitoring and decoding; Added details for Use Case 3 (Low Bandwidth data acquisition). Removed the following: Approvals; localization; logical data model; Requirements Traceability; Removed note on accessibility – it will not be included in Phase 1

CONTENTS

CONTENTS **3**

1 INTRODUCTION **5**

2 USE CASES **6**

2.1. about this section 6

2.2. UI Use Cases 6

2.2.1. USE CASE 1 – Initial Screen and (Optional) Log-in 6

2.2.2. USE CASE 2 – Tangerine Welcome and Configuration Screen 8

2.2.3. USE CASE 3 – Configuration 10

2.2.4. USE CASE 4 – Set Up Data Engine 12

2.2.5. USE CASE 4A – Channel / Antenna Setup 13

2.2.6. USE CASE 5 – Propagation Monitoring 14

2.2.7. USE CASE 6 – Callsign Watching 15

2.2.8. USE CASE 7 – Set Up / Test Magnetometer 16

2.2.9. USE CASE 8 - Start Data Collection / Stop Data Collection 16

2.2.10. USE CASE 9 – Central Control System 17

2.2.11. USE CASE 10 – Data Uploading Control 17

TECHNICAL DETAILS **18**

3 18

3.1 General 18

3.1.1 Operating System 18

3.1.2 Packages to install 18

3.1.3 Communication with Data Engine 19

3.1.4 Data Storage Format: HDF5 and Digital RF 19

3.1.3 Topics for further development include: 20

4 SYSTEM ARCHITECTURE **21**

4.1 Processes 22

4.1.1 Central Request / Response / Upload (standard Data Collection) 22

4.1.2 Heartbeat 23

4.1.3 Large Local Server Data Collection (Use Case 2: Firehose) 24

4.1.4 Low bandwidth multiband monitoring 24

4.1.5 Local propagation analysis and User Notifications 25

4.2 Logging 26

5 HELP **26**

6 REMOTE SYSTEM UPDATES **27**

7 TECHNICAL NOTES 27

1 INTRODUCTION

This Detailed Design Specification describes how the Local Host computer will be used (user interface, or UI), as well as its detailed internal design. It is organized into two parts:

- **Use Cases**, which illustrate the user's experience, how the system will work behind the scenes to execute the user's wishes, and processes
- **Technical Details**, which describe the technical approach, software tools, configurations, required software packages, interoperations, and interconnections required to carry out the required functionality.

Figure 1 shows an overview of the system architecture. This document is concerned with the detailed UI, operation, and technical considerations of the Local Host (SBC).

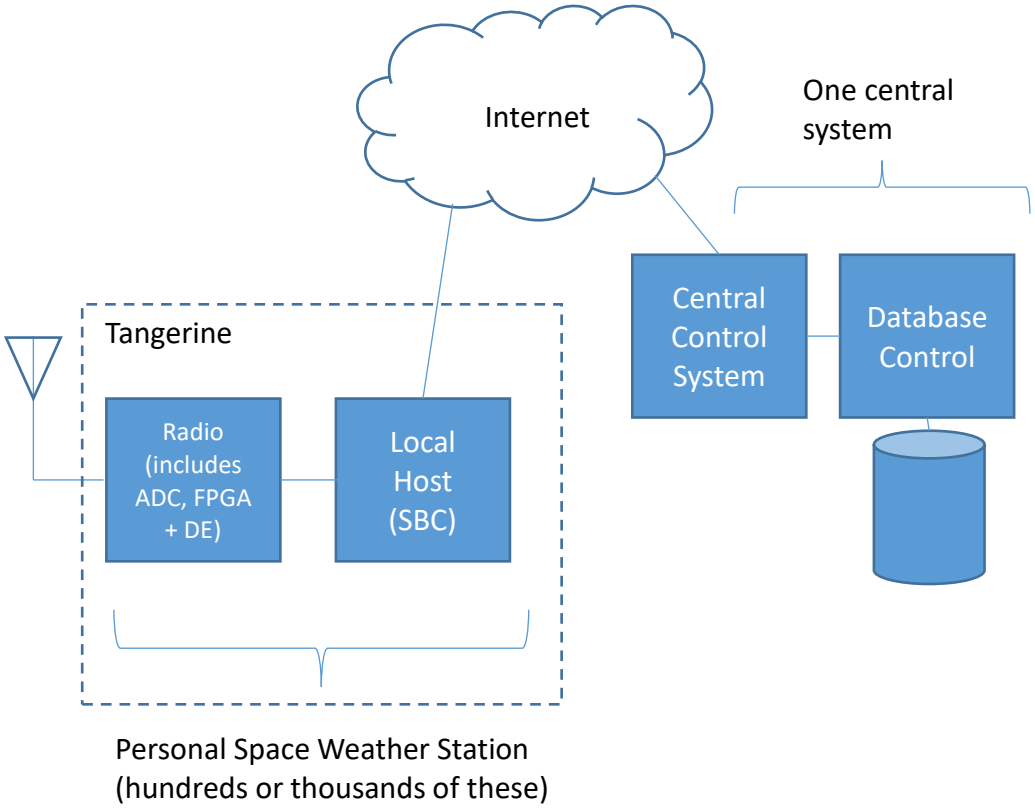


Figure 1. Conceptual Overview.

2 USE CASES

2.1. ABOUT THIS SECTION

The Use Cases section starts with showing how the user will interact with the Local Host through the UI and continues on to describe the automated features that operate on the Local Host as background processes not directly viewable by the user.

2.2. UI USE CASES

This section describes User Interface uses cases, focusing on the user experience, with references to sections in Technical Details on planned approaches for how to implement the described functionality. The Use Cases described here have to do with the various functions of the Local Host for controlling the Tangerine (and should not be confused with the two major “Use Cases” of how the Tangerine saves spectrum data to either the Local Host or to a large local server.)

Note: the Initial Screen / Log-in use cases are standard Ubuntu UI, and are included here only for completeness.

2.2.1. USE CASE 1 – Initial Screen and (Optional) Log-in

The Local Host supports (at minimum) a keyboard, mouse and screen. Upon boot-up the system automatically starts a user interface; depending on previous settings, there are several possible screen displays.

2.2.1.1. System Locked

In the case where (a) the user has previously locked the system or (b) security settings are set to automatically lock the screen, the O/S log-in is shown, which requires a user to provide a password. This is designated as “Optional” because the user may have previously disabled automatic screen locking. (Standard feature of Ubuntu). See Figure 2. Note: in the default configuration (included in the Hardkernel Ubuntu distribution, the UID = odroid, PW = odroid)

When the user enters the correct password, the system proceeds to show the standard Ubuntu desktop.

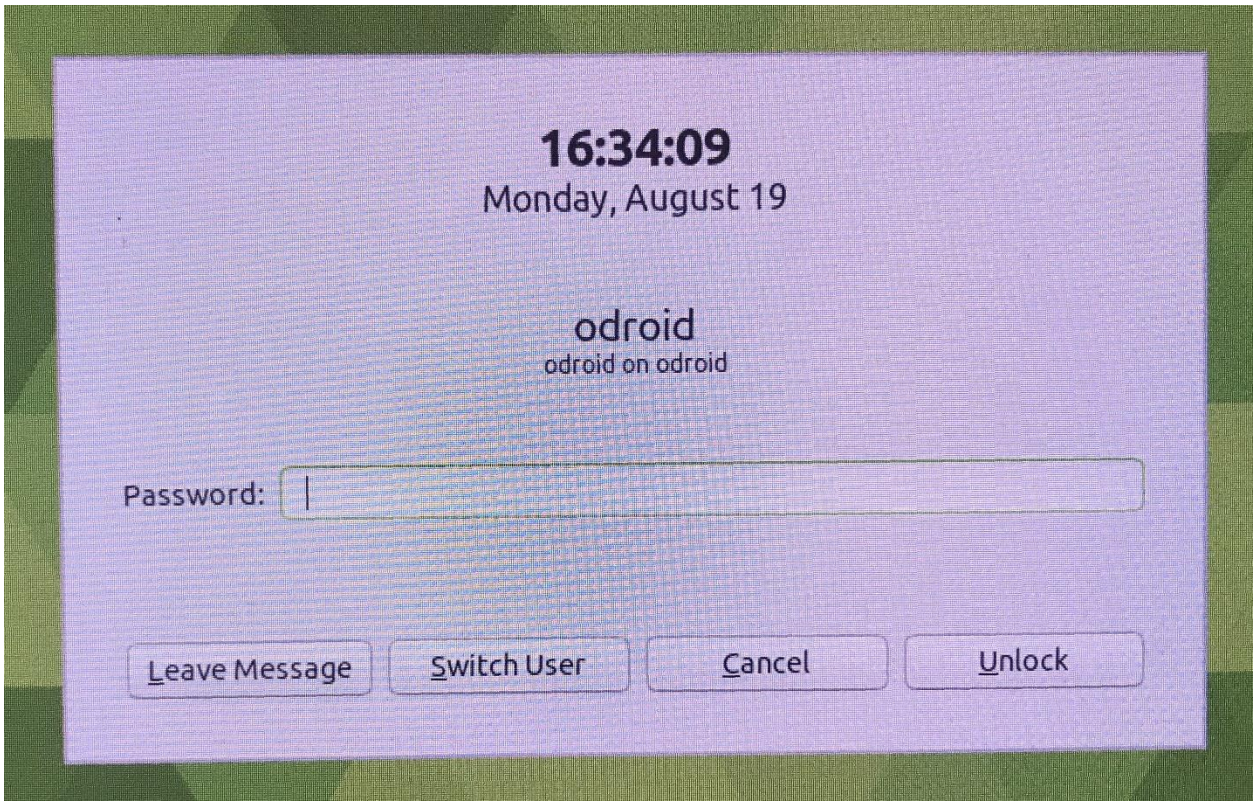


Figure 2. Initial screen when system locked.

2.2.1.2. System Unlocked

In the case where the system is not in a locked mode, the standard log-in shows as a simple button. See Figure 3. When the user clicks the Log In button, the system proceeds to show the standard Ubuntu desktop.

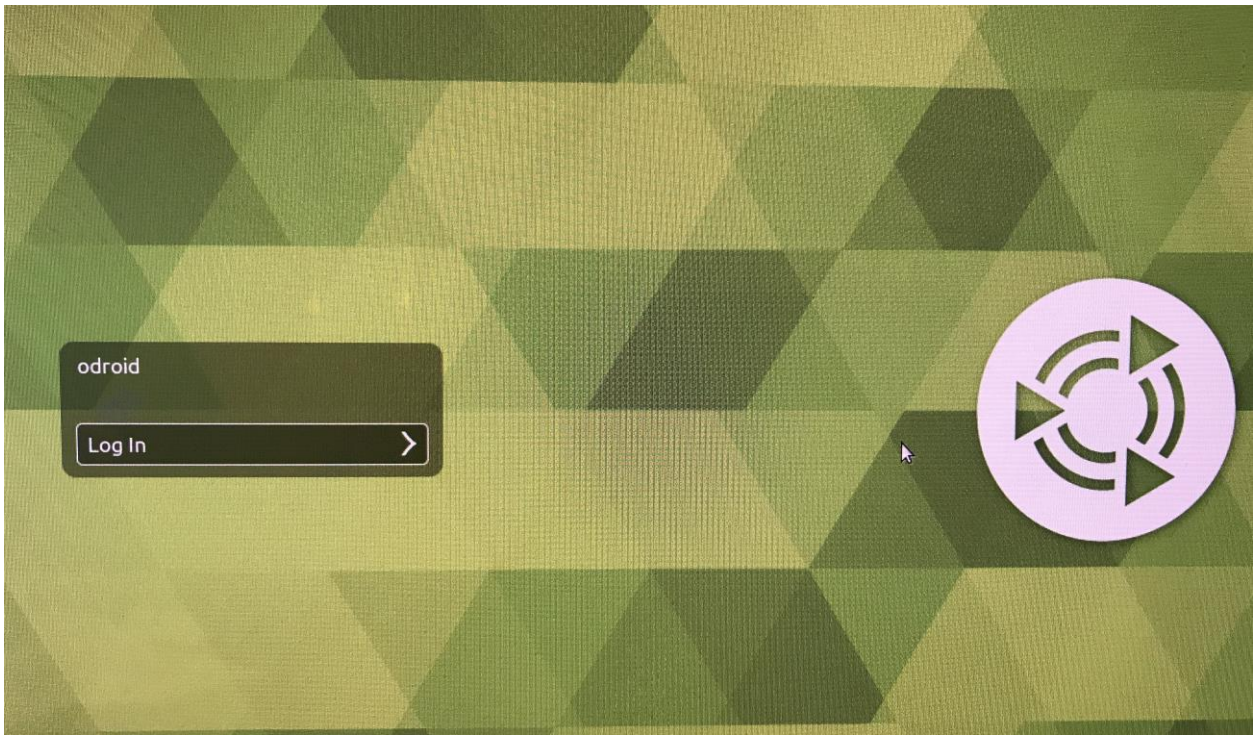


Figure 3. Non-locked system Log In screen.

The remainder of the Ubuntu UI will not be described here, as it is not expected to be changed as part of the PSWS Local Host functionality.

Further user interactions with the system will be accomplished primarily through use of the installed web browser (standard Ubuntu browser being Firefox), which communicates to a web site installed on the Local Host.[1]

2.2.2. USE CASE 2 – Tangerine Welcome and Configuration Screen

The Tangerine Welcome Screen will be reachable by a user who has a keyboard, mouse, and screen connected directly to the SBC, and also by a user on the same network as the Tangerine who is using a separate computer and browser. The SBC subsystem will listen on TCP Port 80, acting as a web site.

Security Note: the user will have the option to require log-in credentials to be entered into the Welcome Screen, to provide security for the case where multiple Tangerine systems will be on the same LAN and reachable by users (in a school setting, for example). It is not intended that the DE be responsive to TCP connections from anything other than the SBC.

[1] Technical note: this is intended to operate just like the UI in the standard implementations of the Red Pitaya Software Defined Radio; refer to online documentation about that system.

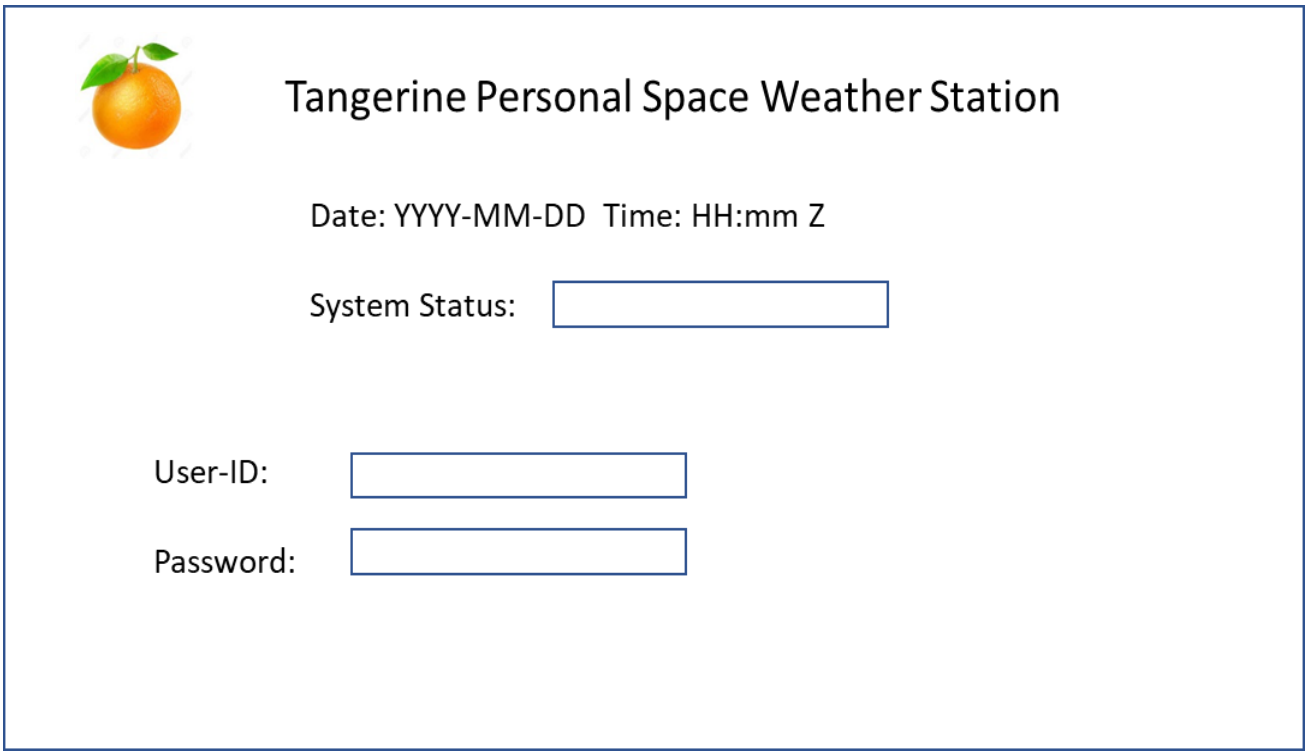


Figure 4. Tangerine Welcome Screen in Security Mode.

Refer to Figure 4. If the Tangerine is configured to require a log-in ("Security Mode"), the Welcome Screen shows the system's Date/Time ("Z", i.e., UTC time zone), the system's status, and User-ID/password blocks. The User-ID and password must be correctly entered to proceed.

For lost password, the password must be reset by using a function on the Odroid, which requires physical access to the Tangerine, and a keyboard/mouse/monitor on the Tangerine SBC hardware.

Refer to Figure 5. If the user has already logged in to the Tangerine or the Tangerine is configured to not require a log-in ("Open Mode"), the Welcome Screen shows the Date/Time and Status information, plus:

- Additional links to other function screens
- A log info sub-panel that scrolls information about the system's operation that would be of interest to the user (content configurable, but defaults to show a history of system activation, deactivation and other related system status detail).

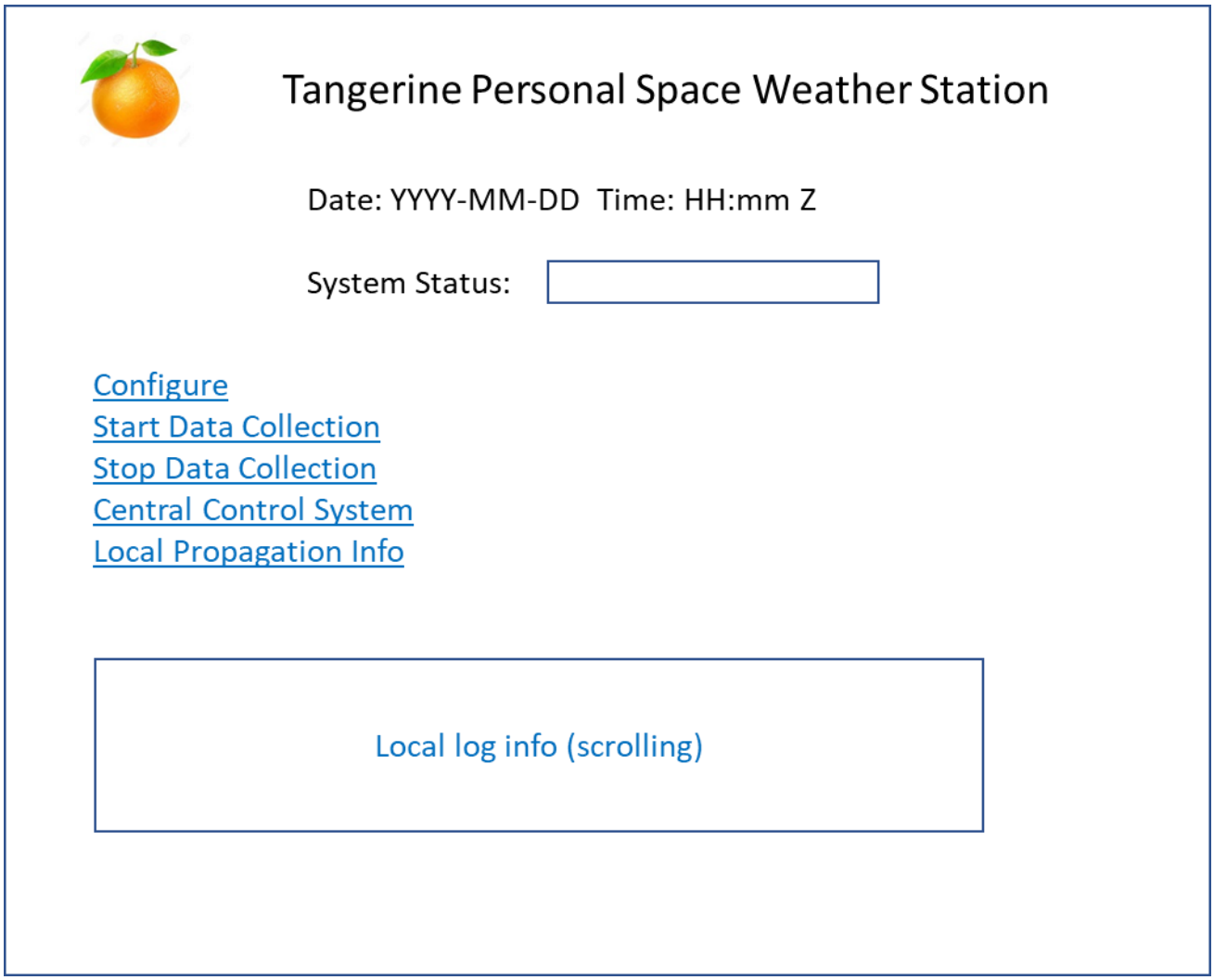


Figure 5. Welcome Screen.

2.2.3. USE CASE 3 – Configuration

This screen is presented after the user has clicked the Configuration link on the Welcome Screen (Use Case 2). Refer to Figure 5.

Link functions available directly on the page are as follows:

- Enter Security Token: This link is paired with a field into which the user copies the security token issued for the user's account in the Central Control System.
- Station Latitude: user enters station latitude in decimal degrees, e.g., 33.61
- Station Longitude: user enters station longitude in decimal degrees, e.g., -87.32
- Station Elevation: user enters station elevation above sea level in meters, e.g., 150

- Throttling of Upload Data: here the user may enter the maximum upload speed allowed for uploaded data (to avoid the Tangerine monopolizing a user's internet bandwidth); in kilobits per second, kbps

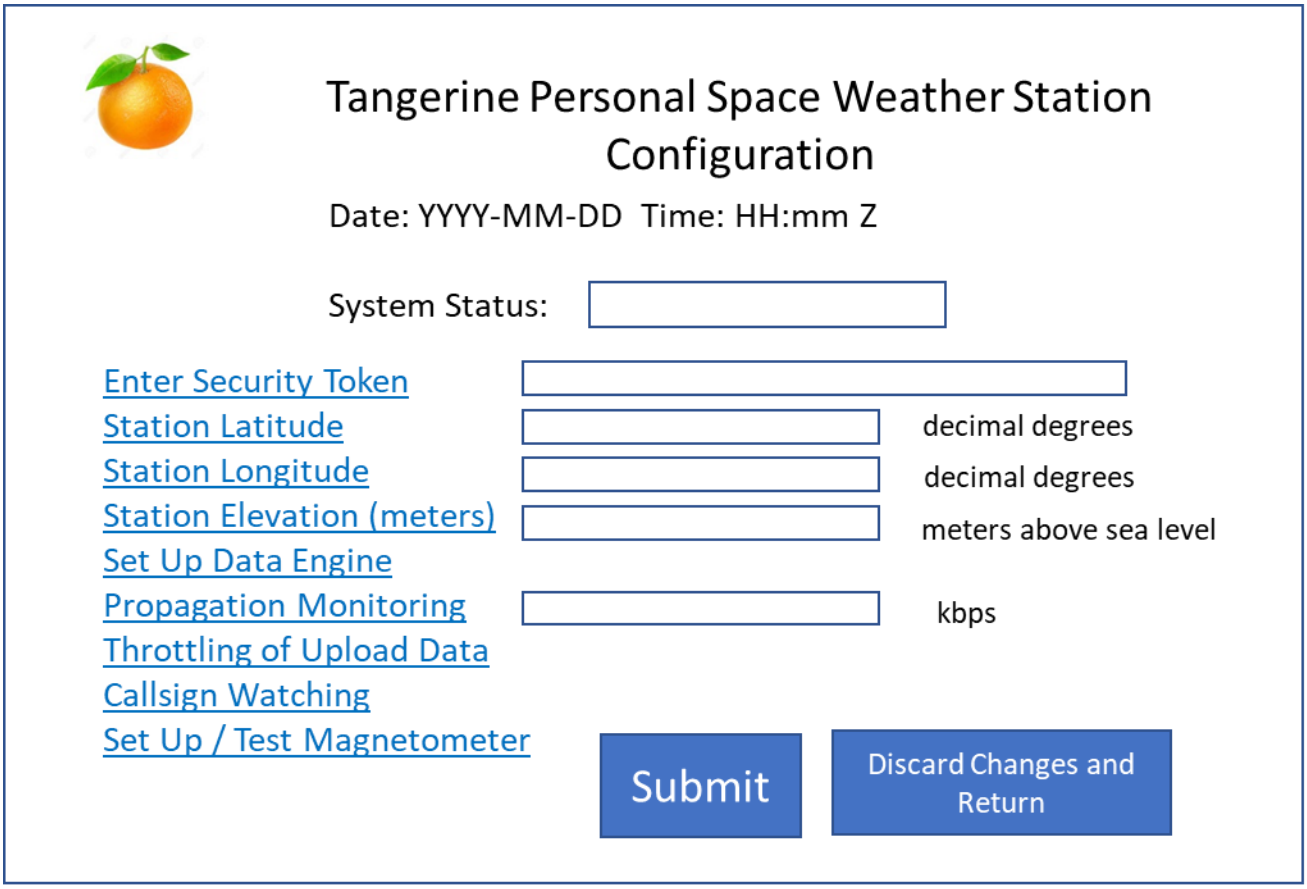


Figure 6. Configuration Main Screen.

Link functions directly available on the screen do not take effect until the user clicks the Submit button; the user may elect to discard all inputs and cancel. This is to preserve a concept of Intentionality, i.e., the user takes an explicit action to make changes effective.

Link functions which bring up a separate screen are as follows (these have their own screen design and Use Case):

- Set Up Data Engine
- Propagation Monitoring
- Callsign Watching

2.2.4. USE CASE 4 – Set Up Data Engine

This use case includes configuration options for the Data Engine, i.e.,

Data Destination (local ring buffer or large server), plus links to separate web pages for configuration of the Clock, Channels/Antennas/ and to test the Data Engine.

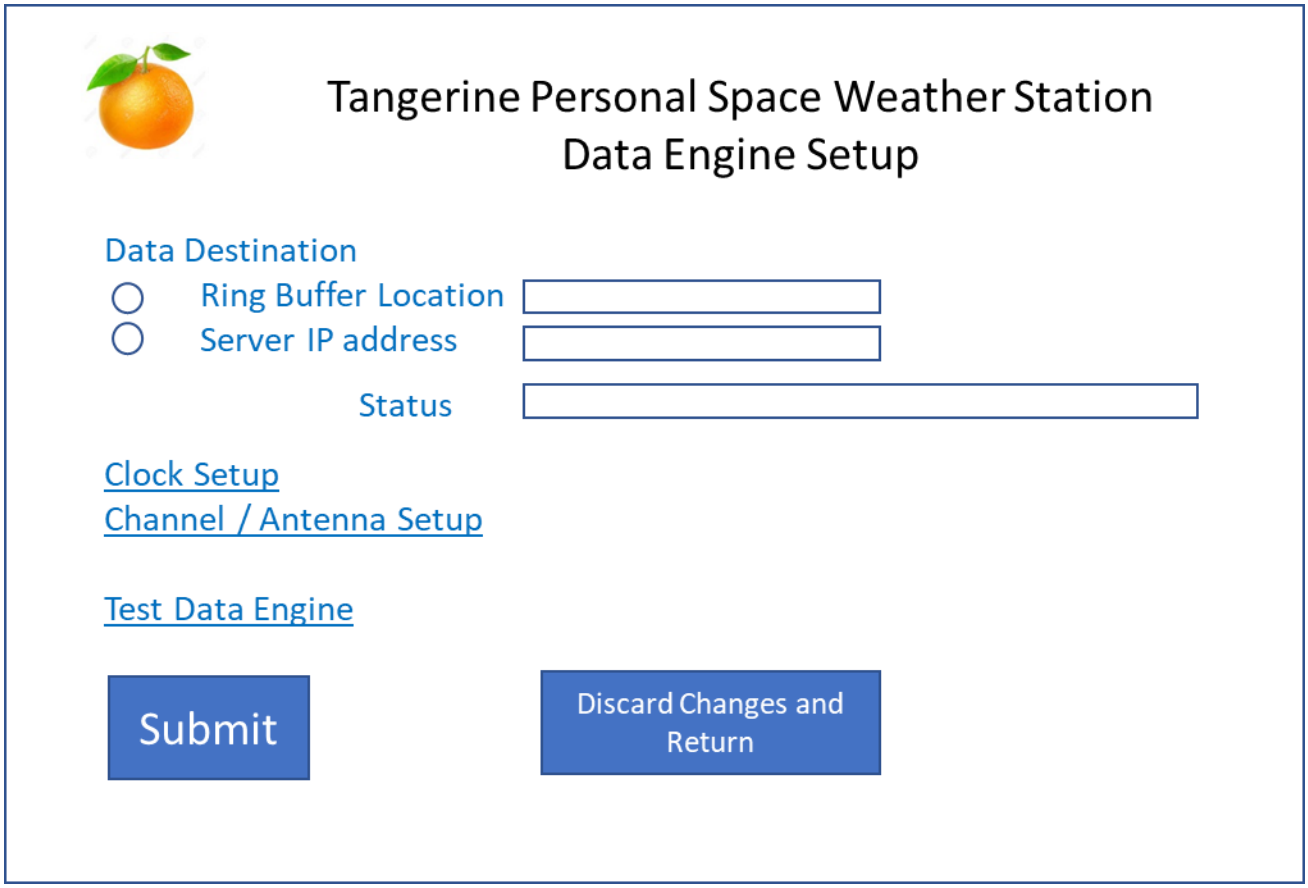


Figure 7. Set up Data Engine, main screen.

Refer to Figure 7. For setting the Data Destination, there are two (mutually exclusive) radio buttons, defaulting to Ring Buffer Location. When Ring Buffer Location is selected, the field to its right allows the user to enter the complete path to the device where the ring buffer is to be located, for example, "/media/odroid/MyDrive1". When the user clicks the submit button, the system will check that this drive exists, is mounted, and that the system has write privileges to the device. A status field will indicate the results of this check.

When Server IP Address is clicked, the user can enter the IPv4 Ip address in the corresponding field; when the user clicks Submit, the system will attempt to connect to this server, reporting the results in the status field.

Clock Setup – action of this link or button is TBD (??)

Channel / Antenna Setup – this link brings up the Channel/Antenna Setup web page; refer to USE CASE 4A

Test Data Engine – this link or button runs a series of checks on settings and the health of the Data Engine, reporting results in the status field. See Technical Details (design TBD (??))

2.2.5. USE CASE 4A – Channel / Antenna Setup

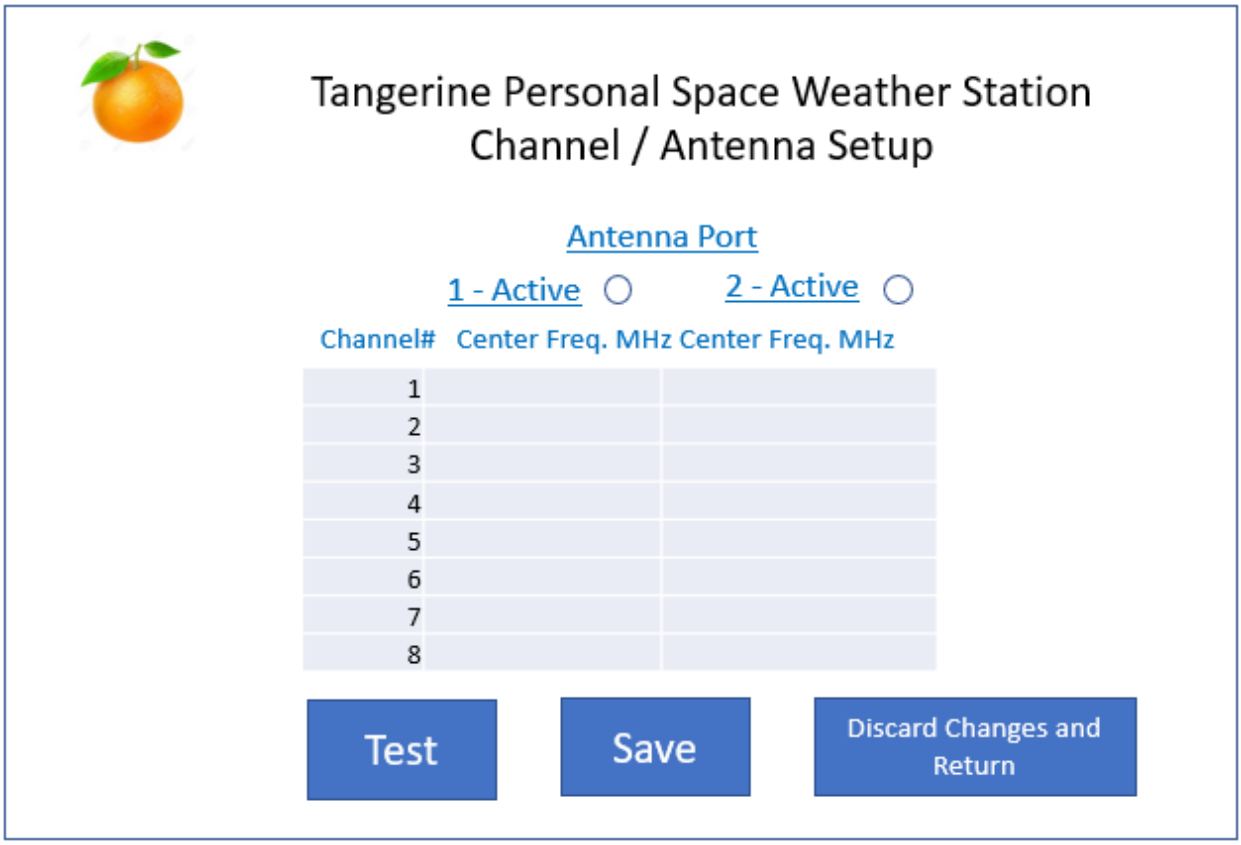


Figure 8. Channel / Antenna Setup.

Refer to Figure 8. Here, the user may click the radio button for Antenna Port 1 and/or Antenna Port 2. (One or both may be selected).

When a port is set to Active, the column under that port becomes available and may be used to set the center frequency (frequencies) of the channels to be monitored. Rules regarding this:

- Valid Frequencies: Frequencies are all in MHz and must be convertible to a floating point number (e.g., 10, 10.1, 0.7, .7, and 50 are all valid), between 0.5 and 60. (TBD ??)
- If a field is NULL, blank, or zero, that port/channel will not be monitored.

- If a field is set to something other than a valid frequency, NULL, zero, or blank, the that port/channel will not be monitored, and the next time the user clicks the Test button, an error message will be displayed indicating the error with that field.

The Test button: clicking this button checks all the fields for Valid Frequencies and reports any errors to the user.

The Save button stores the frequency table in persistent storage in the SBC; the next time the user requests to start monitoring, the frequencies will be passed to the Data Engine.

NOTE: it is possible for a user to select a combination of frequencies that will result in problems such as the following:

- a data rate higher than the SBC can process
- a data rate higher than can be stored on the user's drive
- a data rate higher than can be sent across the network (in the case of Server storage)
- a data rate that will place excessive wear on the user's storage device

Problem management during data collection will need to monitor for these types of problems and notify the user.

2.2.6. USE CASE 5 – Propagation Monitoring

Refer to Figure 9.

NOTE: this feature is secondary to the major objectives of the PSWS project, but is likely to be of interest to a lot of users. A potential problem with this is that its use will consume compute capacity from the SBC, and may therefore interfere with science data collection. It is included here as a potential feature to be implemented in a test mode, and tested for its effects on the rest of the system; it may be deferred to a later phase of the project.

The user has two possible ways of monitoring propagation. (NOTE: these monitoring techniques are constantly evolving, so may be different even by the time this project is implemented.)

NOTE: for either of these propagation monitoring modes to work, the clock of the SBC must be closely synchronized with a standard clock such as the Naval Standard, using SNTP, to have the SBC clock accurate to within ~1 second of actual. This is independent from the high-precision clock used to control the A/D converter in the DE.

FT8 - monitors the FT8 mode. When monitoring, the system collects data for each of the selected frequencies (specifically, in the ~2.5 kHz bandwidth above the configured frequency); at 15 second intervals, the system attempts to decode all signals within this passband, reporting the results to the PSK Reporter web site). The user may observe results by browsing to web site https://www.pskreporter.info/pskmap.html or looking at a local table.

WSPR – monitors WSPR signals. When monitoring, the system collects data for the selected frequencies (within a 200 Hz bandwidth); at 2 minute intervals, the system attempts to decode all signals received, reporting the results to the WSPR web site; refer to https://www.amateur-radio-wiki.net/index.php?title=WSPR

Similar to JT8, the user may observe results by browsing to the wsprnet.org web site or looking at a local table.

For further discussion, refer to the Technical Details section.

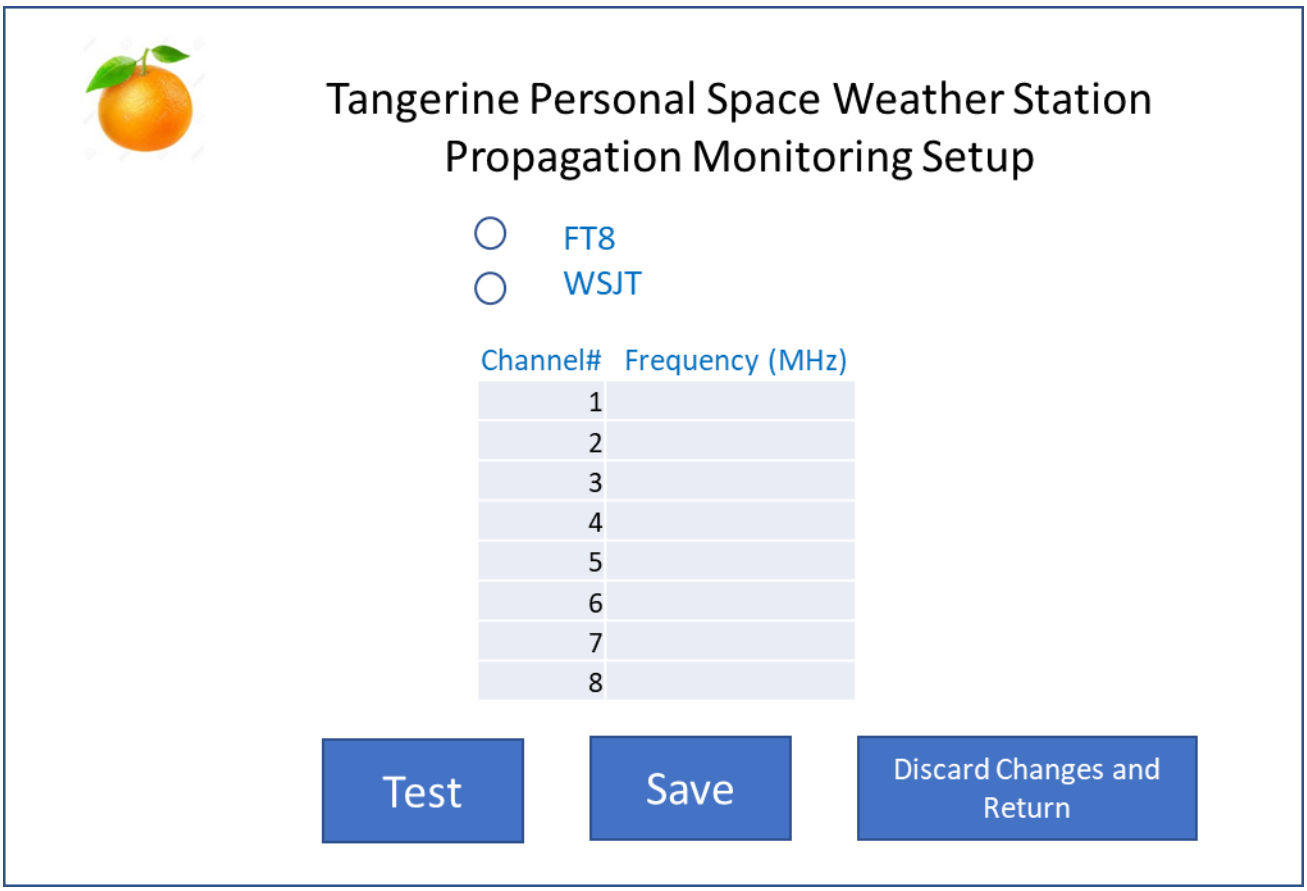


Figure 9. Propagation Setup.

2.2.7. USE CASE 6 – Callsign Watching

Design TBD (??): initial concept: user is able to select one (or two?) modes, such as FT8 and CW, to monitor for any of a set of up to 5 callsigns to appear; upon detection, the system will send an email to a selected email address. The email address selected may also be the SMS adapter of the user's cellular provider, so that the user will get a text message when their selected callsign is heard on the bands.

This is quite feasible to do for FT8, in conjunction with the setup described in USE CASE 5. For CW, some analysis needs to be done to see if a "CW-Skimmer"-like functionality is possible. For both of these, the same caveats apply regarding system capacity; i.e., running these functions may detract from the system's ability to handle

the science-related data collection, which is the first priority. Still, even if this is the case early on, further improvements in processing speeds in available SBCs may overcome the issue.

2.2.8. USE CASE 7 – Set Up / Test Magnetometer

Design TBD (??) – open questions include

Will we support more than one type/brand of magnetometer

What type of settings are required, if any; e.g., is there some adjustment needed based on the length/impedance (etc.) of the wire from the Tangerine to the magnetometer? Are there other calibration constants, and if so, how are they determined?

2.2.9. USE CASE 8 - Start Data Collection / Stop Data Collection

Refer to Figure below.

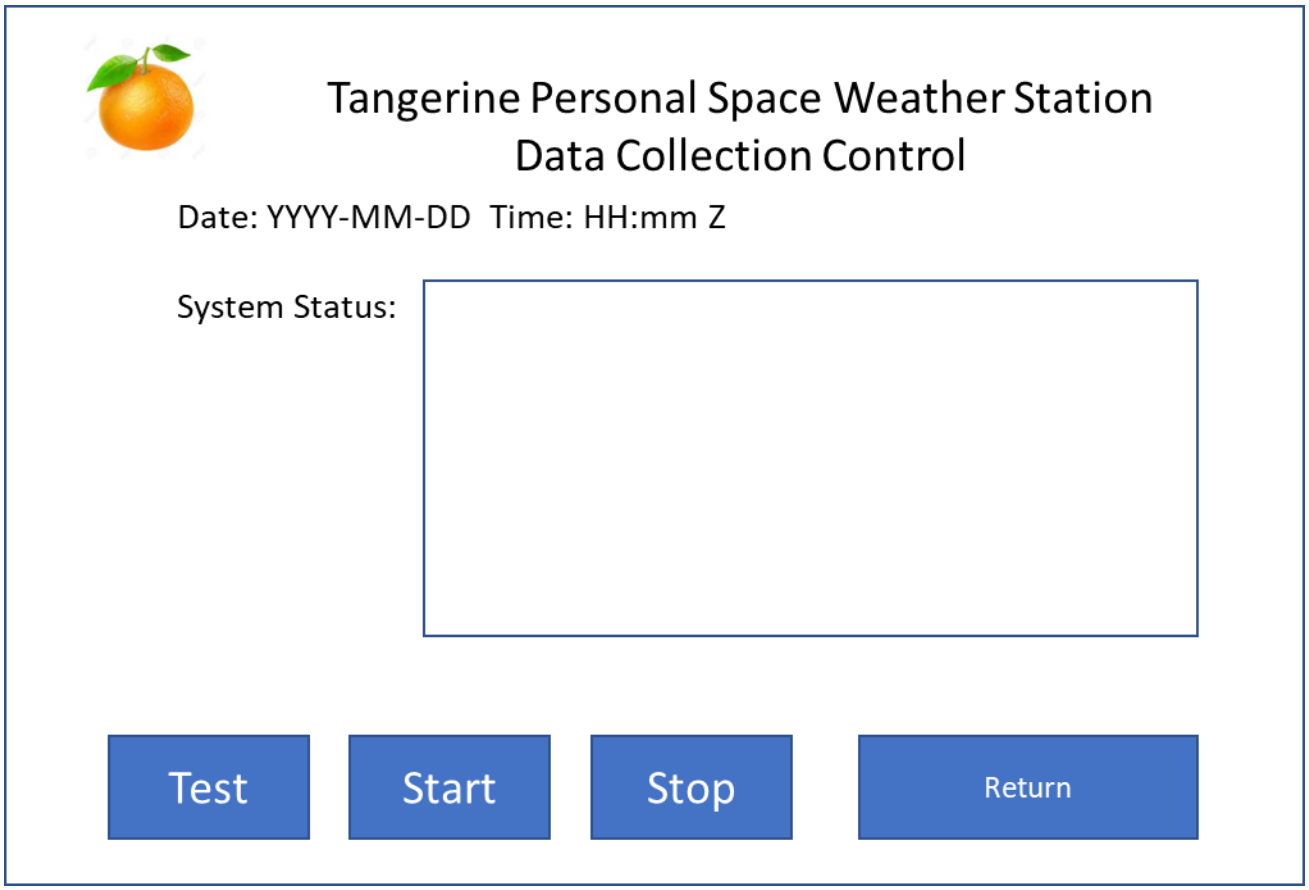


Figure 10. Data Collection Control.

Status Panel – this is a text box which scrolls down, with the most recent entries showing at top of list. Each status line to have a date/time stamp showing when the status line was output, in UTC.

Function buttons:

- Test – communicates with the Data Engine (“health check”) and reports results in the status panel.
- Start – starts data collection; reports successful start or unsuccessful attempt in the status panel
- Stop – stops data collection\
- Return – switches user back to previous web page

2.2.10. USE CASE 9 – Central Control System

This is a simple link which directs the web browser to the Central Control System web site.

2.2.11. USE CASE 10 – Data Uploading Control

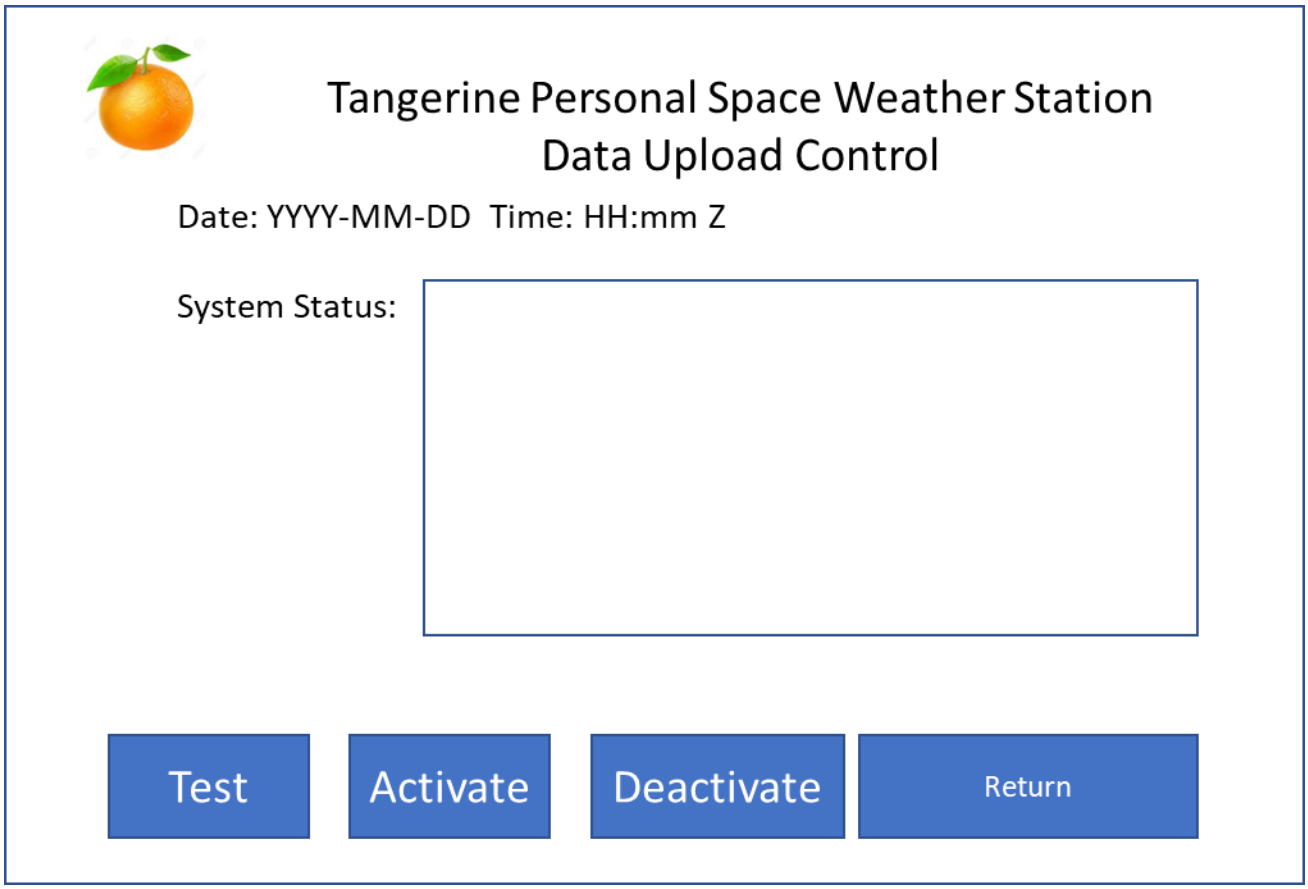


Figure 11. Data Uploading control.

Button functions:

Test – system attempts to initiate communication with the Central Control System, reporting results in the status panel.

Activate – system attempts to go "online," i.e., waits for request-type command from the Central Control system to upload, and will start uploading upon receiving this, according to the parameters in the request received. Online status is confirmed once per minute by sending a "heartbeat" packet to Central Control; when Central Control receives these, it will show this PSWS's status as online.

Deactivate – system goes "offline," – stops sending heartbeat packets, no longer responds to requests for data upload.

Return – return user to previous web page.

3 TECHNICAL DETAILS

3.1 GENERAL

The standard single board computer is to be the Odroid N2 with 4GB RAM. This provides the necessary processing power, has 4 USB-3 ports, and a gigabit ethernet port. The following provides setup tips.

3.1.1 Operating System

Ubuntu, set up for use with Odroid N2, available at:
https://wiki.odroid.com/odroid-n2/os_images/ubuntu

The O/S may be installed either on a Micro SD card (e.g., SanDisk Extreme) or on eMMC chip. You must use the switch on the Odroid board to select which socket (SD or eMMC) to boot from.

There are multiple images; testing so far has validated operation with the full version (not the "minimal"), with version (at least) 4.9, e.g.,
ubuntu-18.04.3-4.9-mate-odroid-n2-20190812.img.xz

Download this and unpack it using 7zip or equivalent.
Burn to chip using Balena Etcher. Use of eMMC is preferred, but there is no need to buy the pre-loaded eMMC from Hardkernel. This can be any eMMC chip, to be loaded with the complete PSWS package.

3.1.2 Packages to install

HDF5: sudo apt-get install `libhdf5-serial-dev`

Digital RF: follow installation instructions at site below. Best results are obtained if you build the package (using make).

https://github.com/MITHaystack/digital_rf

GNURadio – build this following instructions here (note that installation of the many prerequisites is included in the instructions) :

https://wiki.gnuradio.org/index.php/UbuntuInstall

Install GNURadio OOT Module Digital RF. (The system may be connected also to a Red Pitaya SDR, and if so, the OOT Module Red Pitaya (source and sink) must be installed. Testing was done with GNURadio version 3.8 (requires YAML files instead of the XML used with previous versions).

3.1.3 Communication with Data Engine

Both TCP and UDP were considered for communication between DE and SBC; currently UDP is selected, as there should not be any reason for data loss between the SBC and DE, as they are connected through a dedicated GbE port.

A separate Interface Control Document (ICD) governs the logical part of the communication; this includes connection, health check, setting of parameters (ports, channels, channel center frequency, start/stop data acquisition. See Appendix A for a first-draft concept.

3.1.4 Data Storage Format: HDF5 and Digital RF

HDF5 is a standardized format for storing scientific data, used for many applications in storing data from satellites and terrestrial sources. It is lightweight, allows for the association of metadata, and adds very little overhead (beyond storing raw data to a file). There are many software tools available for working with HDF5.

Digital RF is a package which is optimized for saving digital spectrum (radio) observation data. It is able to be aware of the planned data rate, and automatically fill in NaN (“not a number”) values in any cells where data was missed (this can happen when the planned data acquisition rate exceeds the bandwidth of some part of the system at least part of the time). Digital RF uses HDF5 behind the scenes, so that software tools for HDF5 can be used.

For data acquisition, the SBC will use C code to communicate to the DE using techniques included in the ICD. This C code will store received data into the ring buffer.

Note: Digital RF has the following modes for saving data:

- 0 – “no compression” – saves all data in binary format, adding NaN to fill in missing data
- 1 – minimal compression – saves binary data, leaving out missing data
- 2 through 9 – increasing levels of compression

- Add Checksum – self explanatory

Benchmark testing with the Odroid N2 has shown the following

Compression level of 2 or higher poses a higher load than can be handled with the Odroid at the planned data acquisition rates; same with adding a checksum

Using “no compression” and no checksum seems to be the only combination able to keep up with data rates (8 channels on each of 2 antennas, 192,000 samples per second on each). Note that this is a future concept; limits on the number of write operations (for both SSD and spinning disk) are such that running at these speeds would wear out hardware in a matter of months.

3.1.3 Topics for further development include:

- Ring Buffer

  The PSWS Ring Buffer uses the ring buffer design of the Digital RF package which runs in Python.

- Upload / Throttling

  Details on how to limit outgoing bandwidth

  How to cover the case where not all requested data can be uploaded before the data in that portion of the ring will be over-written (mirroring??)

- Interface between SBC and Central Control System

  SBC to Central Control

  - Heartbeat
  - Ready for Upload Command (include this status in heartbeat?)
  - Data Package

  Central Control to SBC

  - Upload Request

  SBC to Server

  - TBD (??)

- **Handling of Metadata** ← this is a big deal, needs to be extensively discussed, maybe benchmarked (will be dependent on ICD DE<-> SBC)

4 SYSTEM ARCHITECTURE

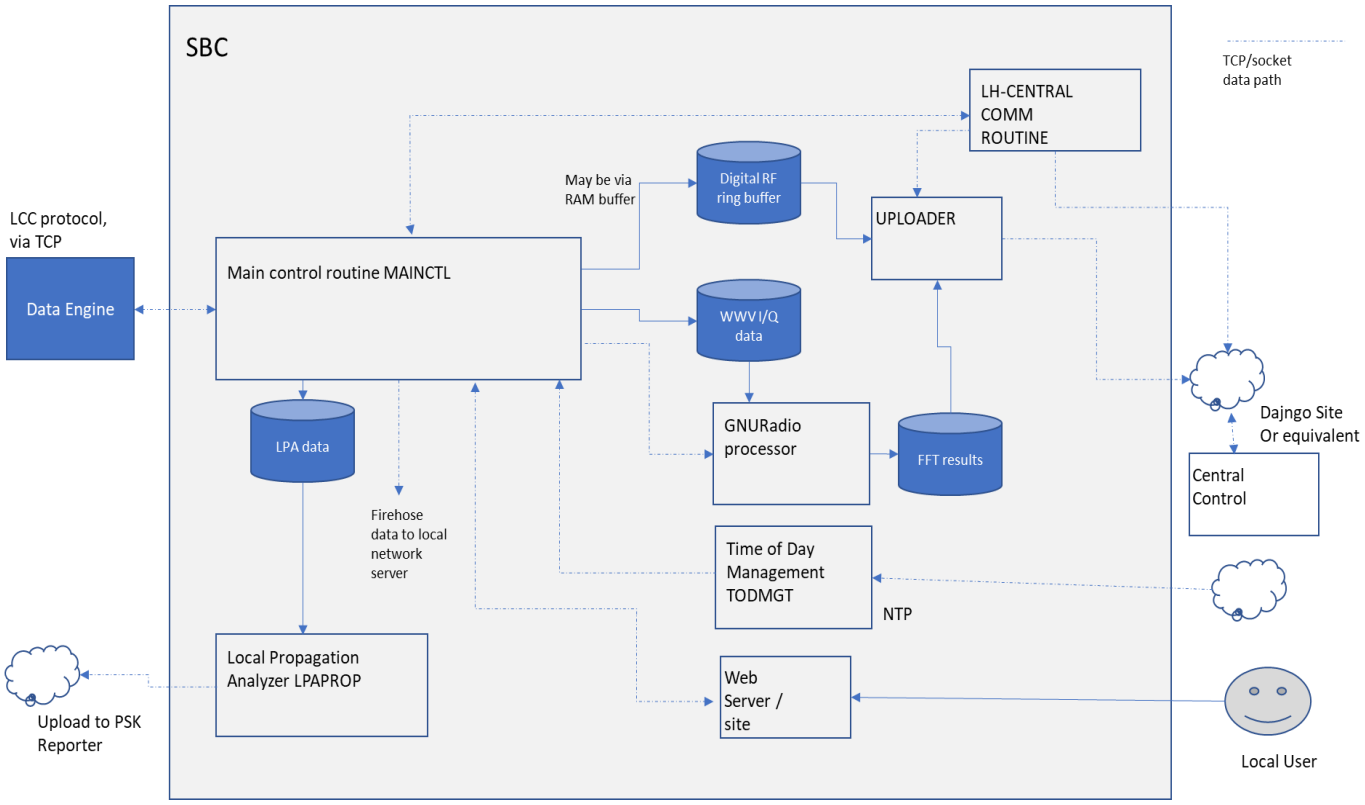


Figure 12. LH system architecture.

Refer to Figure 12. Dotted lines indicate Interprocess Communication via Unix sockets. The LH contains the following processes:

1. MAINCTL. Functioning like the control loop of a process controller, this process accepts commands from the LH Central COMM routine and the webserver site backend, handles data from the DE, sends 4 ksps data to the Local Propagation Analyzer (LPAPROP), sends data to the GNURadio Processor, and interacts with the Time of Day Management (TODMGT) process.
2. LH-CENTRALCOMM interfaces MAINCTL with the Central Control System.
3. LPAPROP. Saves and decodes FT8 signal data, storing results to local database (option), and to PSK Reporter.
4. UPLOADER. When commanded, selects a block of data from the ring buffer and sends it to Central Control, throttling it if so configured.
5. GNURadioPROC. Takes samples of acquired data (incl. signal + magnetometer), performs FFT, and passes to the uploader to be send to Central Control; typically runs once per second.
6. WebServerSite. Provides the user interface and passes commands to other modules when triggered by user interaction.
7. TODMGT. Sets the LH system clock based on a time stamp from DE, or sends timestamp to MAINCTL for forwarding to DE, as configured. Note that time

stamps coming from the DE are based on GPSDO data, and should be handled to avoid continuous updating of the system clock. Consistent with requirements of WSJT-x, the LH system clock should be updated no more frequently than once every 15 minutes, even in the DE is sending more frequent time stamps.

4.1 PROCESSES

4.1.1 Central Request / Response / Upload (standard Data Collection)

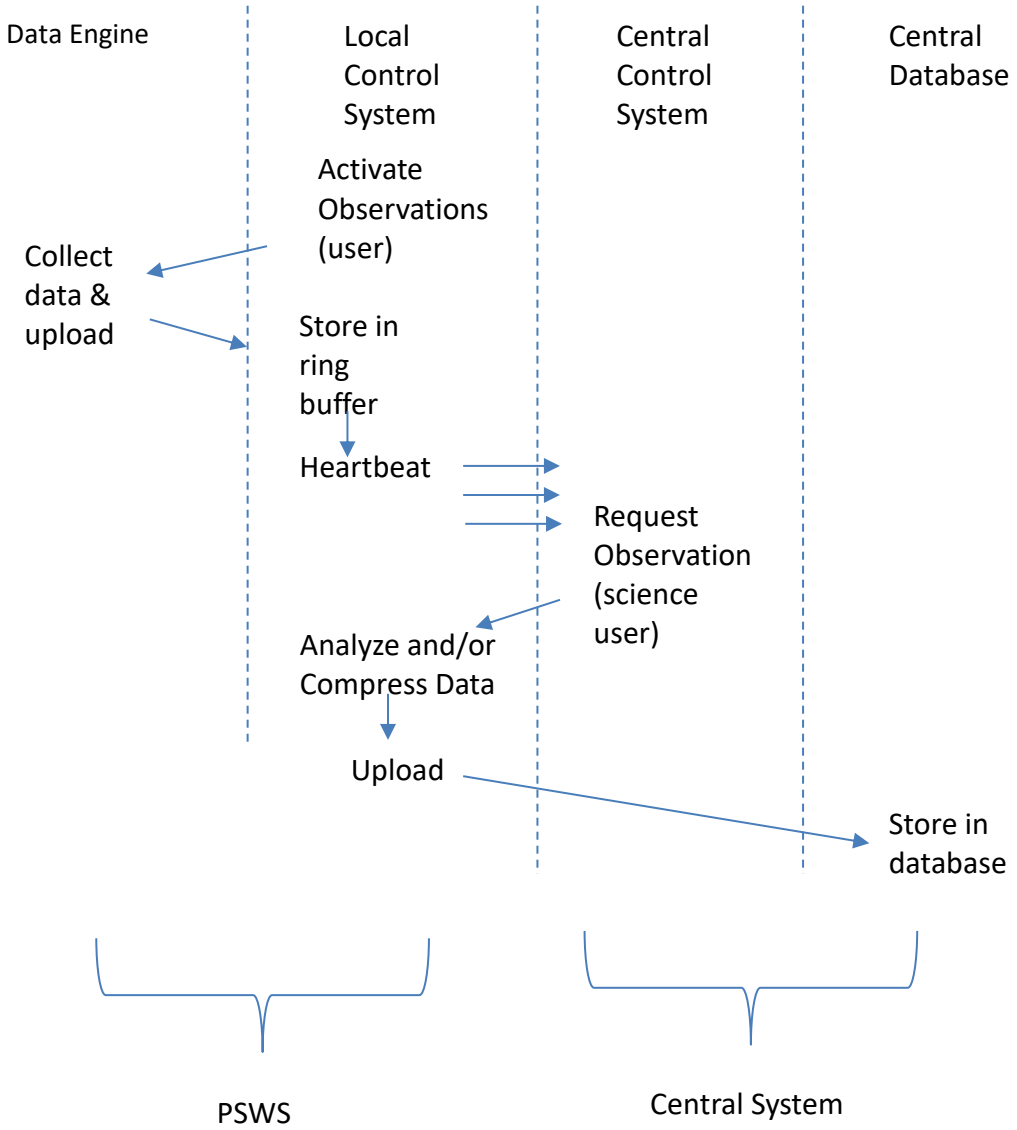


Figure 13. Use Case 1. Request/Response process model.

Refer to Figure 2.

1. Once everything is configured (including user account established, token issued and entered into Local Host configuration), the user Activates Observations (thru the Local Host web interface).
2. The Local Host starts sending heartbeats to Central Control. The user can look at their profile on Central Control and see that their Local Host is connected to Central. The heartbeat includes data to tell the Central System that the Local Host is up, whether the DE is connected, whether data is being uploaded from DE to Local Host.
3. The Local Host triggers the DE to start sending observations.
4. The Local Host receives I&Q data and stores into its ring buffer.
5. A science user triggers a data collection event. Central Control sends a command to active PSWS systems.
6. The Local Host selects the data as requested from the ring buffer, compresses and uploads. Uploads will be throttled according to user preference by including timed pauses between packets.
7. Uploads must be able to start again from where they left off in case of interruption.

4.1.2 Heartbeat

When the Local Host is connected to Central Control, it will send a status message to Central every 120 seconds. The status message will include data on the Local Host's connection to the Tangerine, whether the Tangerine is actively uploading spectrum data, data availability in the ring buffer, etc. (Documented separately – the status of a user's PSWS will be reflected in real time by status information in the user's profile in Central Control).

4.1.3 Large Local Server Data Collection (Use Case 2: Firehose)

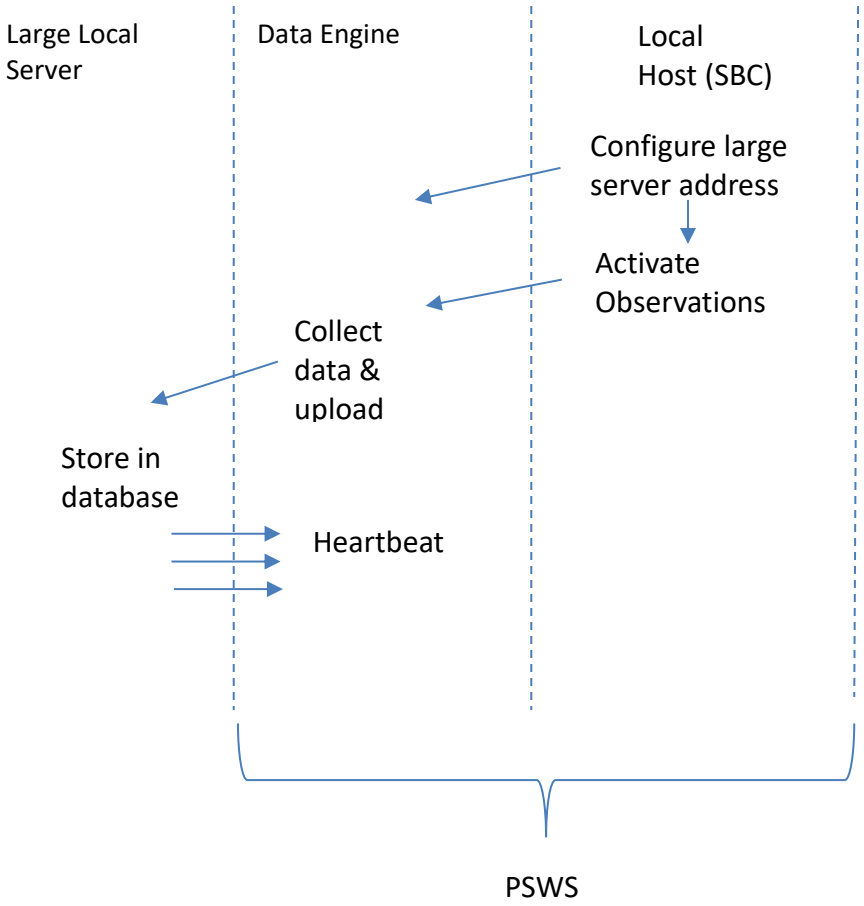
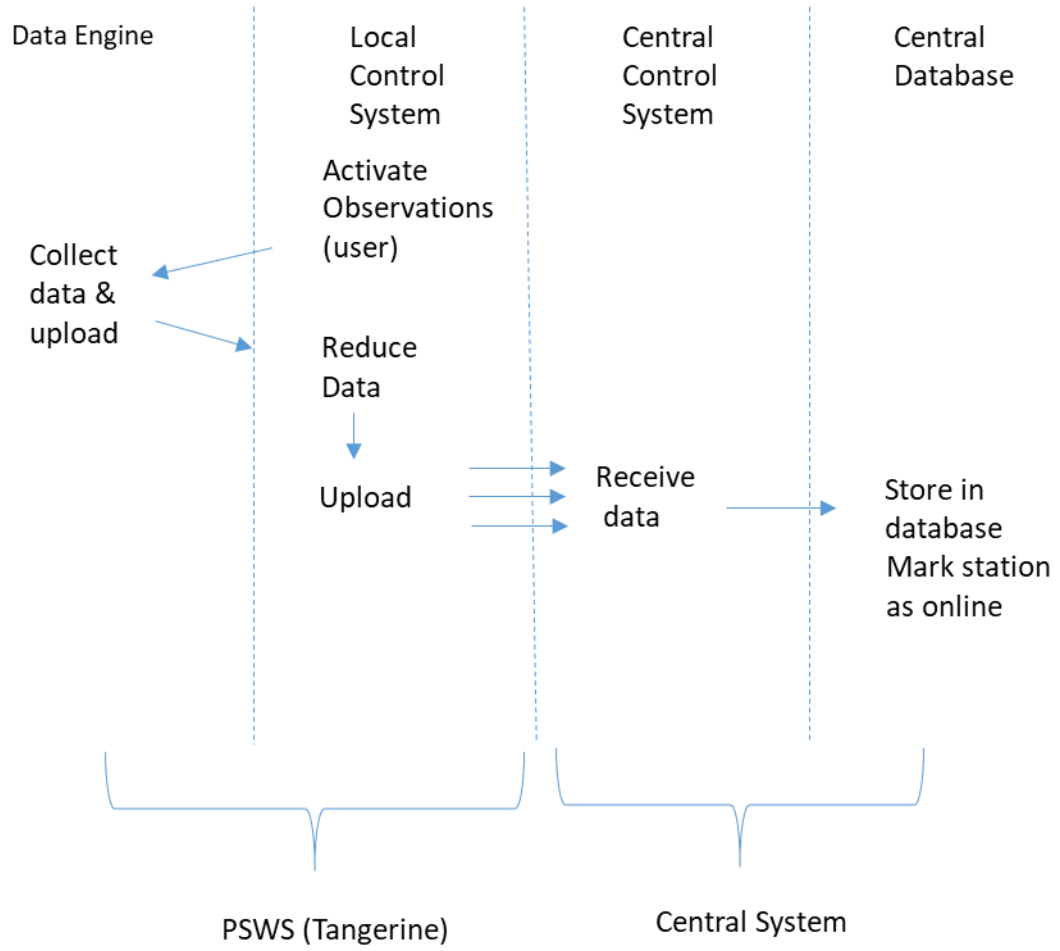


Figure 14. Use case 2, Firehose.

1. Once everything is configured, the user enters the address of the large local server to receive data. (This is a server-to-server case, where the Tangerine acts as a spectrum data server to the large local data collection and analysis server).
2. The SBC shakes hands with the large local server.
3. The SBC sends a command to the DE including the large local server's address and data collection parameters.
4. The DE starts sending data directly to the large local server. Note that this bypasses the SBC and no data is stored to the ring buffer.
5. The large local server heartbeat (every QQQQ msec) maintains the DE in continuous data upload mode.
6. QQQQ NOTE: there may be an additional functionality or use case where Tangerines are discoverable on the network. If so, need to determine the security aspect of that.

4.1.4 Low bandwidth multiband monitoring



1. The user activates Low Bandwidth observations.
2. The DE begins collecting data and passing it to LH.
3. Once per minute, the LH triggers GNURadio to do an FFT analysis of a short package of data, once for each channel being monitored. This is denoted as “Reduce Data” in the process diagram above.
4. Results are uploaded to the Central Host and saved in a database. Note that these data go into a table, whereas in Use Case 1 – ring buffer uploads are saved (in Central Host) as files which are automatically assigned a file name, directory, and cross-referenced in the database.

4.1.5 Local propagation analysis and User Notifications

4.1.5.1 FT8 monitoring and reporting

1. This capability depends on the FPGA being able to send samples from up to 8 channels of 3 khz bandwidth at 4,000 samples/sec (in addition to other data acquisition). (Recommendation – these data should travel from DE to LH via a

different IP port than other data acquisition). Refer to https://github.com/kgoba/ft8_lib .

2. Data will be saved as a binary file from each channel, 60 seconds worth, for one minute. The file includes one 8-byte double value of the frequency, followed by 4,000 I/Q pairs, yielding (for each channel being monitored) 8 + 8 X 4000 X 60 = 1,920,008 bytes.
3. After the files are written, one or more process(es) are triggered to run the FT8 decoder against the saved file(s). The decoder outputs (for each file), a fixed-column output including all the stations heard, frequency, SNR, and grid (when available).
4. Decoding results can be stored to a database to save the results for propagation analysis and/or forwarded to PSK Reporter.
5. The data files are deleted after the data is decoded and saved.
6. OPTION: the user can, via configuration, provide up to 8 callsigns which they want to watch for. When a desired callsign is copied on any monitored band, the user will get an email providing the callsign, datetime copied, and frequency. The user must provide credentials for their own mail server. (Note that most large email servers, i.e., Google Mail, AT&T, Yahoo, etc., no longer accept this type of email. SMTP2GO is suggested. If the user provides an email address that is a text messaging target, the user will get text messages about the detected callsigns.

4.1.5.2 Other Types of Monitoring

OTHER SIGNAL MONITORING. The Section above describes the goals for Phase 1 of the project. Additional types of monitoring (similar to the section above) will be added if sufficient development resources and time are available. Candidates for this include WSPR, CW Skimmer, and algorithms for detecting when a band is open, depending on aggregate signal energy within the passband, and/or similar techniques.

MAGNETOMETER. The standard PSWS magnetometer generally will be connected to the DE in Phase 1. (If time allows, a feature will be added to allow it to be connected to the LH). When connected to the DE, magnetometer readings can be included into acquired data as metadata and stored in the Digital RF files.

4.2 LOGGING

Local Host software will optionally output logging information to a configurable location. Each major step in operation will output to the log.

OPTION: automatic reporting of errors to Central Host will be included if time allows.

5 HELP

For Phase 1, help will be via connection to an online documentation system.

6 REMOTE SYSTEM UPDATES

Planned method of applying updates -- on startup, handshake with Central Control; if an update package is available, download and install. The user will be able to select, via configuration, if updates will be applied automatically, or installed only upon user request. Need to select a package for enabling this.

Re-flashing of FPGA (develop details on this – TBD)

7 TECHNICAL NOTES

Notes from discussions at Dayton, May 2019, as amended. To be incorporated into Detailed Design Specification.

- The plan is to store data into the ring buffer using the Digital _RF HDF5 module from MIT. (Details on how to add metadata need to be developed - TBD)
- The DE will have a certain data packet type to transfer metadata to the SBC.
- Once a metadata variable's content is transferred to the SBC, it will be stored associated with the following signal data, until a new value for that variable is received from the DE.
- The signal data is called the payload, and is planned to be transferred from the DE to the SBC in blocks of 1024 I/Q samples (8,192 bytes). This number is chosen to make it convenient to combine data for FFT, which requires the input data length to be a power of 2. Need to develop details & document on ensuring we know what sample pair corresponds with what channel. The total packet will be longer than 8,192 due to the addition of time stamps and metadata – TBD. Also need to document what other metadata is possible. TBD.
- Time stamps. Each data block will contain the time stamp of the first pair of IQ samples in the payload. (potentially Unix time) (This is metadata and is not included in the 1024 sample payload).
- Data format. The plan is for the payload data to be in 32-bit floating point. IEEE 754-1985.

Appendix A:

This is a notional plan for discussion purposes – to be replaced with final plan once it is completed

LH (SBC) sends:	**DE responds with:**
What is your status?	I am online; my mode is set to xxx
Turn on LED 1 (or set to blinking)	OK
Turn off LED 1	OK
What time is it?	128-bit time stamp; if no GPSDO, return NACK
Here is what time it is: (128 bit time stamp)	OK
Define channel (0 through max. 15), with center frequency f and bandwidth b.	OK
Undefine channel (0 thru 15)	OK
Specify IP address of target server to receive firehose data (UDP)	OK
Start data collection + mode	data packets including time stamps and 8,192 bytes of 192ksps 32-bit floating point complex samples
Stop data collection + mode	OK
Start JT8 data collection	data packets including 4 ksps 32-bit floating point complex samples
Stop JT8 data collection	OK
Start WSPR data collection	TBD
Stop WSPR data collection	OK

Section on LH ←→ Central Control interface to be added here

APPENDIX B: REFERENCES

The following table summarizes the documents referenced in this document.

Document Name	Description	Location
Tangerine SDR Requirements V0.3.pdf	*System requirements*	*https://tangerinesdr.com/TangerineSDR_documents/*

APPENDIX C: Installing GNURadio OOT Modules

OOT creation process by steps:

1. `export LC_ALL=C.UTF-8`
2. `export LANG=C.UTF-8`
3. `gr_modtool newmod` (Name of the new module: e.g. RedPitaya) (creates the module)

1. `cd gr-RedPitaya`
2. NOTE: if you are adding multiple blocks under a single module, it works much cleaner if you add all the blocks at once; this avoids directory confusion & corruption of cmakelists and swig contents
3. `gr_modtool add` <-- this is a stub so that the make process works. Just open the c++ code & fix places where the <+ . . . +> placeholders are

    `source`
    `cpp`
    `red_pitaya_source` <-- example
    copyright holder `(blank)`
    Argument list `(blank)`
    Python QA code `Y`
    C++ QA code `N`

3. `gr_modtool add -l python`

    `source`
    `red_pitaya_source` <-- example
    copyright holder `(blank)`
    Add python QA code `N` (presumably, this is already there from previous step)

4. `mkdir build`
5. `cd build`
6. `cmake ../`
7. `make` <-- this shows all code in c++ files that needs to be fixed to compile
8. Now edit `/lib/red_pitaya_source_impl.cc`
9. correct all c++ code starting with " <+ " e.g., 1, 1, float
10. In `/grc` directory - Copy all the yaml code from file(s) previously converted from xml
11. In `/python` directory - Copy the python code from old version python module
12. `cmake ../`
13. `make`
14. `sudo make install`
15. start: `gnuradio-companion`
16. reload (blue circular arrow icon)
17. Open a grc project and verify that you can see and use the added modules (should show up in right-hand list)