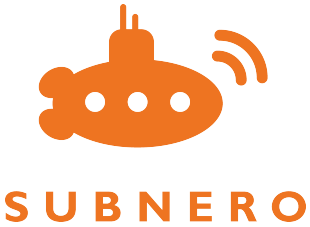

Subnero Modem User Manual

M25MSS3 (Surface Configuration)

Version 1.0 generated on July 08, 2022

Table of Contents

Device Configuration

Serial: M25MSS3

Model: M25MSS3

UnetStack: v3.2.0

Customizations:

Chapter 1. Introduction

The software-defined Subnero underwater modem provides a flexible platform for a variety of underwater networks and applications. With substantial computing power packed into a compact form factor, users can implement and deploy complex algorithms in the modem, hence allowing robust communication between underwater nodes as well as driving the innovation of new protocols and applications. The modem provides options for customization and extension at many levels, allowing network protocols as well as physical layer algorithms to be implemented and tested easily.

Depending on the build quality or application scenario, Subnero modems can be classified by an edition or a configuration. Introduction to Subnero M25M Series Modems is a brief overview video of the various types of Subnero modems.

Subscribe to Subnero's YouTube channel for similar videos.

All Subnero modems run UnetStack - an underwater networking software stack. For more details on UnetStack, refer to the Unet Handbook.

1.1. Editions

1.1.1. Platinum edition

Delivering performance under the toughest environmental conditions, Subnero's platinum edition modems are designed to meet rigorous quality standards mandated by sectors such as defense, oil & gas and sub-sea engineering. All devices are certified for various environmental qualification and EMI/EMC criteria and are subjected to environmental stress screening (PCBA and unit level) before shipment.

1.1.2. Silver edition

Subnero's silver edition underwater modems are the workhorse communication nodes to be used in the general commercial deployments. These modems also provide options for customization and extension at many levels, allowing network protocols as well as physical layer algorithms to be implemented and tested easily.

1.1.3. Research edition

Subnero's research edition underwater modems are designed to bridge the gap between developing applications using a simulator and high-end commercial deployments. These modems are suitable for academic researchers and underwater technology enthusiasts.

1.2. Configurations

1.2.1. Surface configuration (SC)

The SC modem is designed to be powered and deployed from the water surface (E.g. barge, boat, jetty, etc.) and has a pressure-rated housing allowing it to be deployed at depth. It can be connected to the user's computer/machine or a terrestrial network to allow the user to communicate with other modems that are deployed in the same water body (e.g. bottom-mounted nodes, autonomous underwater vehicles (AUVs), etc.)

Figure 1. Surface configuration modem

Figure 2. Surface configuration modem deployment

Applicable Editions: Platinum, Silver and Research

1.2.2. Node configuration (NC)

In this configuration, the modem includes a battery pack, a pressure housing and can be deployed to operate without an external power supply. This configuration is ideal for using the modem as a navigational beacon or connecting sub-sea sensors into an underwater network.

Figure 3. Node configuration modem

Applicable Editions: Silver

1.2.3. Embedded configuration (EC)

This configuration is designed to be embedded/integrated into an external platform, such as an AUV. It depends on the platform for power and pressure housing.

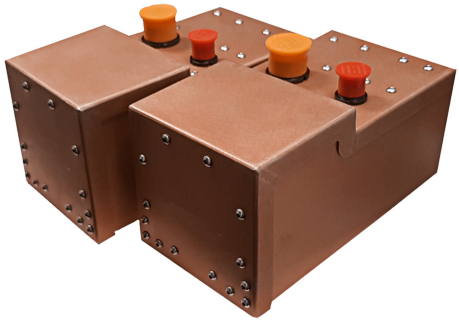

Figure 4. Platinum edition embedded configuration modem

Figure 5. Silver edition embedded configuration modem

Applicable Editions: Platinum and Silver

Chapter 2. Quickstart Guide

This chapter explains how to power up and connect to the modem.

Powering up and connecting to Subnero modems is an overview video on powering up and connecting to the Subnero modems.

2.1. Powering up the modem

1. Remove the dummy bulkhead connector (also known as the dummy connector).

Figure 6. Dummy connector

2. Connect the underwater cable bulkhead connector to the modem's bulkhead connector. Make sure the connector is fully inserted and screw in the locking sleeve completely.

Figure 7. Cable bulkhead connector

Figure 8. Underwater cable connected

3. Connect the power cable to a 24 V power supply. Limit the current to a maximum of 3.5 A.

Figure 9. Power and Ethernet connector

4. Connect the Ethernet connector to the user's computer or to a network that has a DHCP server (e.g. WiFi router).
5. Switch ON the power supply.

2.2. Connecting to the modem

Details of setting up or using an existing IP address is located in Section 3.2.

1. Plugin the modem cable's Ethernet connector to a network that has a DHCP server (e.g. WiFi router).
2. Find the IP address assigned to the modem by the DHCP server (usually available in the router's configuration interface).
3. Connect the user's laptop to the same network.
4. Once the IP address of the modem is known, the modem's web interface can be accessed by entering the modem's IP address in a web browser. Details of the web interface are available in Chapter 4.

5. You will be greeted by the landing page of the modem's web interface.

It is NOT recommended to transmit while the transducer is not submerged in water. However, if the user has to transmit in air, set the transmit power level to -42 dB or lower.

Chapter 3. Basic Operations

This chapter explains the various software and hardware interfaces of the modem, how to configure them to access the modem and the recommended steps in deploying the modem.

3.1. Accessing the modem

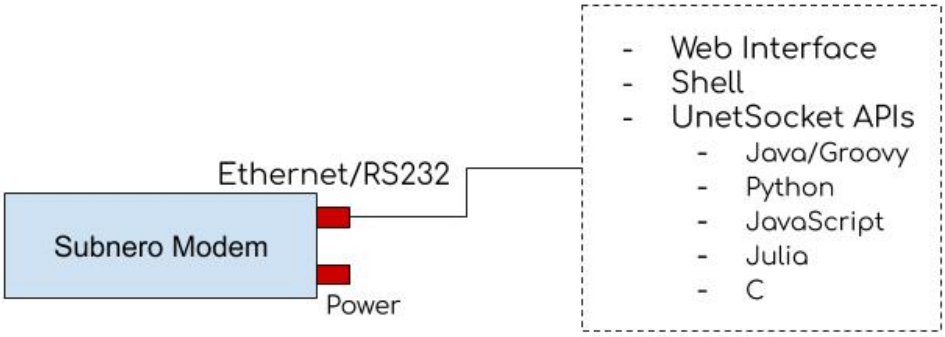

Figure 10. Overview of interfaces to the modem.

The modem along with its exposed interfaces are illustrated in Figure 10. The external interface exposes the Ethernet or the RS232 and power connectors to the modem. To interact with UnetStack running on the modem, there are multiple possible software interfaces.

1. Access using the web interface: Modem's web interface is explained in detail in Chapter 4.
2. Access using UnetSockets (Java/Groovy, Python, JavaScript, Julia, C APIs): UnetSockets are explained in detail in the UnetSocket API chapter in the Unet handbook.

3.2. Setting up IP address

DHCP

1. The modem accepts IP addresses if a DHCP server is available on the network.
2. To use a DHCP to connect to the modem, plug in the modem cable's Ethernet connector to a network that has a DHCP server (e.g. router).
3. The modem will automatically accept the DHCP address assigned by the DHCP server.

Static IP

1. The modem has a pre-assigned static IP address in the `192.168.42.2-254` range.

Check for lable on your modem's hull showing the pre-assigned static IP address of the modem.

1. The following command can be used to set/update the static IP address on the web interface.

```
staticIP - set static IP address

Only applicable for modems with a configurable IP address
```

```
Examples:
  staticIP                        // check current static IP address
  staticIP '192.168.1.214'        // set static IP address
  staticIP none                   // remove static IP address
  staticIP auto                   // automatic static IP in 192.168.42.2/254
```

2. To connect to the modem using the static IP, plug in the modem cable's Ethernet connector to the user's computer.
3. Change the network configuration on the user's computer to assign a static IP of `192.168.42.1` to the network interface on the computer.
4. The modem can now be accessed using the static IP from the user's computer.
5. To remove the pre-assigned static IP of the modem, the following command may be used.

```
> staticIP none
```

6. The following command can be used to revert to the factory assigned static IP.

```
> staticIP auto
```

3.3. Deploying the modem

Below is a recommended setup that the user may follow to mount/deploy the modem.

3.3.1. With Surface Power

1. Follow the steps in Section 2.1 to power up and connect to the modem.
2. The modem is provisioned with 2 x M8 eye-bolts for deployment. Secure the modem to the rope.
3. The modem is negatively buoyant, meaning it will sink in sea water. However, due to tides/currents the modem may not sit vertically in the water column and may rise up to the surface. To avoid this, a weight of at least 5 kg should be attached to the rope to weigh it down.
4. Figure 11 shows a recommended deployment diagram using a weight and a rope.

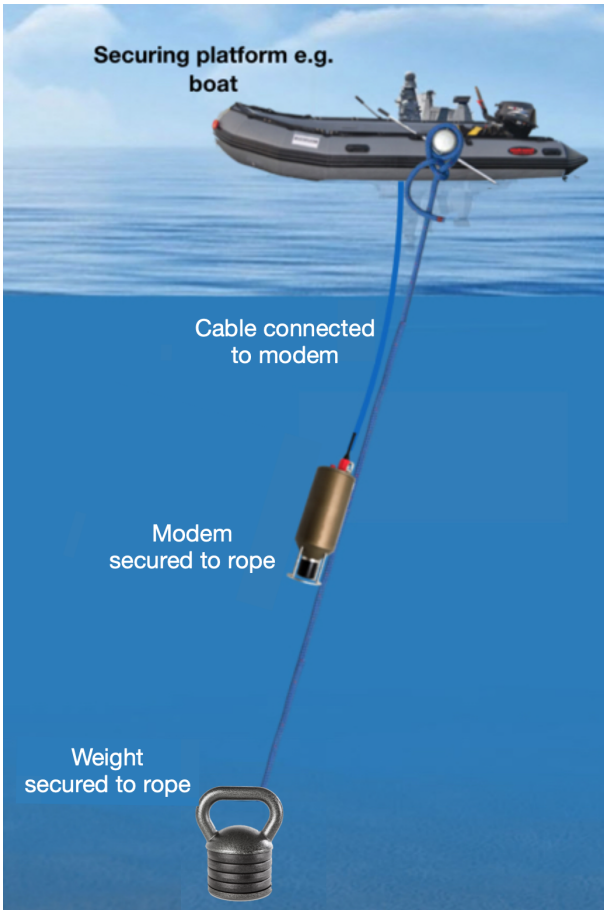

Figure 11. Recommended deployment diagram

3.4. Supported cables

The standard SC modem ships with an underwater cable to be used to power up and connect to the modem. One end of the cable is a bulkhead connector used to connect to the modem and the other end is terminated with Ethernet (RJ45) connector and power terminal (banana plugs).

Figure 12. Underwater cable

Chapter 4. Web Interface

All Subnero underwater acoustic modems ship with a web interface that users can use to operate or configure the modem. Once the modem is powered up and connected to a network or user's computer, open a web browser and type the IP address and hit enter. The user will see the landing page of the modem's web interface as shown below:

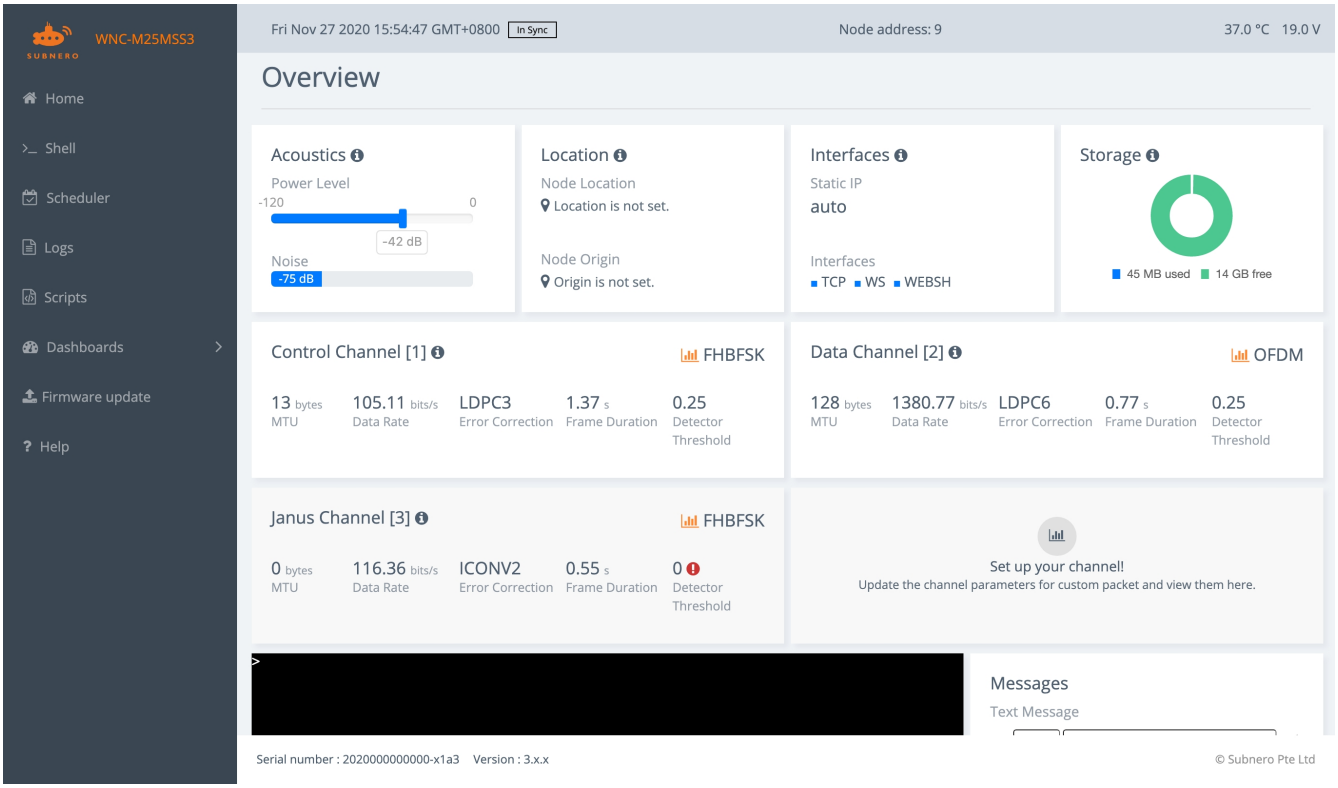

Figure 13. Modem landing page

The landing page brings the user to a dashboard as shown in Figure 13.

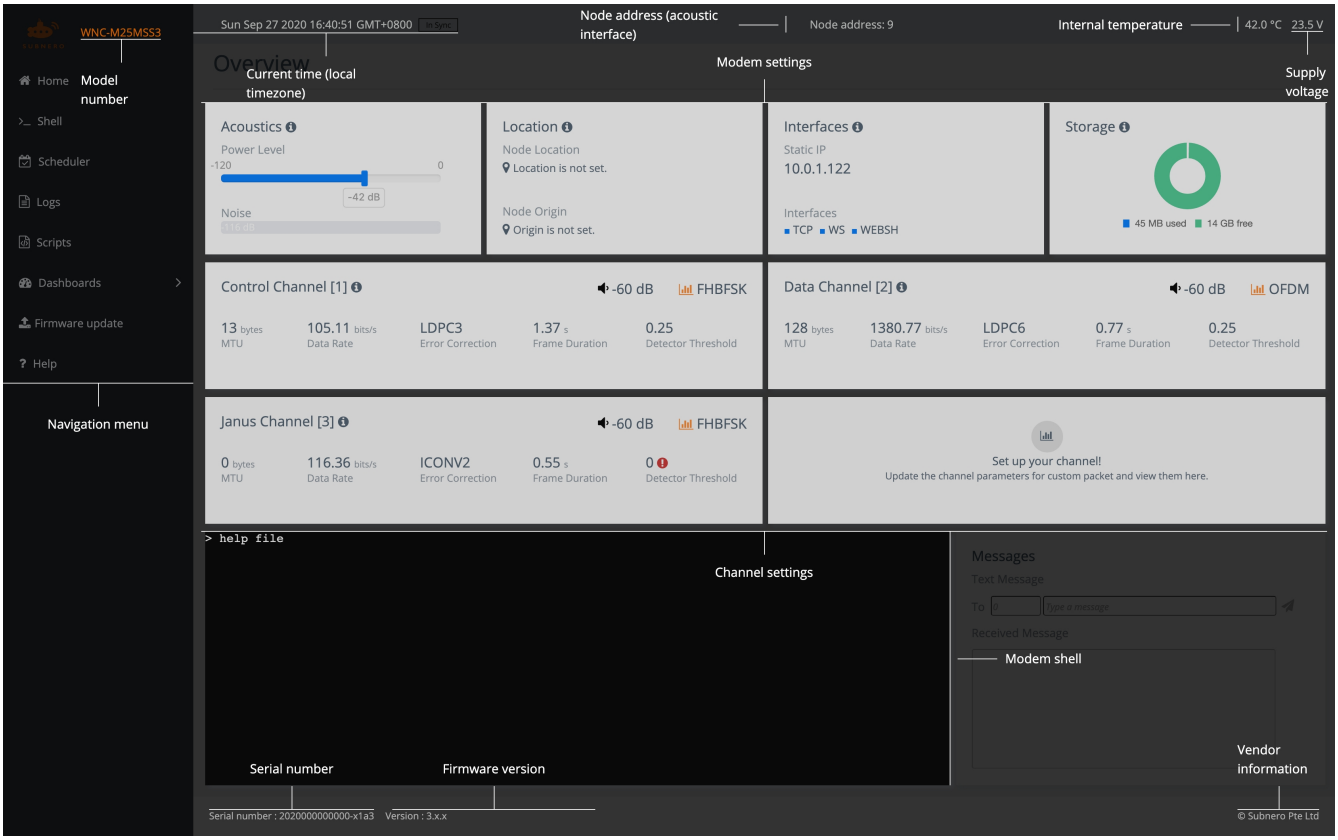

Figure 14. Dashboard components

The various components of this dashboard are listed in Figure 14.

1. Model number: This is the model number of the modem in use.
2. Navigation menu: Various modem functionalities such as shell, scheduler, dashboards, etc. are listed here.
3. Current date and time: The current date and time shown in the local timezone.
4. Serial number: This is the unique serial number of the modem in use.
5. Firmware version: This the version of the firmware running on the modem in use.
6. Vendor information.
7. Modem settings: Few important modem settings are visualized here.
8. Node address: This address is used for the acoustic interface.
9. Internal temperature.
10. Supply voltage.
11. Modem shell: The modem shell presents a space for the user to type in commands to interact with the modem.

4.1. Shell

The shell provides the primary user interface to interact with the modem. Click on the `Shell` in the navigation menu of the dashboard to view a full-screen shell. A user can enter commands to transmit or receive packets, signals or configure the modem using various commands. Most of the modem operations are executed using shell commands.

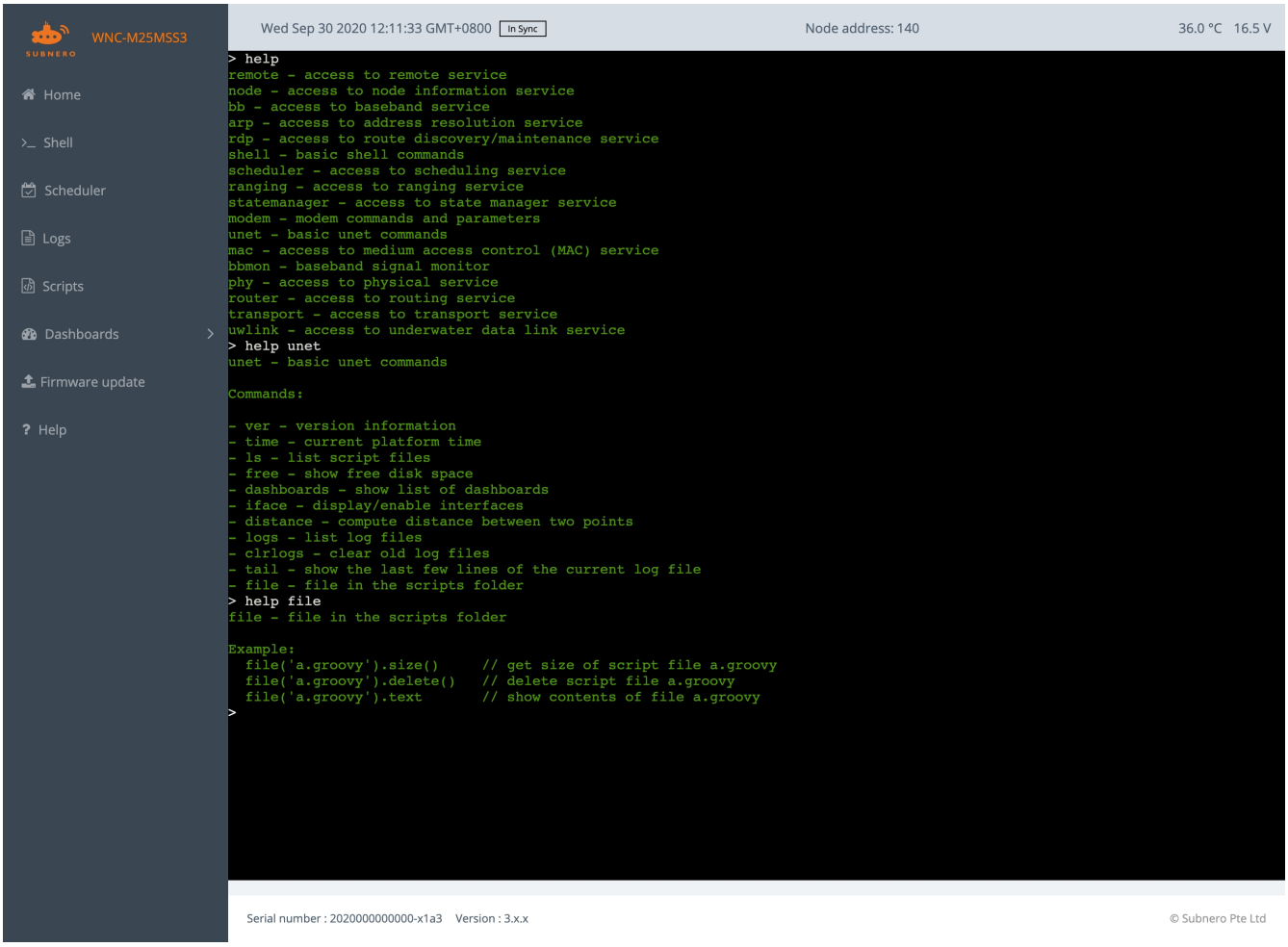

Figure 15. Help command

The modem's shell is provided by the standard Groovy command-line utility. The command `help` and `help unet` lists out most of the major user commands. In order to narrow down the search, the user may use `help` followed by a topic name, for example, to know about `file` command, a user can type `help file`.

```
> help
remote - access to remote service
node - access to node information service
bb - access to baseband service
arp - access to address resolution service
rdp - access to route discovery/maintenance service
shell - basic shell commands
scheduler - access to scheduling service
ranging - access to ranging service
statemanager - access to state manager service
modem - modem commands and parameters
unet - basic unet commands
mac - access to medium access control (MAC) service
bbmon - baseband signal monitor
phy - access to physical service
router - access to routing service
transport - access to transport service
uwlink - access to underwater data link service
```

```
> help unet
unet - basic unet commands

Commands:

- ver - version information
- time - current platform time
- ls - list script files
- free - show free disk space
- dashboards - show list of dashboards
- iface - display/enable interfaces
- distance - compute the distance between two points
- logs - list log files
- clrlogs - clear old log files
- tail - show the last few lines of the current log file
- file - file in the scripts folder

> help file
file - file in the scripts folder

Example:
  file('a.groovy').size()     // get size of script file a.groovy
  file('a.groovy').delete()   // delete script file a.groovy
  file('a.groovy').text       // show contents of file a.groovy
```

The command `ps` lists all the currently running processes.

```
link: org.arl.unet.link.ReliableLink - IDLE
remote: org.arl.unet.remote.RemoteControl - IDLE
state: org.arl.unet.state.StateManager - IDLE
rdp: org.arl.unet.net.RouteDiscoveryProtocol - IDLE
ranging: org.arl.unet.phy.Ranging - IDLE
node: org.arl.unet.nodeinfo.NodeInfo - IDLE
websh: org.arl.fjage.shell.ShellAgent - IDLE
phy: org.arl.yoda.Physical - IDLE
bbmon: org.arl.unet.bb.BasebandSignalMonitor - IDLE
arp: org.arl.unet.addr.AddressResolution - IDLE
transport: org.arl.unet.transport.SWTransport - IDLE
webif: org.arl.yoda.web.WebAgent - IDLE
atm: org.arl.unet.AbnormalTerminationManager - IDLE
router: org.arl.unet.net.Router - IDLE
mac: org.arl.unet.mac.aloha.AlohaACS - IDLE
```

The output of these commands may change depending on the modem configuration and firmware version. Please refer to the `help` command for more details. Most parameters can be read from and written to. However, some commands are read-only. If a user attempts to write to a read-only parameter, it will return an error message. An example is given below.

```
> phy.MTU
13
> phy.MTU=17
org.arl.unet.UnetException: Parameter MTU could not be set [empty response]
```

Any configuration changes are not retained unless the user stores them using the `savestate` command. If the changes are not saved, the settings will revert to factory default after power cycling the modem.

```
> savestate
AGREE
```

The `saved-state.groovy` script file created in the scripts directory. It contains the changes that are saved using the `savestate` command. A user can choose to modify the file directly using the script editor. More information about this feature can be found in UnetHandbook's State persistence chapter.

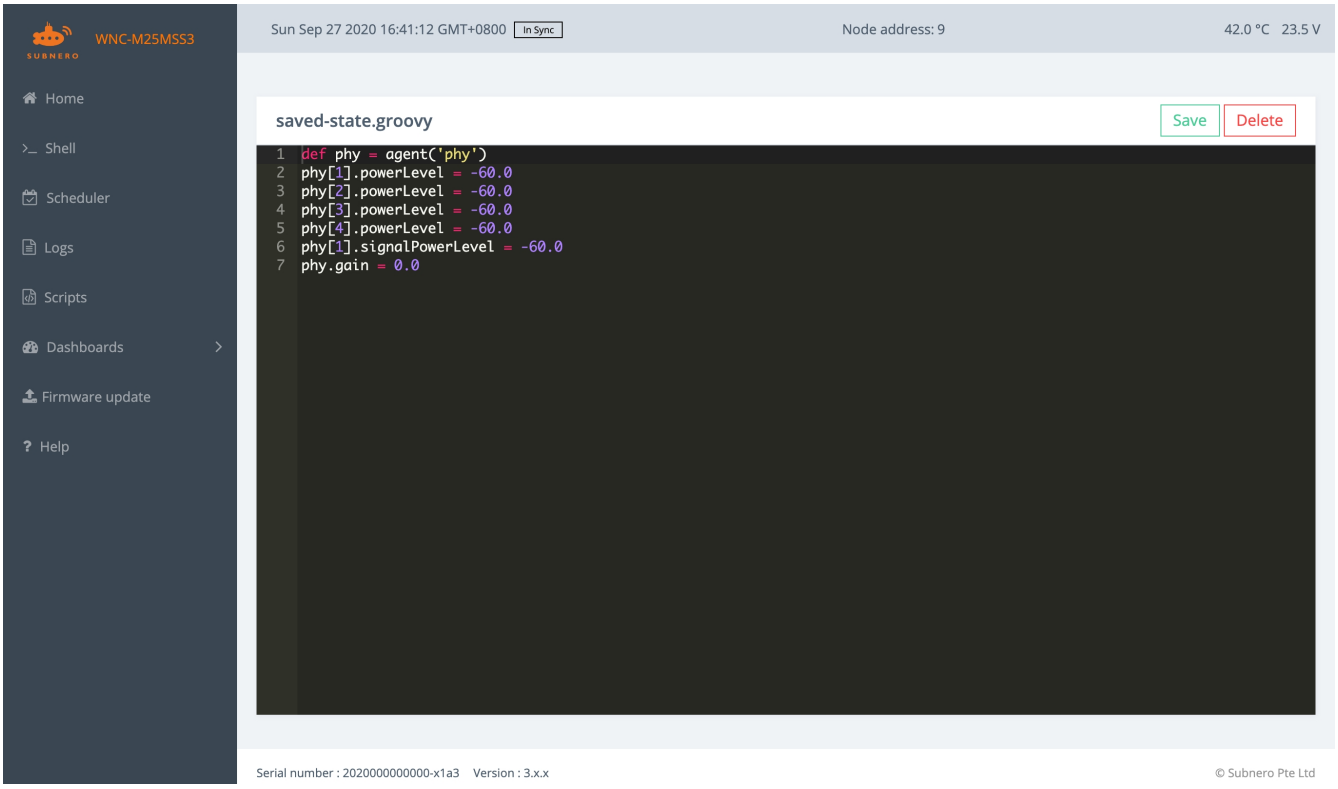

Figure 16. `saved-state.groovy` file

For details on various shell commands, refer to Unet Handbook Command Reference.

4.2. Scheduler

The scheduler allows the user to configure sleep and wakeup schedules so that the modems can enter power save (sleep) mode. A user can schedule a specific time slot for a bottom-mounted node configuration modem to be powered up and ready to transmit and receive data/signals.

4.2.1. Adding a schedule

The steps to add a schedule is explained below:

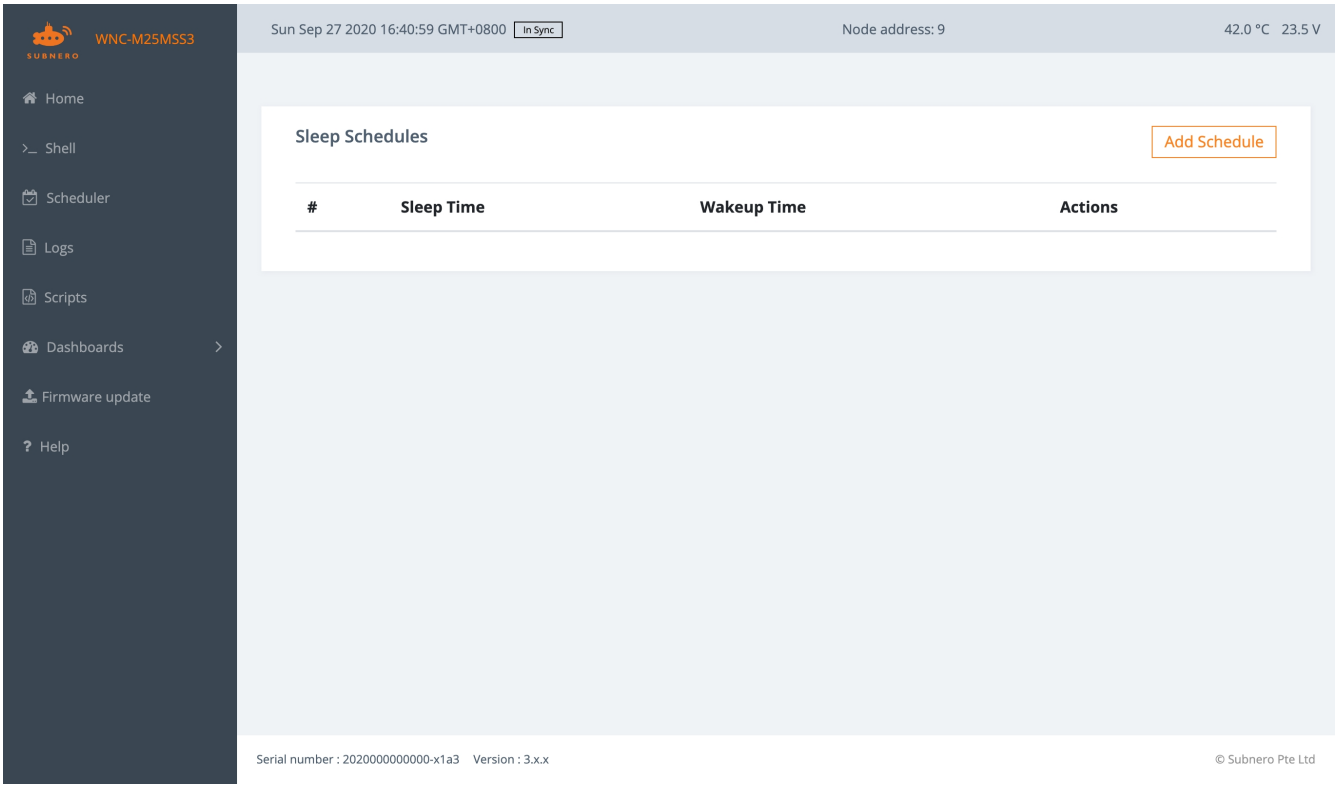

Figure 17. Scheduler

1. Click the "Add schedule" button.

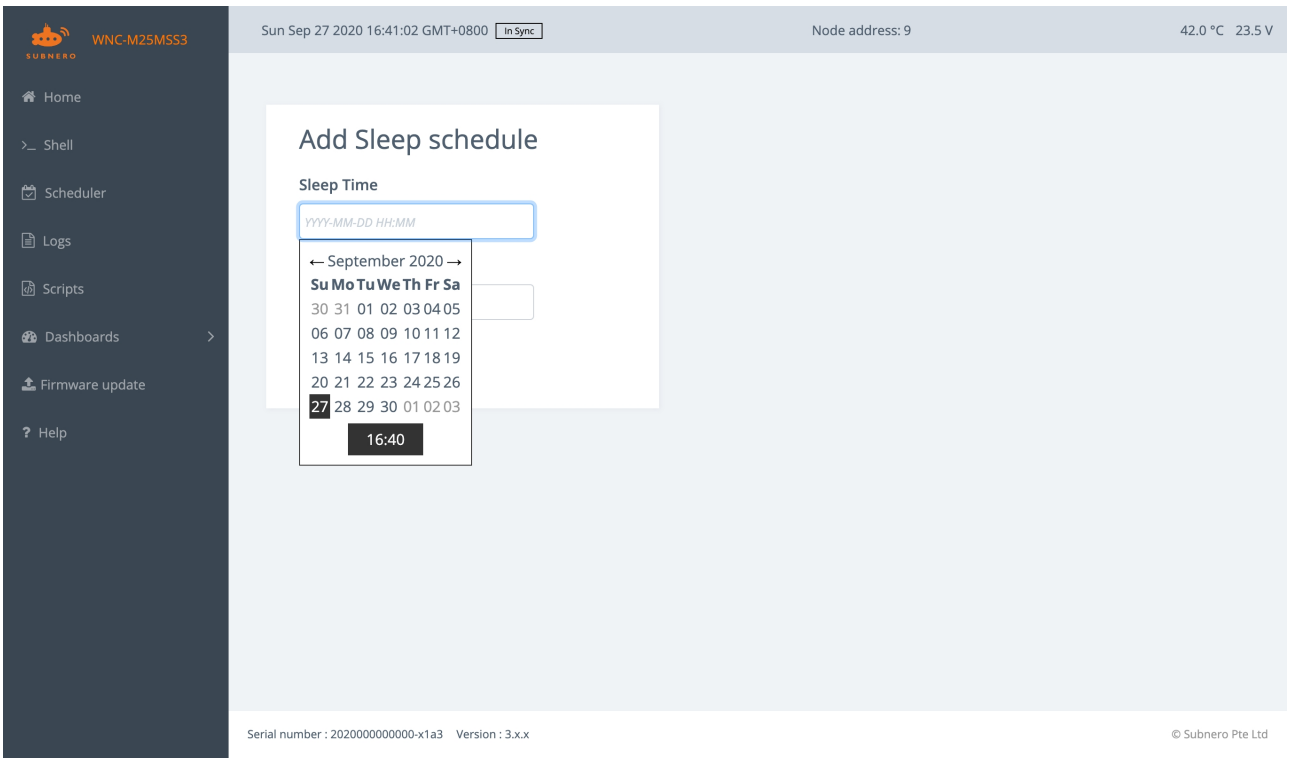

Figure 18. Adding a schedule

2. Add the sleep and wake up time and click "Add Schedule" button.

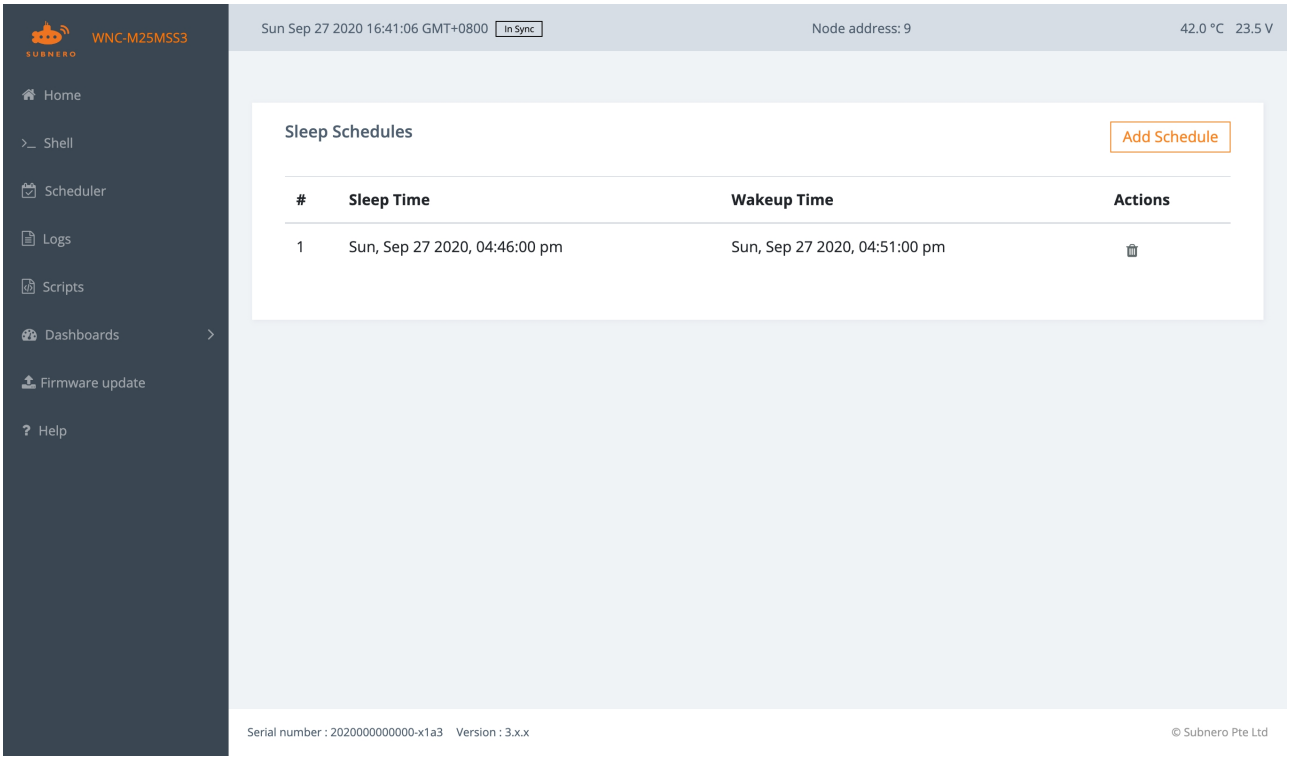

Figure 19. Schedule list

3. New schedule is listed in the Scheduler page.

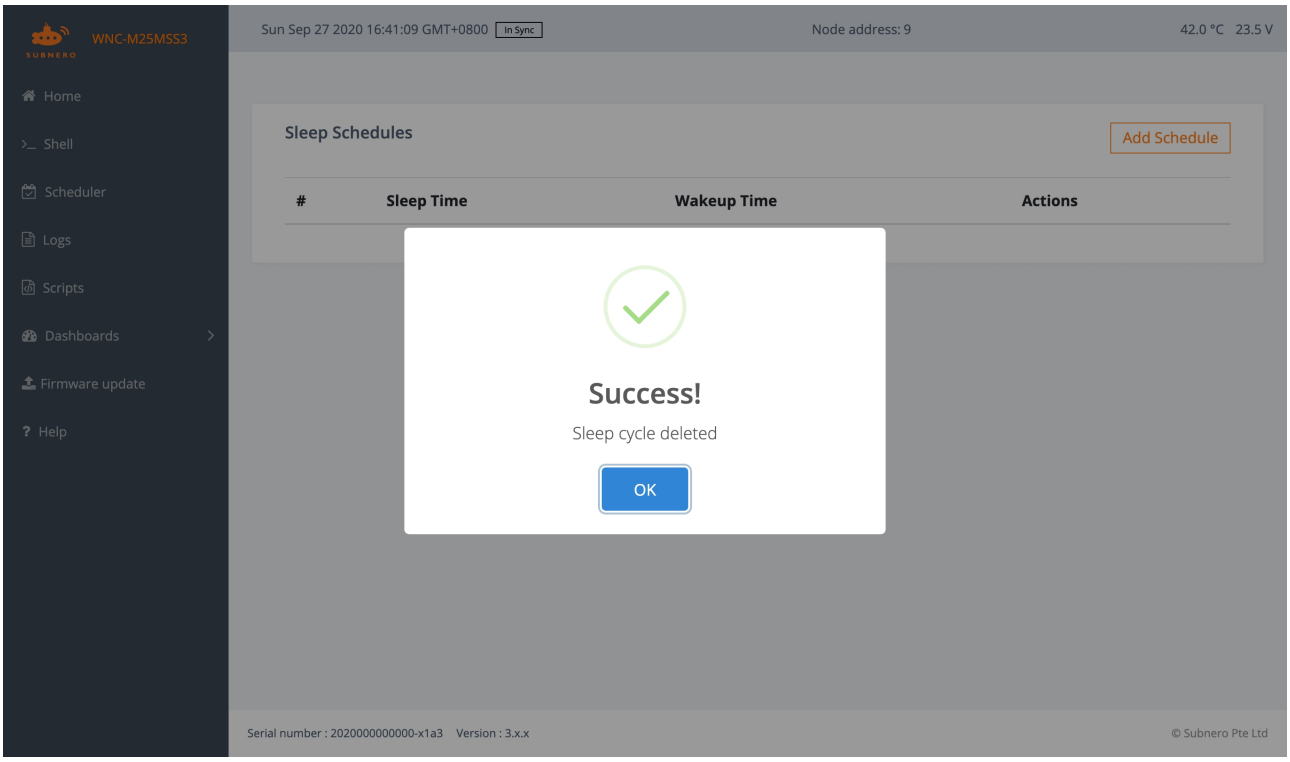

Figure 20. New schedule

4. Click "Remove" button to delete an existing schedule.

In case if the user would like to add/modify/delete a schedule or a configuration parameter after the modem is powered down, simply power ON the modem. After the modem boots up, it waits for 5 minutes for user input. If no input is received and if there is a sleep schedule that is configured, the modem will enter a sleep

state.

Make sure the schedules added are well planned. Once the modem is deployed in water, it is not possible to communicate with the modem while it is in a sleep state unless acoustic wakeup is supported for the modem in use.

4.3. Logs

The logs page displays the current and the past logs of the system.

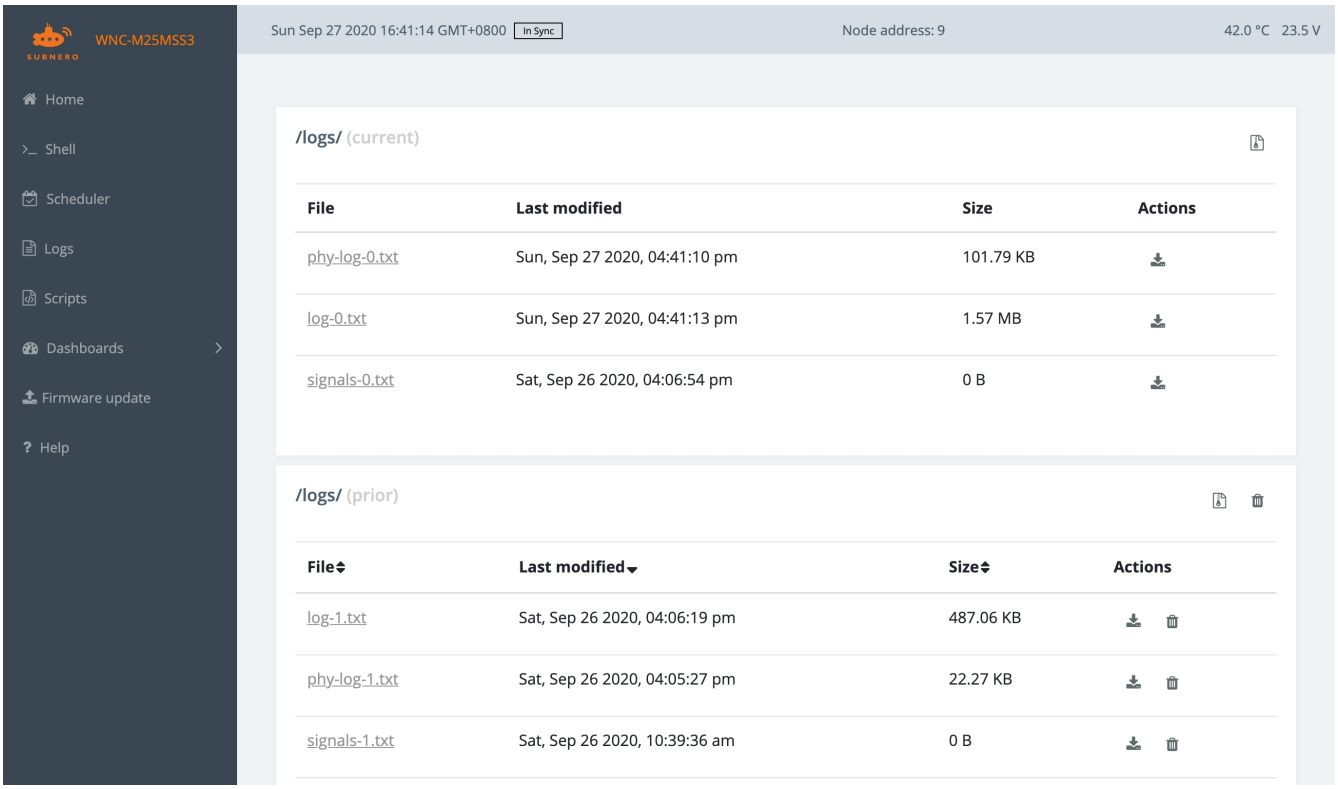

Figure 21. Logs

There are two kinds of log files:

- log-0.txt: This file contains network stack (UnetStack) logs.
- phy-log-0.txt: This file contains firmware logs.

A user can view the logs by clicking the files or download the logs for further analysis. Every time the modem is power cycled, a new log file is generated. The log files marked with "-0.txt" is the current set of log files. The modem implements log rotation. Once the limit is reached, the modem automatically deletes the oldest file. A user can delete the log files using the "Delete" button.

The maximum number of log files is 20. Older log files will be automatically replaced by newer log ones after 20 files. This number may change in the future.

4.4. Scripts

The scripts page can be entered by clicking on the `Scripts` in the navigation menu. The scripts page

lists any existing scripts and also provides the user capability to create/edit/delete scripts. The script editor allows the user to create/edit/delete scripts, classes (e.g. UnetStack agents) and save directly in the modem using the web interface. It also allows uploading scripts or classes or jar files. Various scripts, classes and jar files in the respective folders are listed in the web page.

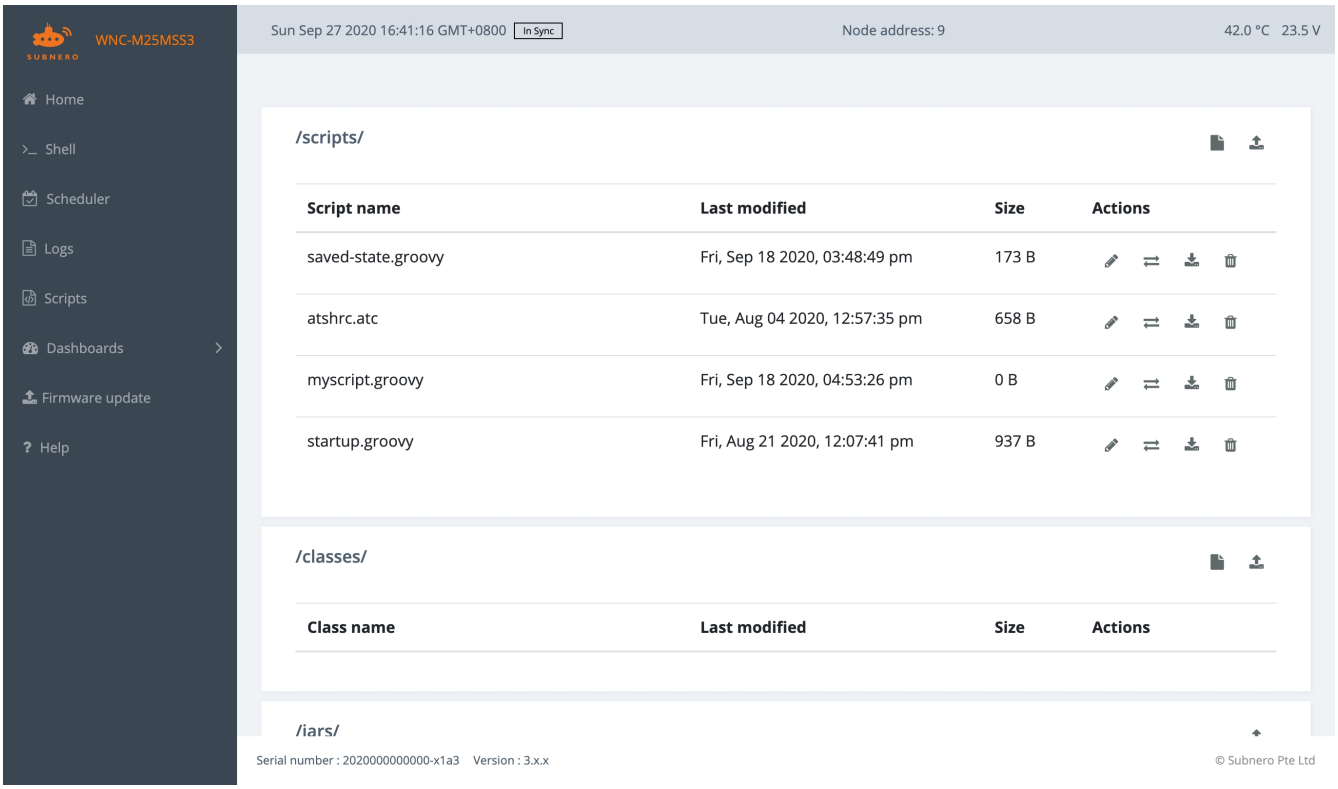

Figure 22. Script editor

The steps to create/edit a script or class is listed below.

1. Click the "Create Script" or "Create Class" button.

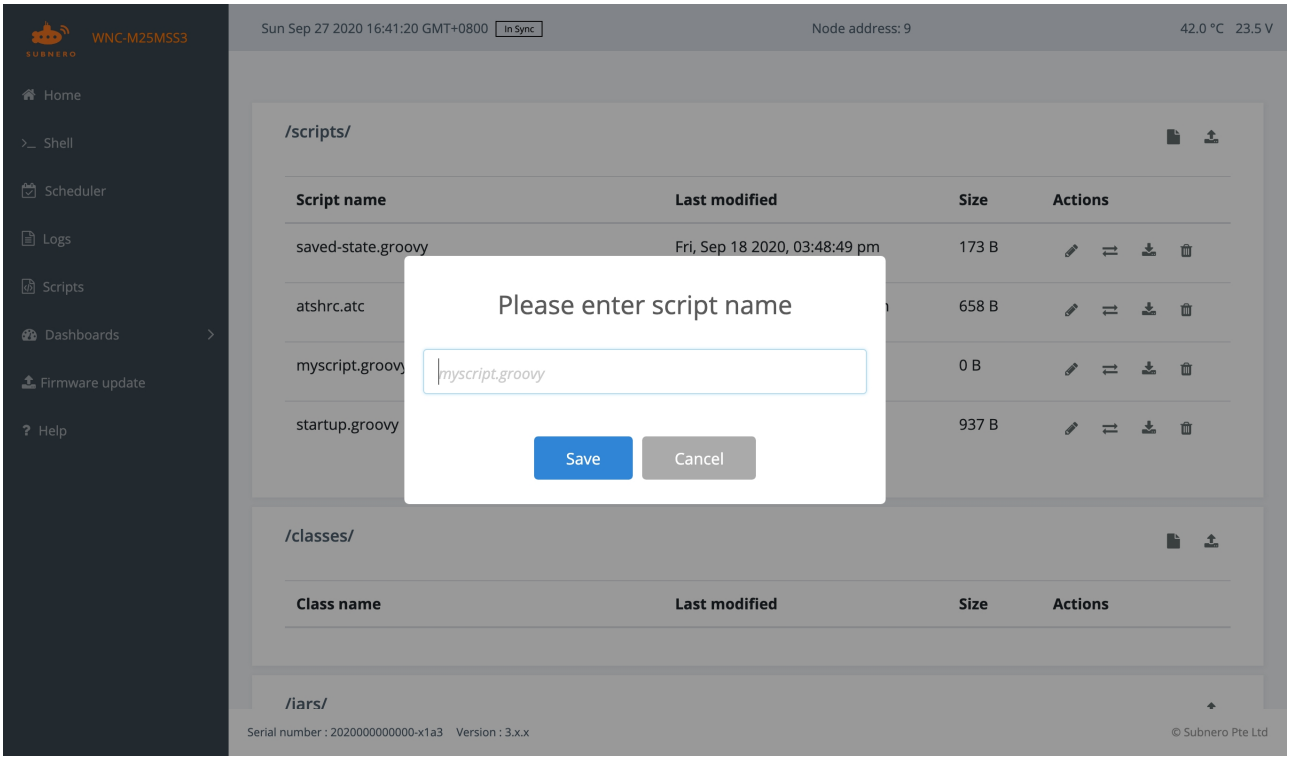

Figure 23. Creating a script

2. Give a name to the script or class file.

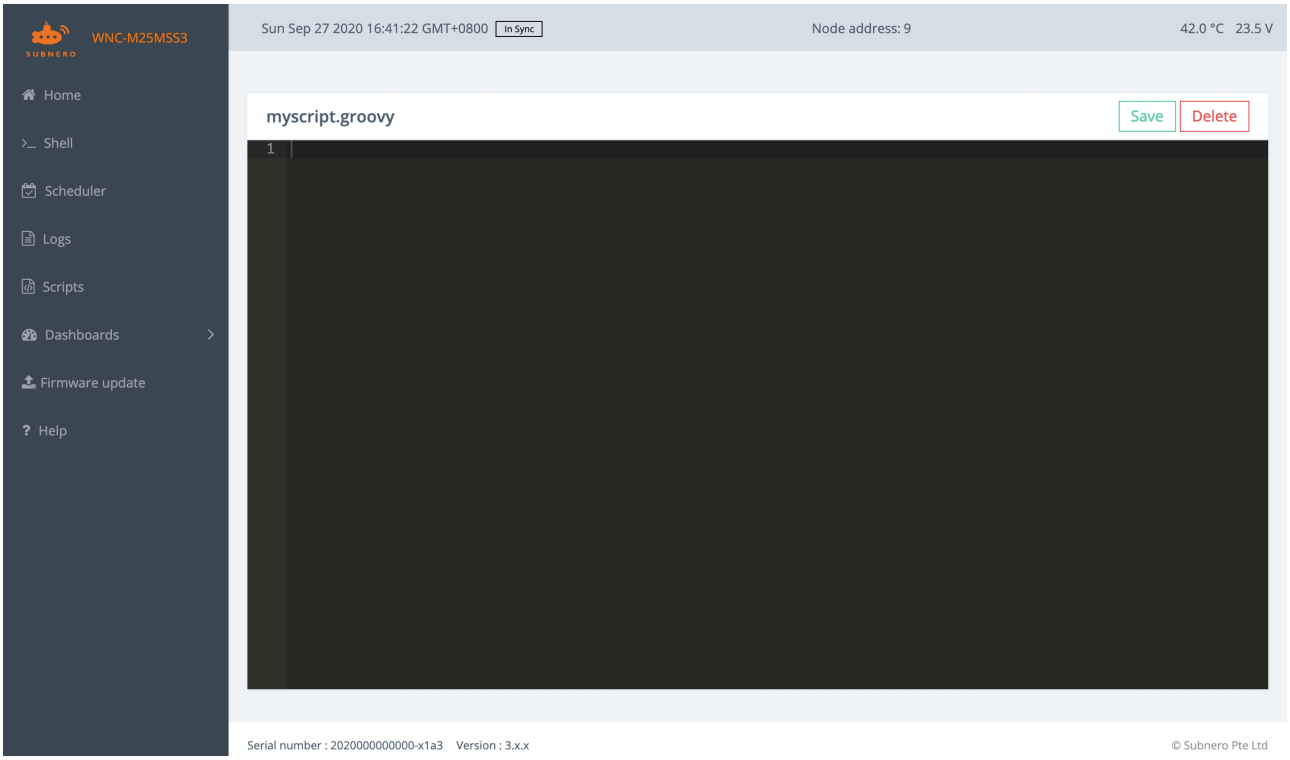

Figure 24. Naming the script

3. This will open an editor window where the user can write their own scripts or classes.

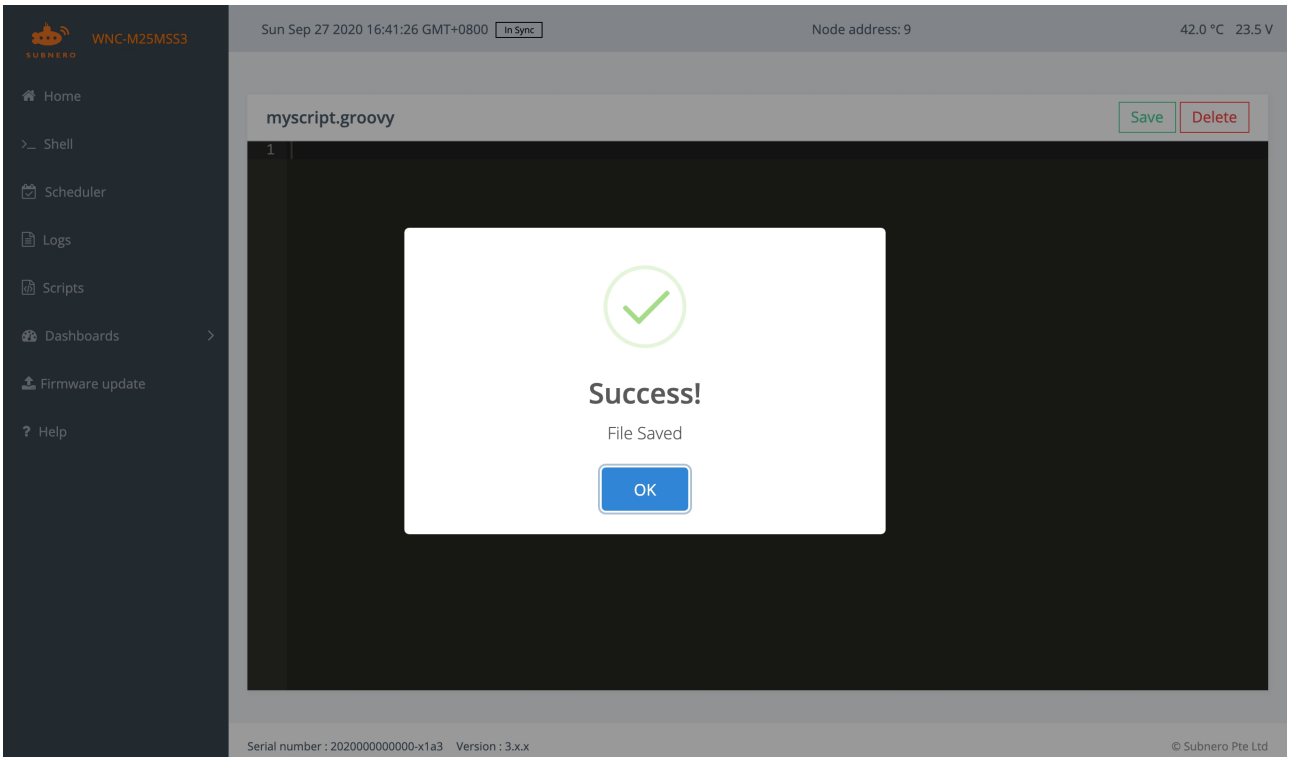

Figure 25. Script editor

4. Once done, click "Save" button to save the script.

If the user prefers to create/write the script or class in his/her computer, they can choose to upload the script to the scripts folder later. To upload a script or class or

a jar file, click the "Upload" button, choose the file and click the "Upload" button again. Once completed, the script, class or jar file can be accessed from the shell.

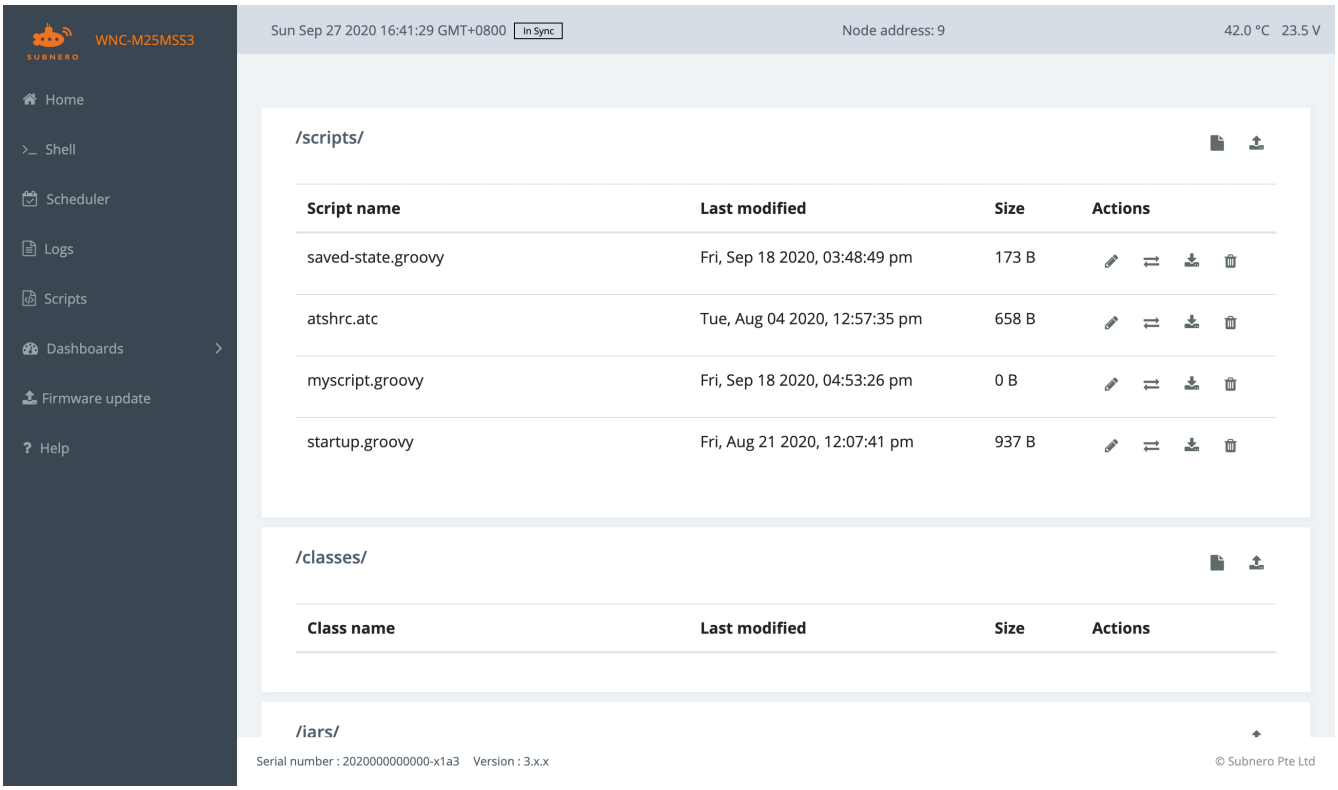

Figure 26. Uploading a script

The various folders listed in the "Script Editor" page are as follows.

1. Scripts folder: All scripts located in this folder can be accessed from the shell by the user.
2. Classes folder: All classes or groovy files in this folder will be in Java's CLASSPATH so that users can access them from their scripts.
3. Jars folder: Any jar files in this folder will be in Java's CLASSPATH so that users can access them from their scripts.

4.5. Dashboards

By default, the modem ships with certain dashboards that might be useful for a user. The available dashboards can be viewed by clicking on the `Dashboard` in the navigation menu. This contains various web-based dashboards for the modem in use. You can create and add your own Dashboard to UnetStack. Dashboards are simple HTML files, which are served by UnetStack. Any HTML files in the `scripts` directory of UnetStack will be served on the URL scheme `http://<modem-ip>/scripts/<dashboard-name>.html`.

By clicking on the `Dashboard` in the navigation menu, a user can see the following dashboards:

1. Diagnostic Scope
2. Speed Test
3. Overview
4. Configurations (beta)

4.5.1. Speed Test

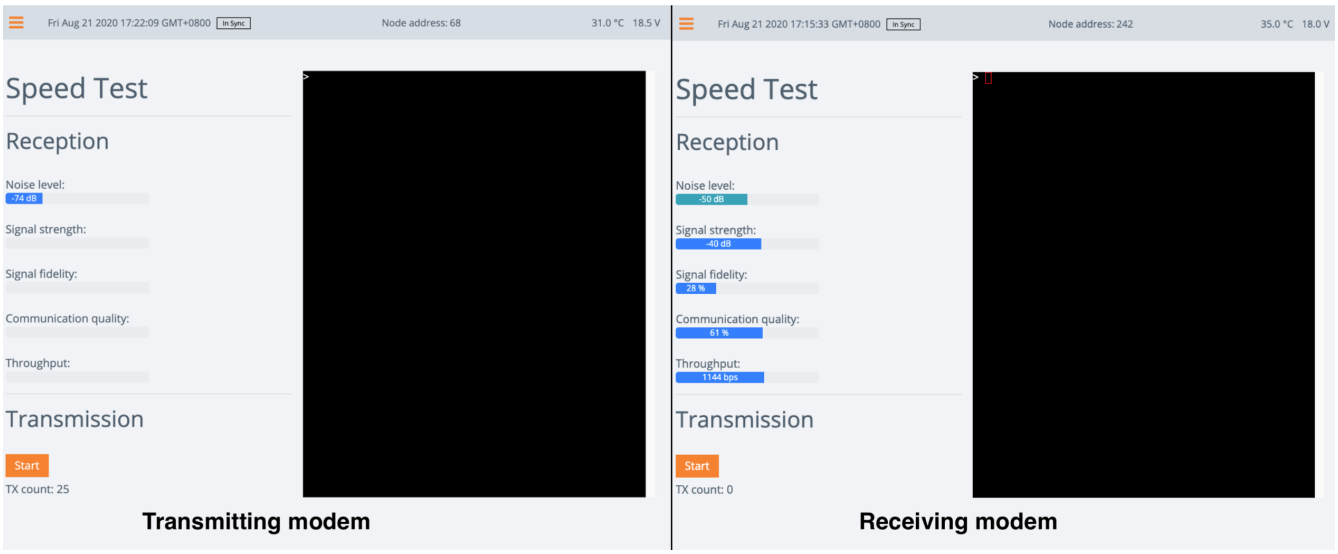

Figure 27. Dashboard for speed test

The objective of the speed test dashboard is to measure data rate (speed in simple words). To use this dashboard to measure data rate, follow the steps described here:

1. Deploy 2 modems in water at some distance and bring up the web interface on both the modems.
2. Click on the `Dashboards` section on the left side of the web interface and select `Speed Test` on the transmitting modem and receiving modem.This should open a page as shown in Figure 27:
3. Click on `Start` button and this starts transmitting frames with the `DATA` scheme (i.e., `phy[2]` modulation scheme) continuously. **NOTE:** Make sure the transmission power level is set appropriately.
4. On the receiving modem, you will be able to see the metrics that are computed based on the frames that are received as shown in figure above. The most important metric here is `Throughput` which shows the effective throughput (in bps) over the communication link on which these transmissions and receptions are being carried out.
5. To stop the test, click on `Stop` button on the transmitting modem to stop the transmissions.

Metrics

1. **Noise level**: The ambient noise level is averaged and continuously updated on this bar.
2. **Signal strength**: The signal strength is computed based on the received signal strength indicator (RSSI).
3. **Signal fidelity**: The signal fidelity is computed based on the detector output. The detector running at the receiving modem outputs a value based on how good a detection it has made.
4. **Communication quality**: The communication quality is computed based on the bit error rate (BER) measured on the receiving frames. This metric is a good indication of the communication quality in the channel.
5. **Throughput**: The effective throughput is measured based on the number of packets transmitted and received.

4.6. Firmware update

The firmware update page lets the user update the modem firmware.

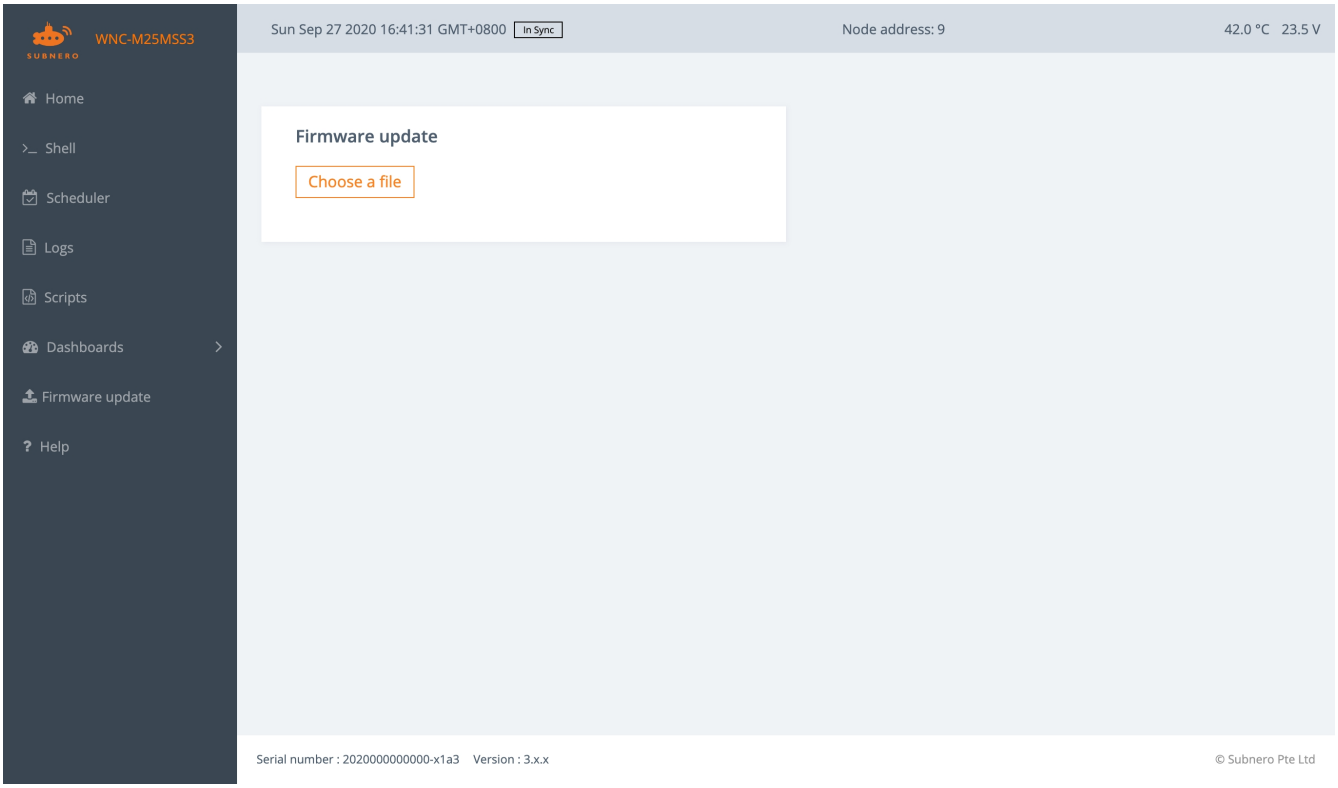

Figure 28. Firmware update page

The steps to update the modem firmware is listed below:

1. Contact Subnero support to get the latest firmware for your device.
2. Click the "Choose a file" button and point to the downloaded firmware file.
3. Click "Yes".

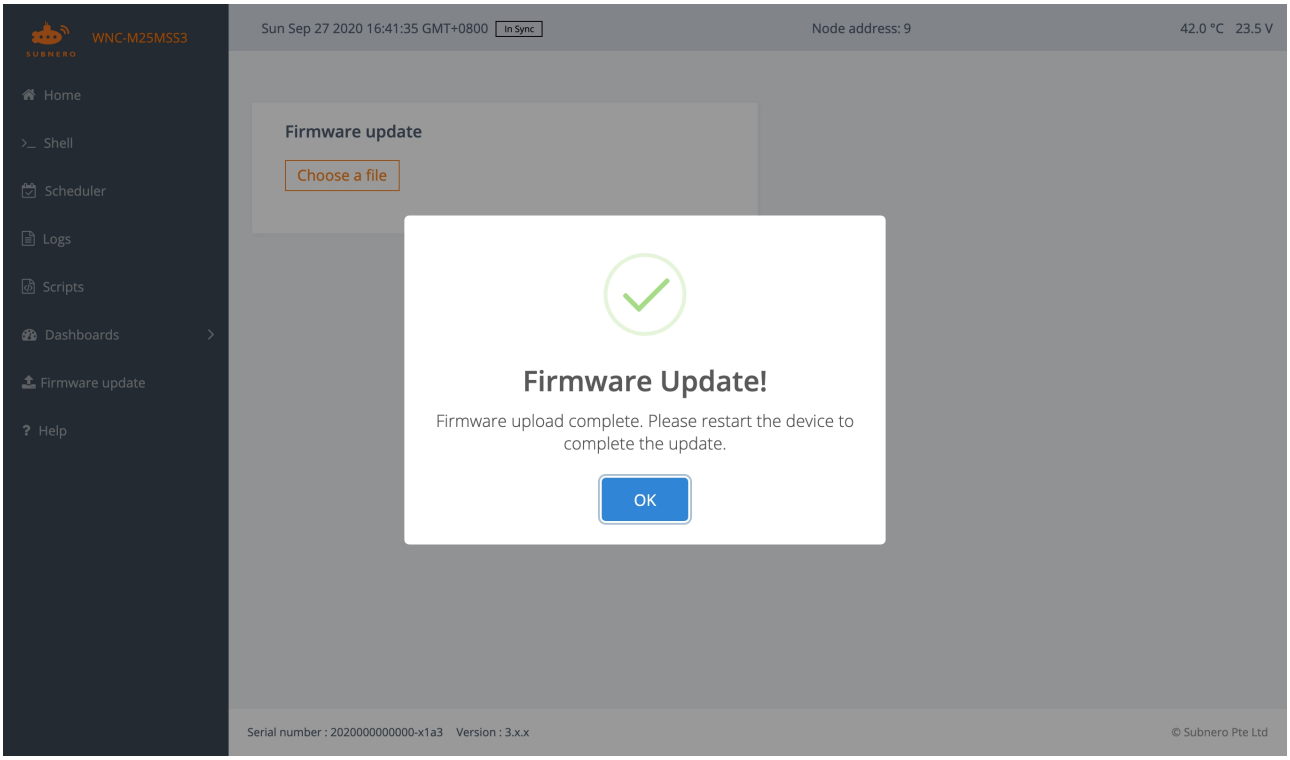

Figure 29. Firmware upgrade

4. Reboot the modem.

Chapter 5. Software Operations

The primary function of the modem is to be able to transmit and receive data. In addition, Subnero modems also support the transmission and reception of arbitrary signals, ranging, etc. This section explains how these functions can be executed using the modem's web interface using the web shell or the dashboard.

The command shell is great for manual configuration and interaction, but often we require programmatic interaction from an external application. For this, we have the UnetSocket API (available in Java, Groovy, Python, Julia, and C). While the exact syntax differs across languages, the basic concepts remain the same.

Refer to UnetSocket API section of Unet Handbook for more details on how to access the modems using the UnetSocket APIs.

5.1. Modes of operation

The modem has 3 different modes of operation:

1. Transmit mode
2. Receive mode
3. Sleep mode

The modem will be in transmit mode only for the duration of the transmission, and will automatically switch to receive mode to listen for incoming data after the transmission is completed.

The user may manually put the modem into sleep mode to conserve power when data transmission or reception is not expected for an extended period.

5.2. Transmit and receive a frame

The transmission of a frame is one of the basic functionality needed in the modem. Each frame can carry data. In the following examples, we show an example of transmitting a CONTROL or DATA frame using the web interface.

Refer to the overview section of Physical service chapter of the Unet Handbook for the definitions of CONTROL and DATA frames.

5.2.1. Transmit a frame

To transmit a CONTROL frame containing 7 bytes of data using the web interface, type the following command in the web shell:

```
phy << new TxFrameReq(type: CONTROL, data: [1,2,3,4,5,6,7], to: 0)
```

Similarly, in order to transmit a DATA frame:

```
phy << new TxFrameReq(type: DATA, data: [1,2,3,4,5,6,7], to: 0)
```

`phy` is the AgentID of the Physical agent running in the UnetStack. When a message `TxFrameReq` with relevant data and type is sent to the Physical agent, the frame is transmitted in the medium and a `TxFrameNtf` message is sent back to the web shell notifying that the transmission was successful along with the precise time at which the transmission started. The contents of this `TxFrameNtf` can be accessed on the web shell using the `ntf` variable. Type `ntf` in the web shell to look at the contents.

In order to receive this notification message in a variable other than `ntf`, the user can type the following command in web shell:

```
phy << new TxFrameReq(type: DATA, data: [1,2,3,4,5,6,7], to: 0); txntf =
receive(TxFrameNtf, 2000); println txntf.txTime
```

The above code snippet receives the `TxFrameNtf` message, stores it in a variable called `txntf` and prints the transmission start time.

Note that the field `to` in the `TxFrameReq` message is the address of the intended receiver node for this transmission. In this case, we have set it to `0`, which means that it is a broadcast frame and will be decoded by all the modems in the communication range of the transmitting modem.

5.2.2. Receive a frame

In order to receive a frame on the receiving modem, type the following command on the web shell:

```
subscribe phy
rxntf = receive(RxFrameNtf, 2000); println rxntf.data
```

On successful detection and decoding of the frame at the receiver modem, a `RxFrameNtf` message containing data is generated by UnetStack and published on the Physical agent's topic. Any agent subscribing to that topic will receive the `RxFrameNtf` message. The code snippet above will receive the `RxFrameNtf` message and print the data received.

Refer to Physical service chapter of the Unet Handbook for more details on transmitting and receiving CONTROL and DATA packets.

5.3. Transmit & receive arbitrary waveform

The modem supports arbitrary waveform transmissions. The signal to be transmitted can either be a baseband or a passband signal. In both cases, the sample values must be normalized between +1 and -1. When transmitting a baseband signal, the signal array should contain alternate real and imaginary values.

Refer to Baseband service section of the Unet Handbook for more details on arbitrary waveform transmission and reception.

5.3.1. Streaming style functionality

The modem supports the continuous recording of data in baseband and passband representation.

To record signals in baseband format, user can use the Physical agent's parameter `bbscnt`. Setting `phy.bbscnt = p` will return `p` number of baseband recordings to the user. Each of the recordings will contain `phy.bbsblk` number of baseband samples.

Setting `phy.bbscnt = -1` records back to back baseband signals indefinitely. To stop the recording, set `phy.bbscnt = 0`.

Similarly, in order to continuously record signals in passband format, user can use the Physical agent's `pbscnt` parameter. Setting `phy.pbscnt = -1` records back to back passband signals with `phy.pbsblk` samples in each recording.

Type `help phy.pbscnt` and `help phy.pbsblk` to see the documentation of these commands.

```
> help phy.pbscnt
phy.pbscnt - number of passband data blocks to stream

Setting this parameter starts streaming of passband data for
a specified number of blocks. A value of 0 stops streaming.
A value of -1 enable streaming forever.

Example:
  phy.pbscnt = 10       // stream 10 blocks of pasband data

> help phy.pbsblk
phy.pbsblk - passband streaming block size (samples)
>
```

5.4. Ranging

Subnero modems ship with a Ranging agent for measuring the range between two modems.

Refer to Ranging and synchronization of the Unet Handbook for details on ranging operation using Subnero modems.

5.5. Configuration of parameters

A modem is configured for use of specific applications by setting various parameters. To configure various modem parameters getters and setters are implemented.

UnetStack basics chapter of the Unet Handbook explains how various modem

> parameters can be configured.

Chapter 6. Maintenance & Support

6.1. General maintenance

Avoid metal to metal contact during deployment.

Avoid scratching, hitting or bending any of the anodized surfaces.

After every use it is highly recommended that the modem is cleaned and inspected as follows:

1. Thoroughly wash modem and underwater cables with fresh water. After each deployment, wash off the seawater from the hull and any underwater cables with fresh water. Wash thoroughly before long term storage to remove any seawater residue.
2. With a clean cloth, dry the modem.
3. Inspect for any damage to the modem such as cracks, dents, or scratches. If such damages begin to cause corrosion or have the potential to allow water ingress, contact the supplier.
4. Inspect the transducer and transducer cage for any bends, or dents. If such damages occur, contact the supplier.
5. Inspect all fasteners, replace them if damaged.
6. Remove eye-bolts from end-cap and thoroughly wash with fresh water. Inspect eye-bolts and thread for any corrosion, galvanic or otherwise. Replace eye-bolts, if damaged. Contact supplier if the heli-coil is damaged beyond use.

6.2. Diagnostic information

The modem performs a Power-ON Self Test (POST) while booting up. The results of these tests are logged in the log file and the overall result is available through the following command:

```
> phy.post
0
```

On success, the value of the parameter `phy.post` is `0`. A non-zero value is an indication that one of the tests might have failed. The result of the test is available through the `modem_selftest()` function call. If successful, the function call returns 0, or on failure, a negative return code. If detailed self-test output is desired, the user may take a look at the `log-0.txt` file in the `Logs` folder as given in Section 4.3. An example of the diagnostic output is given below.

```
1558009228645|INFO|org.arl.yoda.POST@34:output|[TEST] --------------------------------------------------------------------------------
1558009228645|INFO|org.arl.yoda.POST@34:output|[TEST] POST
1558009228645|INFO|org.arl.yoda.POST@34:output|[TEST] --------------------------------------------------------------------------------
1558009228682|INFO|org.arl.yoda.POST@34:output|[TEST] Thu May 16 12:20:28 UTC 2019
1558009228682|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009229003|INFO|org.arl.fjage.connectors.WebSocketConnector@16:onConnect|New connection from /192.168.1.101:54269
1558009229922|INFO|org.arl.fjage.connectors.WebSocketConnector@11:onConnect|New connection from /192.168.1.101:54270
1558009232871|INFO|org.arl.fjage.connectors.WebSocketConnector@12:onConnect|New connection from /192.168.1.101:54271
1558009233537|INFO|org.arl.unet.nodeinfo.NodeInfo@31:obtainAddress|Node name is unet-da140216, address is 241, address size is 8 bits
1558009233563|INFO|org.arl.yoda.Physical@36:run|RThread started
1558009233610|INFO|org.arl.yoda.POST@34:output|[TEST] # Information
1558009233610|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009233613|INFO|org.arl.yoda.POST@34:output|[TEST] * Platform version:                fjage-1.5.2-SNAPSHOT/17-02-2019_01:41:05
1558009233618|INFO|org.arl.yoda.POST@34:output|[TEST] * PHY version:                     1.4/25-03-2019_07:15:13
1558009233623|INFO|org.arl.yoda.POST@34:output|[TEST] * Make/model:                     Subnero/WNC-M25MSS3
1558009233627|INFO|org.arl.yoda.POST@34:output|[TEST] * Serial number:                  2019021300000-k1a3
1558009233627|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009233629|INFO|org.arl.yoda.POST@34:output|[TEST] # Hardware
1558009233630|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009234667|INFO|org.arl.yoda.POST@34:output|[TEST] * PHY clock:                      PASS
1558009234911|INFO|org.arl.yoda.POST@34:output|[TEST] * Noise level:                    -74.5 dB
1558009236391|INFO|org.arl.yoda.POST@34:output|[TEST] * Loopback:                       PASS
1558009237323|INFO|org.arl.yoda.POST@34:output|[TEST] * Gain control:                   PASS
1558009238407|INFO|org.arl.yoda.POST@34:output|[TEST] * Temperature:                    34.0 C
1558009238412|INFO|org.arl.yoda.POST@34:output|[TEST] * Battery voltage:                0.0 V
1558009238412|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009238414|INFO|org.arl.yoda.POST@34:output|[TEST] # Baseband
1558009238414|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009238418|INFO|org.arl.yoda.POST@34:output|[TEST] * Carrier frequency:            24000.0 Hz
1558009238421|INFO|org.arl.yoda.POST@34:output|[TEST] * Baseband rate:                 24000.0 Hz
1558009238421|INFO|org.arl.yoda.POST@34:output|[TEST]
1558009238423|INFO|org.arl.yoda.POST@34:output|[TEST] --------------------------------------------------------------------------------
1558009238423|INFO|org.arl.yoda.POST@34:action|[TEST] Summary: PASS
```

Figure 30. Example diagnostic output

6.3. Frequently Asked Questions (FAQ)

1. I lost the static IP address setting of the modem. How can I find the current static IP address of the modem?

   By default, the modem's static IP address is in the `192.168.42.x` subnet.

2. Connect your computer to the modem.
3. Set the computer to have a static IP in the `192.168.42.x` subnet for the interface connected to the modem.
4. On Linux, open a terminal and type `arp -a`. That should list all the IP addresses connected to your laptop. One of them (in the `192.168.42.x` subnet) will be the modem's IP address.
5. On Windows, you can use any network scanning utility (e.g. IP Scanner) to scan your network and find the IP address.
6. You can open a browser from your computer and key in this IP address to access the modem's web interface.

   If the above doesn't work, you can also use dynamic IP (DHCP) to connect to the modem. For this, you will need a WiFi router with admin access (or any device that has a DHCP server running).

7. Connect the modem to the router
8. Connect your computer to the router. (Do not configure your laptop with static IP in this case).

9. Login to the router and look for IP addresses of all the connected devices. One of the connected devices will be the modem.
10. You can open a browser from your computer and key in this IP address to access the modem's web interface.
11. I am trying to communicate between two modems in air. However, I am getting `BadFrameNtf`. How can I fix this?

    The modems are meant to be operated underwater. If you must transmit in air, you need to place the modems right next to each other, set a power level lower than `-42 dB` (using `plvl` command) and transmit.

12. I am trying to communicate between two modems in a bucket of water. I am unable to get it to work. What do I do?

    Small water bodies like a bucket or a tub of water are very reverbant environments. If you are trying to get two modems to communicate in a small space, make sure the power level is set to `-70 dB` or lower (using `plvl` command). If that doesn't work, experiment by lowering it further (e.g. `-75 dB`). If you see `BadFrameNtf`, that means the modems are able to detect each other's packets, but unable to decode.

6.4. Support

- Email: support@subnero.com
- Website: https://subnero.com/support/

Appendix A: Technical Specifications

Table 1. Specifications

Item	Value
Edition	Silver edition
Configuration	Surface configuration
Model number	M25MSS3
Data rate	Up to 15 kbps (depending on channel conditions and reliability requirements)
Operating range	3-5 km (nominal, depending on channel conditions)
Ranging precision	0.1 m
Doppler resilience	up to ±4 knots
Modulation (software-defined)	PSK-OFDM, FH-BFSK
FEC (Forward Error Correction)	LDPC, up to $^{1}/_{6}^{th}$ rate code
Software framework	UnetStack3 (www.unetstack.net)
Software interface	UnetStack3 (Java, Groovy, Python, C, MATLAB, JavaScript, Julia), interactive web UI, JSON/TCP
Hardware interface	Ethernet, power
Transducer beam pattern	Omni-directional
Carrier frequency	24 kHz
Bandwidth	12 kHz (20 - 32 kHz)
Source level	185 dB re 1 µPa (rms) @ 1 m (nominal)
Power consumption	• < 4 W (receive mode, nominal) • < 60 W (transmit mode, avg.) • < 80 W (transmit mode, max.)
Power source	External power: 22-28 V DC (24 V DC recommended)
Operating depth	300 m (Aluminum hull)
Modem weight (in air / water)	6.0 / 1.0 kg
JANUS compatibility	Yes, subject to operating frequency band
Wake up module	Included (Ethernet)
On-board storage	32 GB
Arbitrary waveform transmission & reception	Included

Table 2. O-ring specifications

Item	Value
Vent screw O-ring	2 mm wide, 6 mm ID

Appendix B: Mechanical Drawings