# Operations Manual for the Progammable Hyperspectral Seawater Scanner

Jonathan Turner
Mote Marine Laboratory Technical Report
Ocean Technology Group

November 26, 2019

**Abstract**

The Programmable Hyperspectral Seawater Scanner (PHySS) is a next-generation marine spectrophotometer based on Mote Marine Laboratory's Optical Phytoplankton Detector (OPD). Like the OPD, the PHySS performs automated sampling and analysis of seawater in order to detect the presence of various pigments associated with harmful algae blooms, such as *Karenia Brevis*. Unlike the OPD, the PHySS can be programmed by users, allowing far greater flexibility in both the sampling procedure and the analysis of the resulting data. In addition, it has two hardware configurations, one of which allows two chemical reagents to be mixed with seawater samples prior to optical spectrum acquisition. This report provides a detailed description of the PHySS to allow operators to deploy and use it effectively. [1]

[1] This report has been formatted to be read conveniently on an ipad or similar tablet. Paper copies are best printed with two pages per sheet in landscape mode.

Since early 2017, the Mote Research Laboratory's Ocean Technology Group has been developing a next generation version of the Optical Phytoplankton Discriminator (OPD) [1, 2]. The new system, called the Programmable Hyperspectral Seawater Scanner (PHySS) was designed to build on the established legacy of the OPD, while introducing a number of improvements.

- upgraded hardware components, including new pumps, valves and a more sensitive spectrophotometer (16 bits vs 12),
- an extended configuration that enables two separate reagents to be mixed with seawater samples in user-controlled proportions,
- a fully programmable data acquisition process, giving users more control over how data is collected,
- a web-based control interface, allowing remote users to operate the PHySS without having to install special-purpose software on their computers,
- automatic transmission of complete hyperspectral data to a cloud-server, giving users access to full spectra, not just summary metrics,
- a web-based data analysis tool, enabling users to review raw data from all operating PHySS units and to process that data in a variety of ways.

This report provides a detailed description of the PHySS and how it is used.

# 1 Overview of hardware and software

Figure 1 is a diagram showing the first of the two PHySS hardware configurations. Seawater enters the system through one of two entry/exit ports and is pumped by a peristaltic *sample pump* through a filter and valve combination that allows either filtered or unfiltered seawater to enter an optical

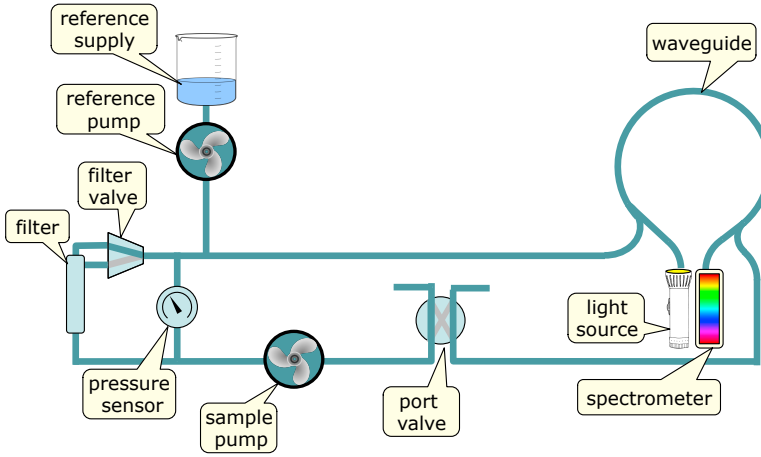


**Figure 1 Basic hardware configuration**

waveguide, where it is illuminated by a light source to produce an optical spectrum in the visible light range (the spectrometer acquires 16 bit intensity values in 2048 wavelength bands from 190 to 880 nm). The configuration of the *port valves* is changed periodically to reverse the roles of the entry and exit ports; this prevents large particles in the seawater from accumulating on the mesh screens that keep such particles out of the internal plumbing. A separate *reference pump* is used to pump a reference fluid into the waveguide (the reference fluid is typically seawater collected from far off-shore). The spectrum of the reference fluid is compared to the spectrum of seawater samples during the data analysis process. A pressure sensor is used to monitor the pressure across the filter when collecting a filtered seawater sample, so as to avoid exceeding the transverse pressure limit on the filter (30 psi). While shown on the diagram as a single component, there are actually two pressure sensors, one measuring the pressure *upstream* of the filter and valve and one measuring the pressure *downstream* of the filter and valve. The difference between these two values is the pressure across the filter. The downstream sensor is also used to determine the approximate depth of the unit in the water. The basic hardware configuration also includes a battery sensor and

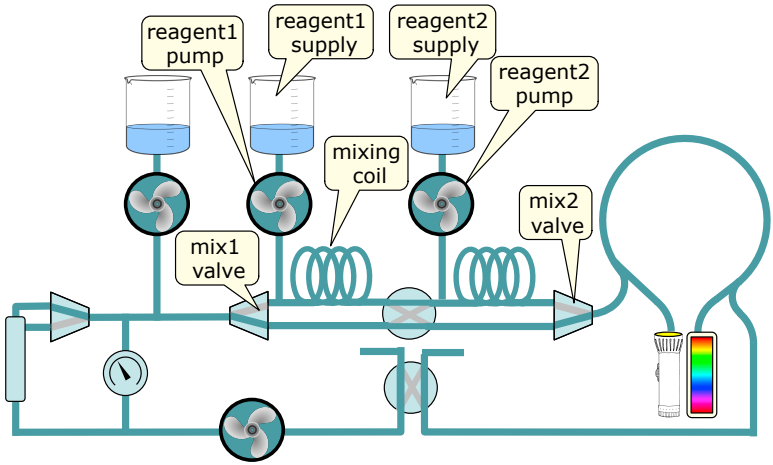


**Figure 2 Extended hardware configuration**

a temperature sensor, which are not shown.

The PHySS hardware is controlled by a *Beaglebone* processor [3], a 32 bit embedded processor that runs the Linux operating system. The processor has a variety of capabilities. In the PHySS context, the key features are an Ethernet network interface and a USB interface used to communicate with an Ocean Optics spectrometer module and an Arduino microcontroller, which is used to control the various hardware components (valves, pumps, lights, sensors).

Figure 2 shows the second of the two PHySS configurations. This configuration adds two reagent pumps with associated supply reservoirs, plus a system of valves and mixing coils that allow two distinct reagents to be mixed with seawater samples in user-specified proportions.

The major software components of the PHySS are shown in Figure 3. The *Data Collector* is the component that operates the PHySS hardware and implements the automated data collection process. Data collection is performed as a series of numbered *sample cycles* where the steps in each sample

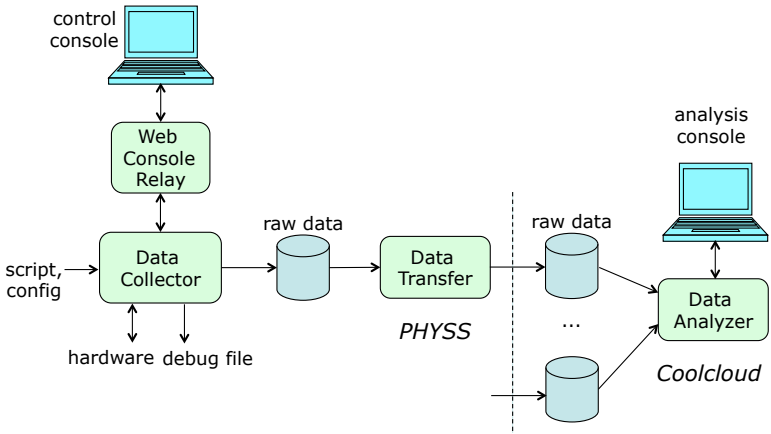


**Figure 3 Software components**

cycle are determined by a user-specified script. A *configuration file* is used to specify various operational parameters.

The PHySS is operated through a web-based *operations console*. The console is implemented as a JavaScript program that is downloaded from the PHySS to a user's web browser and runs within the browser. Consequently, there is no need for users to install software in order to use the PHySS. An*operations server* component serves as an intermediary between the remote console and the data collector. A separate *data transfer* component sends data to a remote cloud server, where it can be accessed using a *data analysis console*. The analysis console can also be run from an individual PHySS unit; this feature is mainly used to allow the PHySS operator to verify that the system is operating properly when first starting up.

# 2 Overview of PHySS Operations

Figure 4 shows the control console running in a web browser. When performing automated data collection, the central graphic display is updated

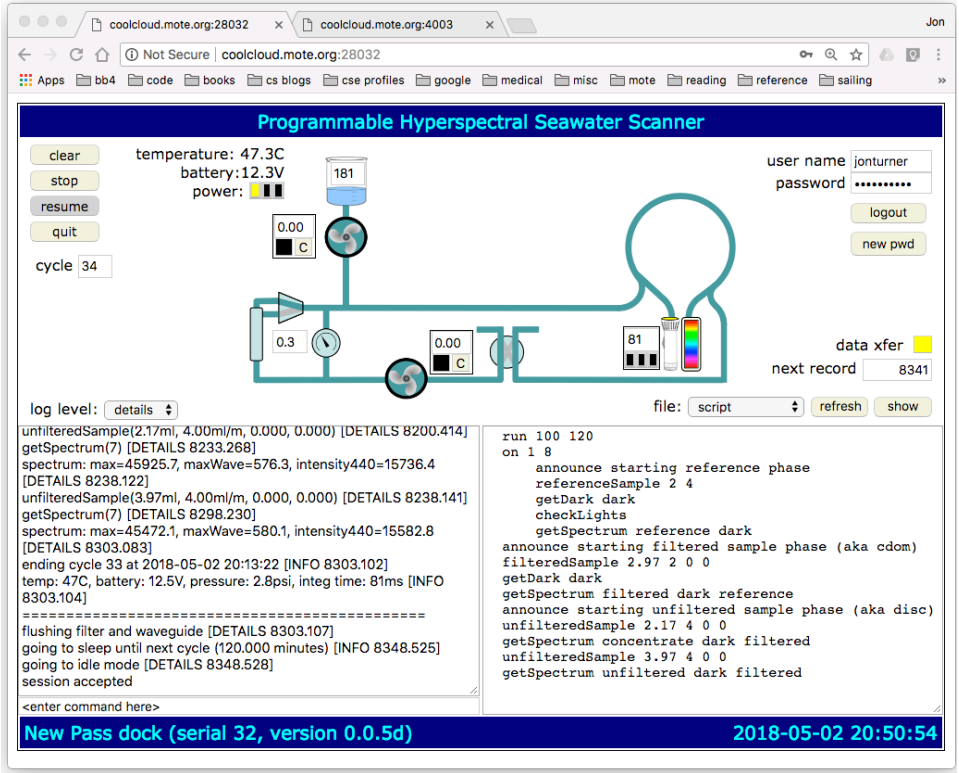


**Figure 4 Control console**

continuously to show the state of the hardware. Users can also use the console to control the PHySS hardware manually by interacting with the graphic representation. To change a valve state the user simply clicks on the valve. This causes a command to be sent from the user's brower to the relay, which forwards the command to the data collector. The data collector sends a reply message, which triggers an update to the graphic display showing the new valve state.

Each pump in the system has an associated set of controls. There is a text box used to specify a pump rate (in milliliters per minute), an on/off button

and a *calibration button* (labeled with a "C"). To turn a pump on, just enter a non-zero pump rate in the text box and click the on/off button. This causes a command to be sent to the data collector, which turns the pump on and sends an acknowledgment message, causing the display to be updated. The on/off button is yellow when the pump is running, black when it is not. The reference pump has an associated *supply reservoir*. The estimated amount of fluid in the reservoir is displayed (in milliliters). This can be adjusted (typically after fluid is added to the reservoir), and is updated whenever the reference pump is run, in order to track the remaining volume of fluid. If the estimated volume drops below 10 ml, the data collector will not turn the pump on.

The rate at which pumps move fluid through the system varies from pump-to-pump and also varies over time. Consequently, pumps must be calibrated from time-to-time to determine what the maximum flow rate is. To calibrate a pump, the user clicks on the calibration button, then measures the amount of fluid passing through the exit port, clicking a second time when 10 ml of fluid has been pumped. During the calibration process, the pump is run at its top speed and the data collector records the time required to pump 10 ml. This allows it to determine the maximum rate at which fluid flows when the pump is operated at top speed.

There is another set of controls for the spectrometer. This includes three buttons that control the light source. The light source includes two lamps, a *deuterium lamp* and a *tungsten lamp*. To turn on the deuterium lamp, click on the leftmost of the three buttons, to turn on the tungsten lamp, click on the middle button. The buttons are yellow when the lamp is on, black when it is off. The rightmost button controls the light source's *shutter*. This button is yellow when the shutter is open, black when it is closed. The textbox displays the spectrometer's *integration time*. This is the amount of time for a single spectrum acquisition (in milliseconds). This can be set manually, but is more typically adjusted by the automated data collection process in order to maximize the spectrometer sensitivity. When a PHySS unit is first deployed, the typical values are in the range of 50–100 milliseconds, but

over the course of a long deployment, changes in the light sources, the light detector and the waveguide may require the integration time to be increased (up to a maximum of 500 ms).

The temperature inside the PHySS enclosure (in degrees C) is shown just to the left of the reference fluid supply reservoir. If this exceeds 65 degrees, the system will shut down. The voltage for the battery that powers the PHySS is also displayed. The nominal value is 12 volts. If it drops below 10 volts, the system is shut down. There are two buttons used to control power to different parts of the system. The left button controls the power to the pumps and valves. The right button controls the power to the light source. When the system first starts up, both power controls are turned off. The user can turn them on manually by clicking on them. If the user attempts to operate a pump, valve or the light source while the power is off, the system will automatically turn them on and update the console display to reflect the chage in state. During automated data collection, both power controls are turned on during a sample cycle and turned off duriing the idle periods between sample cycles.

The buttons at the top left are used to control the automated data collection process. The *start/stop* button is used to start automated data collection or to stop it. When data collection is stopped, it can be "resumed" (using the *resume button*, or "restarted". On a restart, the sample cycle number is reset to 1; when resuming the sample cycle number is unchanged, and the interrupted sample cycle is restarted. The *clear button* is used to discard all the records in the raw data file and the file of debug messages. Note that while automated data collection is enabled, the manual controls are disabled. In order to control the hardware manually, the user must first stop the automated data collection. The quit button causes the data collector to terminate and the Beaglebone to shutdown.

Because the PHySS can be accessed over the internet, it is possible for multiple consoles to interact with a specific PHySS unit at the same time. To prevent conflicts among different users, only one is allowed to actually control

a specific PHySS at any given time. Others may observe the PHySS's activity, but may not operate the controls. The login area at the top right allows a single user to login to the PHySS, by entering a login name and password, then clicking on the *login/logout* button to gain access. If no other user is currently logged into the specific PHySS, the user's password is verified and the login session is either accepted or rejected. A user who is logged in can change the password by entering a new password in the password textbox and clicking on the button labeled "new pwd." If a logged in user does not interact with the console for 15 minutes or more, the login session is terminated, the console display is grayed out and a large *wakeup button* is shown. By clicking on this, the user can resume interacting with the PHySS. To take control again, the user must also click on the login button. This mechanism ensures that an inactive user does not prevent others from getting control of the PHySS.

The text area at bottom left of the console shows log messages generated by the data collector, as well as replies to commands sent from the console. The log messages all have a tag in square brackets at the end that shows the "level" of the message and the time at which it was generated (expressed in seconds since the time the data collector started running). The menu just above the textbox controls which log messages are displayed. Just below the large textbox is a one line textbox that can be used to manually send commands from the console. This feature is used infrequently, since most of the available commands can be issued more conveniently through the main control panel. The full list of commands is documented in the appendix.

The automated data collection process is controlled by two files whose contents are specified by the PHySS operator. The *configuration file* includes a variety of static configuration variables that are tailored to individual PHySS units and deployment situations. The *script file* specifies the sequence of steps that are performed during each sample cycle. The large textbox at the bottom right of Figure 4 displays the script. However, by selecting from the pulldown menu, users can also access the *configuration file*, a *collector state file*, and a *debug file* containing copies of debug messages. If a user is

logged in and automated data collection is turned off, the user can edit the the script and configuration files using the *edit/save/refresh* button. The *show/hide* button controls the display of comments in the script file.

The data transfer button on the the right side of the figure controls the transfer of data from the PHySS to the remote cloud server. When data transfer is enabled, the button is yellow, when it is disabled the button is black. The "next record" text box shows the record index of the next record to be transferred to the cloud server.

An example of a PHySS configuration file is shown below. The individual parameters are explained below.

```
# basic config file
serialNumber = 33
deploymentLabel = Sanibel Marker 2
gpsCoordinates = N26.4344/W81.9648

waveguideLength = 0.28
maxFilterPressure = 25
maxDepth = 20

hardwareConfig = BASIC
execMode = LAB
autoRun = -1
powerSave = 0
singleCycleMode = 0
portSwitching = 1
```

Config files may include comments, which are introduced with a pound sign '#' and extend to the end of the line.

The *serialNumber* parameter is a unique integer identifying a specific PHySS unit. PHySS serial numbers are restricted to values larger than 30. The older OPD units use smaller serial numbers.

The deployment label is a text label provided by the operator, which typically describes the location of the deployed unit.

The *gpsCoordinates* parameter specifies the GPS coordinates of a deployed PHySS unit. The first value is the latitude in degrees (with four fractional digits), the second is the longitude. The latitude includes a prefix 'N' or 'S' to indicate north or south. The longitude includes a prefix 'E' or 'W' to indicate east or west. This parameter is set by the operator and is included with the data sent to the cloud server.

The *waveguideLength* parameter specifies the length of the waveguide in meters. This value is used by the analysis console to compute absorption spectra.

The *maxFilterPressure* parameter specifies the maximum safe operating pressure across the filter in pounds-per-square-inch (psi). This is typically set slightly below the true maximum in order to provide some safety margin.

The *maxDepth* parameter specifes the maximum safe operating depth for the PHySS unit.

The *hardwareConfig* parameter specifies which of the two hardware configurations is used by the PHySS unit. The allowed values are BASIC and TWO_REAGENTS.

The *execution mode* parameter determines what the data collector should do when all of the sample cycles specified by the script have completed. In LAB mode, the data collector simply suspends script execution and returns to manual mode. In FIELD mode, it terminates and shuts down the PHySS.

The *autoRun* parameter determines whether automated data collection is started automatically or not. A negative value (typically $-1$) causes the data collector to start in manual mode. It is up to the user to start automated data collection through the console. A value of 0 causes automated data collection to start automatically, when the data collector starts. A positive value is

interpreted as a delay interval (in minutes); automated data collection begins after this delay interval has transpired.

When the *powerSave* parameter is 1, the system shuts off hardware components that are not needed during the periods between sample cycles. This typically includes the Beaglebone processor, the spectrometer and whatever communications device is used for communicating with the cloud server. This can be useful in situations where the system is running off a battery with limited capacity. One drawback of this mode is that an operator cannot interact with the PHySS between sample cycles, since the Beaglebone is powered off. It is still possible to connect to it during a sample cycle.

The *portSwitching* parameter is used to enable/disable the port switching feature. If the value is 1, the role of the entry/exit ports is switched after each sample cycle. If the value is 0, this feature is disabled.

As mentioned earlier, the automated data collection process is controlled by a user-specified script. An example of such a script appears below, followed by a detailed explanation of each command.

```
# basic PHySS operational script
run 100 120
on 1 8
    announce starting reference phase
    referenceSample 2 4
    optimizeIntegrationTime
    getDark dark
    checkLights
    getSpectrum reference dark
announce starting filtered sample phase (aka cdom)
filteredSample 3 2
getDark dark
getSpectrum filtered dark reference
announce starting unfiltered sample phase (aka disc)
```

```
unfilteredSample 2 4
getSpectrum concentrate dark filtered
repeat 2
    unfilteredSample 4 4
    getSpectrum unfiltered dark filtered
```

Like config files, scripts may include comments, which are introduced with a pound sign '#' and extend to the end of the line.

The *run* command must appear on the first line of any script. It has two arguments. The first specifies the number of sample cycles to be run prior to termination. A zero value causes data collection to continue indefinitely. The second argument determines the times at which sample cycle should be run. For example, a value of 120 minutes specifies that cycles should take place every two hours, starting on the hour. More generally, if the value is $n$ minutes, each automatic sample cycle will start at a multiple of $n$ minutes after midnight each day (the timezone is UTC). Sample cycles that are initiated manually by an operator take place on demand, so typically the first sample cycle of a deployment may take place any time, but subsequent sample cycles will be scheduled as specified by the run command.

The *on* command is used to specify operations that are only to be performed on some sample cycles. So for example, the command `on 1 8` specifies that the following commands are to be performed on the first of every group of eight cycles (so, sample cycles with numbers, 1, 9, 17, 25, etc). Indenting is used to identify which commands are controlled by a given on command (each level of indenting must add at least two spaces and the number of added spaces must be consistent). So, in the example, the five commands following the on comand are performed every eight sample cycles, while all other commands are performed every cycle. A script may include multiple on commands, but nesting of on commands is not recommended.

The *announce* command specifies a message that is to be displayed on the operations console whenever the command is executed. This can make it easier for an operator to monitor the execution of a script.

The *referenceSample* command causes the PHySS to pump reference fluid into the waveguide. The two arguments specify the volume of fluid to be pumped (in milliliters) and the rate at which the pump should operate (in milliliters per minute). So the command `referenceSample 2 4` specifies two milliliters of reference fluid at the rate of 4 milliliters per minute. The range of pump rates depends on the specific pump being used, but the maximum rate is typically between 7 and 8 milliliters per minute. Pump rates are not at all precise; they should be interpreted with a precision of $\pm 10\%$.

The *optimizeIntegrationTime* command causes the PHySS to adjust the spectrometer's *integration time* parameter. This specifies the amount of time (in milliseconds) that the light detector is exposed to the light passing through the waveguide. The integration time is adjusted to produce light measurements near the top end of the range of possible values. This is done to maximize the sensitivity of the spectrometer to the incoming light.

The *getDark* command causes the PHySS to acquire a "dark" spectrum and associate a label with it. So the command `getDark dark` associates the label "dark" with the acquired spectrum. When a dark spectrum is acquired, the light source is turned on, but the shutter for the light source is kept closed, preventing the light from reaching the sample in the waveguide. Dark spectra are used to compensate for the fact that the spectrometer's light detector produces a non-zero reading even when no light is reaching the detector. Dark spectra are subtracted from sample spectra during data analysis to compensate for this effect. The light source is turned on, when acquiring a dark spectrum to enable compensation of the electrical noise generated by the light source. The acquired spectrum is saved in the raw data file produced by the PHySS along with its label.

The *checkLights* command causes the PHySS to perform a simple test on the light source, to ensure that it is operating correctly. If it is not, then an error message is displayed on the console and written to the debug file and raw data file.

The *getSpectrum* command causes the PHySS to acquire a spectrum of the seawater sample currently in the spectrometer's waveguide and associate a

label with it. So the command `getSpectrum reference dark` associates the label "reference" with the acquired spectrum. The second argument identifies a *prerequisite spectrum* associated with the newly acquired spectrum. In this example, there is a single prerequisite spectrum, the most recently acquired spectrum with the label "dark." When the reference spectrum is saved in the raw data file, the saved record includes a "pointer" to the most recent dark spectrum, so that the dark spectrum can be accessed during data analysis. A spectrum may have up to two prerequisite spectra.

The *filteredSample* command causes the PHySS to pump a seawater sample into the waveguide, with the filter valve configured to pass the sample through the filter. It has two required arguments plus two optional arguments. The first argument specifies the volume of fluid to be pumped (in milliliters) and the second specifies the rate at which to pump (in milliliters per minute). The two optional arguments are only used in the TWO_REAGENTS hardware configuration and determine how much of the sample is to be pumped from each of the two reagent supply reservoirs. So for example, the command `filteredSample 3 2 .2 .3` specifies that a total of 3 ml of fluid should be pumped at a total rate of 2 ml/m, with 20% of the total volume coming from the first reagent supply reservoir and 30% coming from the second reagent supply reservoir (so 50% is fresh seawater). The system chooses rates for all three pumps to produce the specified volumes and aggregate pump rate. During the acquisition of a filtered sample, the system monitors the pressure across the filter valve and if at any time it exceeds the specified maximum pressure, the operation is interrupted and the entire sample cycle is restarted. If the cycle continues to fail, automated sampling is suspended, after 10 failed attempts.

There is a second filtered sample command called *filteredSampleAdaptive*. This command does not include a pumping rate argument. Instead, it adapts the pump rate(s) in response to the pressure readings. If the pressure reading exceeds a safe target range, it reduces the pumping rate and if the pressure drops below the target range, it increases the pumping rate.

The *unfilteredSample* command has the same arguments as the filteredSample command. It causes the PHySS to pump a seawater sample into the

waveguide, with the filter valve configured to bypass the filter. Any particles that accumulated within the filter during a previous filtered sampling operation are flushed through to the waveguide, along with the new seawater sample. This increases the concentration of particles in the sample, effectively increasing the sensitivity of the spectrometer.

The *repeat* command causes a group of commands to be executed repeatedly. Its one argument specifies the total number of repetitions. The commands to be repeated are specified using indentation. Repeat commands may be nested.

The output of the data collector is a series of data records formatted as JSON strings. Most of the output consists of spectrum records, but the output also includes various metadata records, including records containing the script and configuration files used to produce the spectra, plus status messages generated during each sample cycle. The output is written to a raw data file on the PHySS and can be transferred from there to a remote cloud server (coolcloud.mote.org) where it can be accessed by users from anywhere in the world. A small excerpt from the raw data file is shown below (records have been abbreviated for clarity).

```
{ "serialNumber": 31, "index": 33985, "recordType": "deployment",
"dateTime": "2018-05-07 19:34:08", "label": "OT Lab", "wave
guideLength": 0.2800, "gpsCoord": "0.0/0.0", "wavelengths":
[188.65, 189.04, 189.43, 189.82, 190.21,  ...]}
{ "serialNumber": 31, "index": 33986, "recordType": "config",
"dateTime": "2018-05-07 19:34:08", "deploymentIndex": 33985,
 "configString": "logLevel = DETAILS@@@@maxFilterPressure =
 30@@gpsCoordinates ...}
{ "serialNumber": 31, "index": 33987, "recordType": "script",
"dateTime": "2018-05-07 19:34:08", "deploymentIndex": 33985,
 "scriptString": "run 100 120@@    # 100 cycles, 120 minute
between cycles@@ ...}
{ "serialNumber": 31, "index": 33988, "recordType": "debug",
```

```
"dateTime": "2018-05-07 19:34:08", "deploymentIndex": 33985,
"message": "going to idle mode [DETAILS 47.241]" }
{ "serialNumber": 31, "index": 33989, "recordType": "debug",
"dateTime": "2018-05-07 19:34:08", "deploymentIndex": 33985,
"message": "purging air bubbles [DETAILS 47.244]" }
{ "serialNumber": 31, "index": 34008, "recordType": "spectrum",
"dateTime": "2018-05-07 19:35:51", "deploymentIndex": 33985,
"prereq1index": 0, "prereq2index": 0, "label": "dark",
"spectrum":  [0.00, 0.00, 1608.76, 1551.00, 1569.68,  ...]}
{ "serialNumber": 31, "index": 34015, "recordType": "spectrum",
"dateTime": "2018-05-07 19:36:10", "deploymentIndex": 33985,
 "prereq1index": 34008, "prereq2index": 0, "label": "reference",
spectrum": [0.00, 0.00, 1591.77, 1561.67, 1590.80, ...]}
```


Note that every record starts with the *serial number* of the specific PHySS unit, followed by a *record index*. Record indices are unique for a given PHySS unit. Each time the data collector starts a new data collection run, it generates a *deployment record*, which includes some metadata information, including the wavelength values used by the PHySS unit's spectrometer (these wavelengths can vary slightly from one spectrometer to another). Other records in the given run include a reference to the deployment record's index so that the data analysis console can easily locate the wavelength information associated with a given spectrum record. The *config record* contains a copy of the configuration file used to set the values of various data collector configuration variables. Similarly, the *script record* contains a copy of the script used for the specific run. The data analyzer on the cloud server reads raw data files and presents their contents to the user.

# 3 Connecting to a PHySS

Because the PHySS is an internet-resident device, it is subject to attack from internet hackers . This is not a hypothetical issue; hackers routinely search

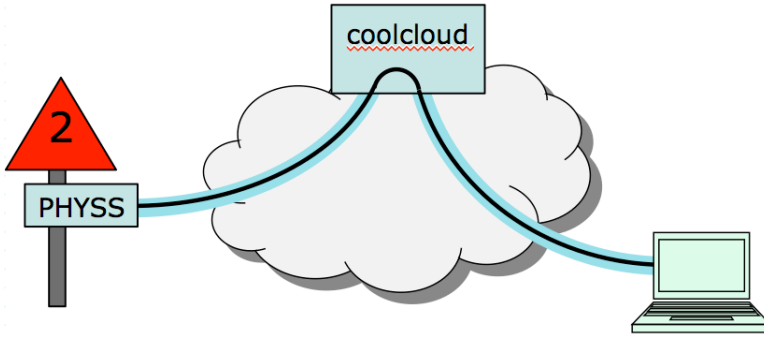


**Figure 5 SSH tunnel to PHySS through cloud server**

for poorly protected systems and probe for weaknesses. To limit exposure to such attacks, access to the PHySS control console is carefully managed.

The PHySS handles all communication using the *Secure Shell* protocol (SSH). SSH enables users to access a remote computer system and communicate with it securely. All communication is encrypted and the SSH software provides a mechanism that allows users to login to the remote system using public key encryption, eliminating the need for passwords. In addition, web browser traffic can routed through an SSH session, allowing authorized users to gain access to private web applications, while excluding others. This feature (called *SSH tunnels*) is used to provide secure access to the PHySS control console and to enable the PHySS to communicate with the cloud server securely.

PHySS units are usually configured with dynamic private internet addresses, which are not directly accessible from the internet. To enable access by remote users, a PHySS opens an SSH session to the cloud server when it starts up, and configures several tunnels that can be used for communication, by different programs. To access a particular PHySS, a user opens another SSH session to the cloud server with tunnels that are configured to pass data to and from the PHySS tunnels. This is illustrated in Figure 5.

This all sounds very complicated, but with a little bit of advance setup, the actual usage is pretty straightforward. The details appear in the following two subsections.

## 3.1 Instructions for Mac or Unix/Linux users

These instructions assume that the application OpenSSH is already installed. This will normally be the case for Macs and Unix/Linux computers. The instructions also assume that the user is affiliated with Mote Marine Lab and that PHySS units with serial numbers 31, 32 and 33 are installed and operating.

The first step is that the user must be authorized to login to the cloud server. This requires having a public/private key pair, with the public key saved on the server. To create a public/private key pair, type the command "ssh-keygen" in a terminal window. You will be prompted several times for input; simply hit return to accept the defaults. This generates two files, id_rsa and id_rsa.pub, which are stored in the ~/.ssh directory. Send the id_rsa.pub file to the system administrator for the cloud server, who will include it in the list of authorized keys used for login authentication. Do not share your private key with anyone.

The next step is to create a file ~/.ssh/config containing the following text.

```
Host coolcloud
        HostName coolcloud.mote.org
        User moteUser
        Port 28102

        LocalForward 28310 localhost:28310
        LocalForward 28311 localhost:28311
```

```
LocalForward 28320 localhost:28320
LocalForward 28321 localhost:28321

LocalForward 28330 localhost:28330
LocalForward 28331 localhost:28331
```

With this file in place, you should be able to type "ssh coolcloud" in a terminal window to open an SSH session to the cloud server, *coolcloud.mote.org*. You will be logged into the shared user account *moteUser* and you should not be prompted for a password, since your public key is used for authorization.

You can now access the PHySS with serial number 31, by typing "localhost:28310" in the url window of your web browser. This should bring up the web control console in your browser window. To access the PHySS with serial number 32, enter "localhost:28320," for serial number 33 enter "localhost:28330." The numbers that follow the colon in each of these examples are internet *port numbers*. The PHySS units use port numbers of the form 28SSX where SS is the serial number of a specific PHySS unit and X designates a particular program on that PHySS. You can examine the current state of the PHySS without logging into it. If you need to be able to take control of the PHySS, ask the system adminstrator for the PHySS to provide you with a PHySS login name and password.

You can also access the analysis console running on PHySS 31 by typing "localhost:28311" in the url window of your browser (use "localhost:28321" or "localhost:28321" to access PHySS 32 or 33). Note, this will only give you access to the most recent data. You will not be able to observe past deployments, or the data for other PHySS units. To do that, use the data console to access the data stored on coolcloud.

In the SSH configuration file shown above, there are two lines for each of the PHySS units used in this example. If you need to communicate with another PHySS unit as well, simply make a copy of the last two lines in the file and change the digits designating the serial number to match the serial number of the new unit.

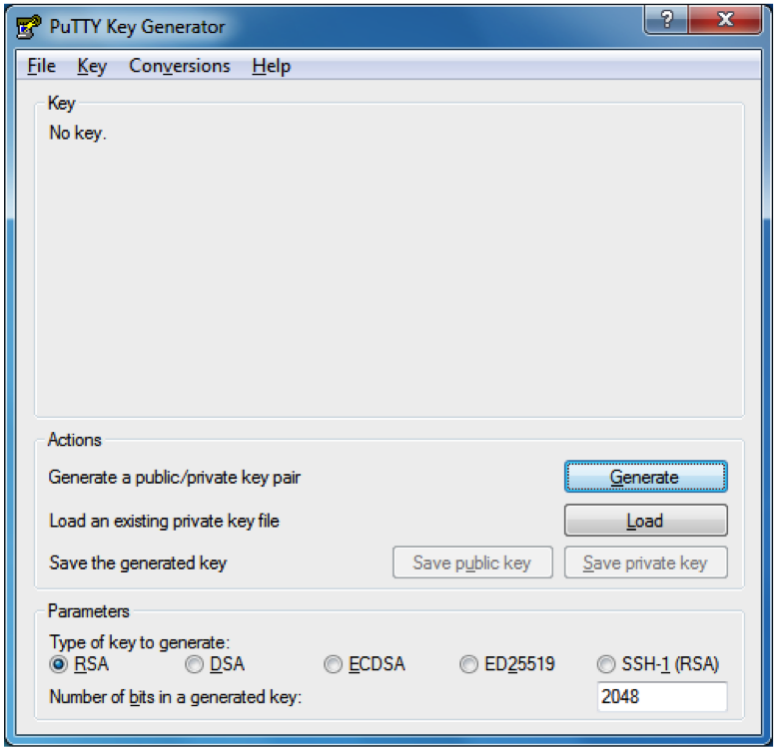


**Figure 6 PuttyGen application**

## 3.2 Instructions for Windows users

OpenSSH can be run on recent versions of Windows (version 10 and later). If your computer has OpenSSH installed, you can use the instructions in the previous section. For earlier versions of Windows (or as an alternative) you may use *putty*, a free SSH application for Windows. If putty is not already installed on your computer, it can be downloaded from `www.putty.org`.

The following instructions assume that putty is installed, that the user is affiliated with Mote Marine Lab and that PHySS units with serial numbers 31, 32 and 33 are installed and operating.

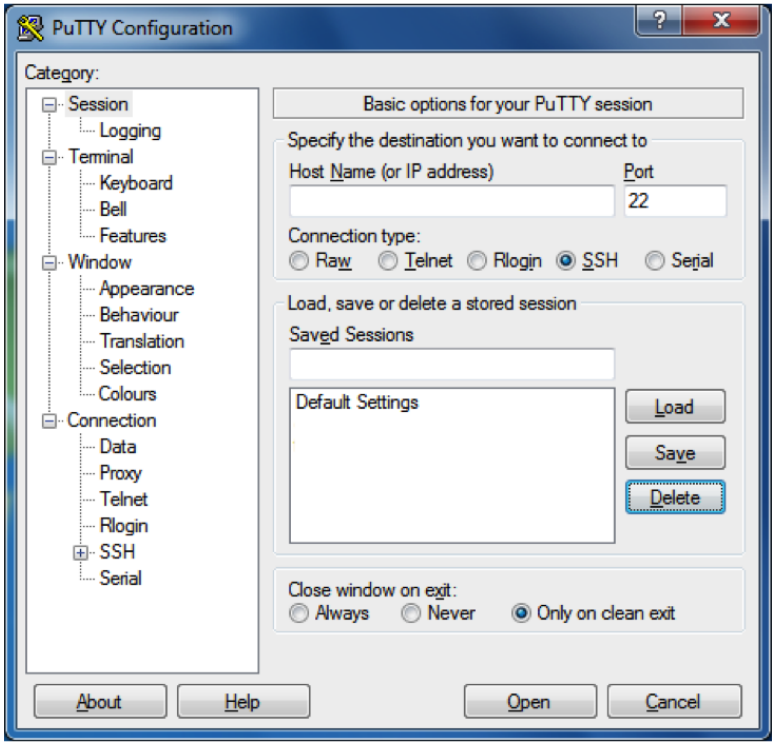


**Figure 7 Putty application**

The first step is that the user must be authorized to login to the cloud server. This requires having a public/private key pair, with the public key saved on the server. With putty, this is done using an associated application called *puttyGen* that is included with the putty package. Figure 6 shows a screenshot of puttyGen. To generate a key pair, simply click on the *Generate* button. You will be instructed to move your mouse back and forth in the window in order to generate some random input for the key generation process. When the key pair has been generated, save the public key and the private key in some suitable location on your computer (a good choice is a sub-folder named putty within your My Documents folder). Then email the

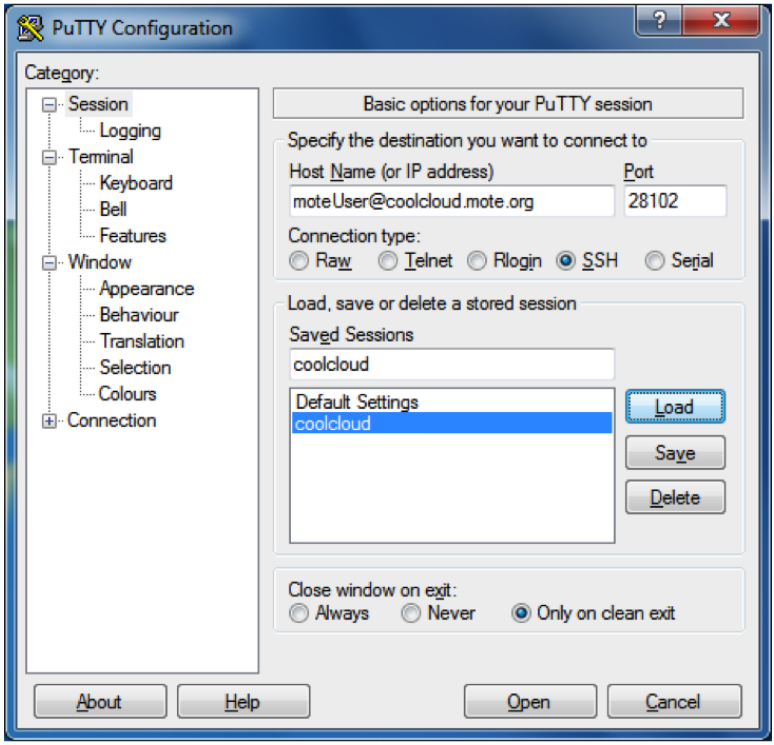


**Figure 8 Coolcloud session profile**

public key to the system administrator for the cloud server, who will include it in the list of authorized keys used for login. Do not share your private key with anyone.

The next step is to configure putty to enable you to easily login to the coolcloud server. Figure 7 shows the putty application as it first comes up. This displays the *Session information*. To create a session profile for logging into coolcloud, type "moteUser@coolcloud.mote.org" into the *Host Name* text box, "28102" into the *Port* text box and "coolcloud" into the *Saved Sessions* box, then click on *Save*. Your application window should now look like Figure 8

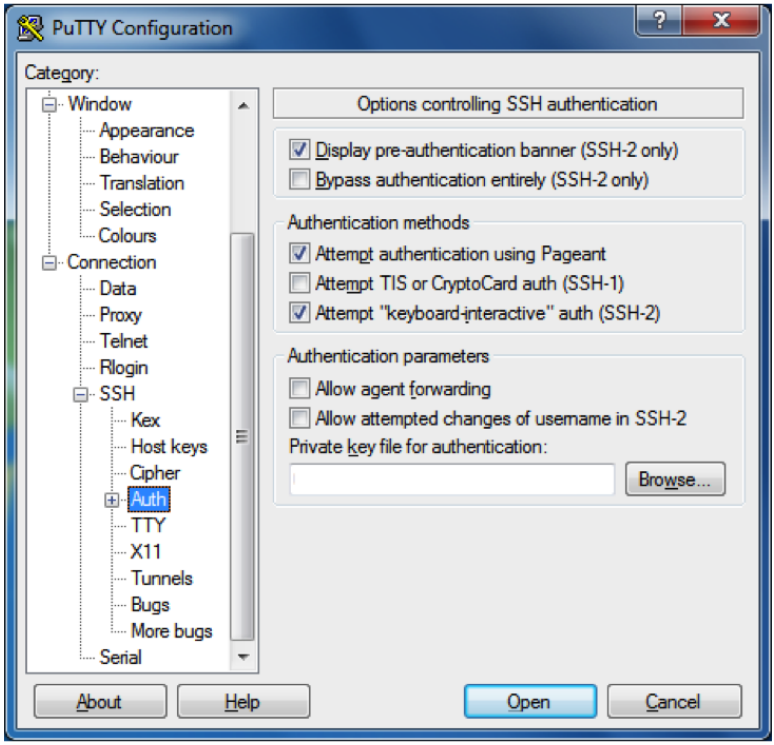


**Figure 9 Putty SSH authorization window**

To complete the new session profile, there are two more steps. The first is to configure putty so that it knows the location of the file containing your private key. To do this, in the sub-window on the left-side of the main putty window, click on the plus sign next to the item labeled "Connection". This will open up an additional list of menu items, including one labeled "SSH." Select the plus sign next to this item to expand the menu again. Now, click on the item labeled "Auth." The putty window should now look like Figure 9. Use the *Browse* button to locate the file containing your private key and select it.

The last step is to configure the SSH tunnels. Select the menu item labeled

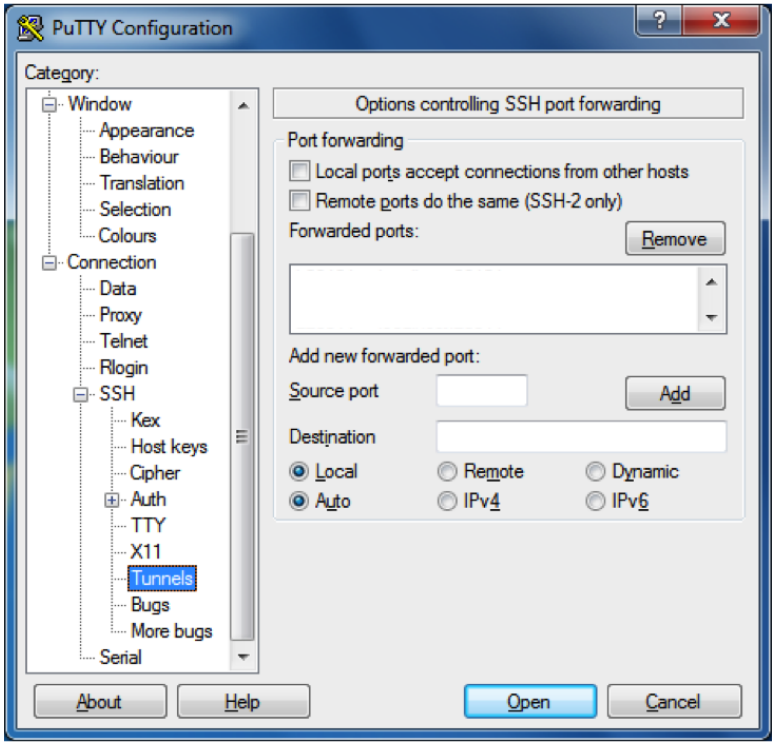


**Figure 10 Putty SSH tunnels window**

"Tunnels." The putty window should now look like Figure 10. Type "28310" into the textbox labelled "Source Port" and "localhost:28310" in the textbox labeled "Destination," then click on the *Add* button. This information should now appear in the larger text area, just above the "Source Port" textbox. Repeat the above steps several more times, with the following values.

```
Source Port        Destination
   28311         localhost:28311
   28320         localhost:28320
   28321         localhost:28321
```

```
28330        localhost:28330
28331        localhost:28331
```

If you make a mistake, you can remove any item in the list using the *Remove* button and repeat the process. Note that the order in which the tunnel specifications appear does not matter.

Finally, review the list of tunnels to verify that it is correct, then go back to the Session view (Figure 8) and click on *Save* to ensure that your SSH configuration is added to the coolcloud session profile. At this point, you can open an SSH connection to coolcloud by clicking on the *Open* button in the putty window. You should see a terminal window open containing a login message and a prompt. You should not be prompted for a password, as your public key is used for authorization.

You can now access the PHySS with serial number 31, by typing "localhost:28310" in the URL window of your web browser. This should bring up the web control console in your browser window. To access the PHySS with serial number 32, enter "localhost:28320," for serial number 33 enter "localhost:28330." You can examine the current state of the PHySS without logging into it. If you need to be able to take control of the PHySS, ask the system adminstrator for the PHySS to provide you with a PHySS login name and password.

You can also access the data analyzer running on PHySS 31 by typing "localhost:28311" in the URL window of your browser (use "localhost:28321" or "localhost:28321" to access PHySS 32 or 33). Note, this will only give you access to the most recent data on that specific PHySS. You will not be able to observe past deployments, or the data for other PHySS units. To do that, use the data analyzer running on coolcloud.

To extend the configuration to handle another PHySS unit, simply add two more lines to the list of SSH tunnels and save your configuration changes. For example, to extend it to handle a PHySS with serial number 34, add these lines.

```
Source Port       Destination
   28340        localhost:28340
   28341        localhost:28341
```

# 4 Deploying a PHySS Unit

Before deploying a PHySS unit, there are two calibration procedures that must be carried out. First, both the sample pump and the reference pump must be calibrated. Second, the depth sensor must be calibrated.

To calibrate the sample pump in the lab, attach short lengths of tubing to the two inlet/outlet ports and place the ends in a container of water. Run the sample pump at high speed for two minutes to flush any air bubbles that may be present in the internal plumbing and observe which of the two ports has water coming out of it. Call this the outlet. Place the outlet tubing in a small empty graduated cylinder, then click on the sample pump's calibrate button. This should cause the button to turn red and the pump to run at its maximum speed. When the graduated cylinder contains exactly 10 ml of water, click on the calibrate button again. This should stop the pump and the console should display a message reporting what the maximum pump speed is.

The calibration procedure is similar for the reference pump, but you will first need to connect a short length of tubing to the input side of the reference pump and place the other end in a container of water. Before doing the calibration, run the reference pump for a couple minutes at high speed to eliminate any air bubbles.

To calibrate the depth sensor, you must first connect an air pump with a gauge to the input of the depth sensor. With zero pressure applied to the sensor, type "depth 0" in the command line text box at the bottom left of the console window (and hit return). You should see a status message reporting

that the offset has been set. Now apply 30 psi of pressure to the sensor input and type "depth 20.5" in the command line textbox and hit return.

Once the unit is calibrated, the various parameters in the configuration file must be set. This can be done using the control console. Simply connect to the PHySS and login, then display the config file in the text area at bottom right and click on the *edit* button. You can now make any required changes to the config file. Verify that the serial number is correct, then set the deployment label and gps coordinates to match your deployment location.

If you want the system to start sampling automatically after the system starts up, set the *autoRun* variable to 0 or to the number of minutes that the system should wait before starting automated sampling. Similarly, set the *powerSave* and *singleCycle* parameters as required for your deployment. If you choose to start the system manually, be sure to enable data tranfer, so that collected files will be uploaded to the cloud server (if the autoRun feature is turned on, data transfer will be enabled automatically).

In addition to setting the configuration parameters, be sure to check that the fluid level that appears in the console window for the reference reservoir reflects the actual fluid level. If deploying a PHySS with the extended hardware configuration, be sure to set the reagent levels also.

# 5 Monitoring a Deployed PHySS

A deployed PHySS can collect data continuously for 4-6 weeks under ideal operating conditions. The main limitation on its length of deployment is the amount of reference fluid in the supply reservoir. However, it is important to monitor the PHySS on a regular basis during a deployment to make sure that it is continuing to operate normally.

The primary tool for monitoring the PHySS operation is the data analysis console, which provides access to a great deal of operational information.

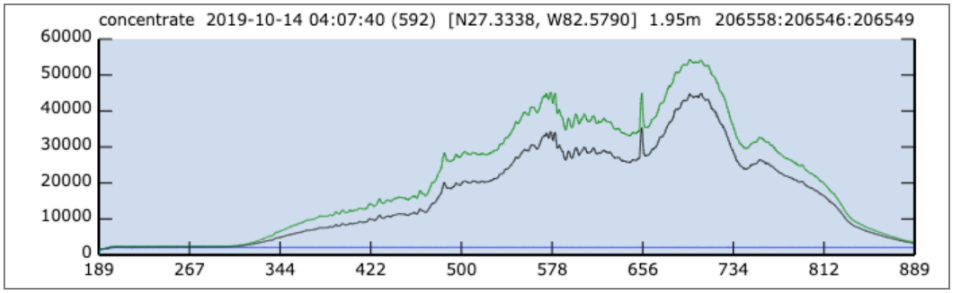


**Figure 11 Raw data spectrum from deployed PHySS**

See [4] for details on how to use the console. Here we briefly discuss some of the ways the analysis console can be used to monitor the health of deployed PHySS units.

The first thing to check on a deployed PHySS unit is the raw data spectrum at the end of the current data file for that unit. An example is shown in Figure 11. First, note the date and time of the displayed spectrum and compare that to the current date and time (keep in mind, that the date/time field uses GMT, not local time). If the last spectrum in the current file is more than a few hours old, it may be that the PHySS has suspended operation or is unable to communicate with the cloud data server. Before concluding that there is a problem, make sure that you are looking at the last spectrum in the current file. You may need to click on the *reload* button to make sure that the local copy in your browser includes the most recent data. You may also need to restart your browser session to ensure that you have the most recent data files. If you conclude that there is a problem, examine the debug messages in the window at the lower right of the analysis console (scroll down to the end to observe the most recent messages). These may give you a clue as to what the problem is.

If the recent data is up-to-date, go ahead and examine the spectra near the end of the file to make sure that they look "reasonable." While there can be some variation from unit-to-unit, most data spectra should look similar

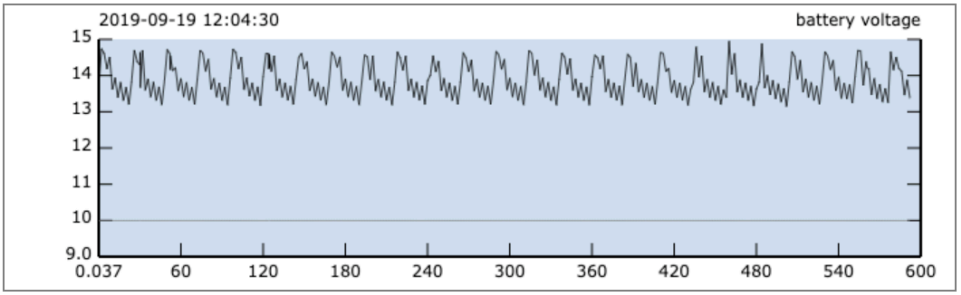


**Figure 12 Chart showing battery voltage**

to the one in Figure 11. If you see something dramatically different, it may indicate a problem. For example, in a recent deployment the shutter on the light source failed to operate properly. Sometimes it would fail to open, other times it would fail to close. This resulted in dark spectra that looked like typical data spectra and vice versa.

The console's standard mode library provides a time-series presentation of several operational variables. Problems with a deployed unit will usually produce unusual values for at least some these variables, so it's worth checking them on a regular basis.

Figure 12 shows a typical plot of the supply voltage, for a PHySS unit that is powered by a battery, which is re-charged by a solar panel. Notice how the 24 hour cycle of charging and discharging affects the voltage. The small alternating variations in voltage are caused by the port switching feature, since the port valves consume more power in the "crossed" state than they do in the "straight-through" state. A problem with the solar charging system (for example, an accumulation of guano on the solar panel) can cause the voltage to drop below the normal range seen in the figure. If the voltage drops below 10 volts, the PHySS will shut down.

Figure 13 shows a typical plot of the maximum pressure across the PHySS filter. With a new filter, the pressure typically stays below 1 PSI. However,

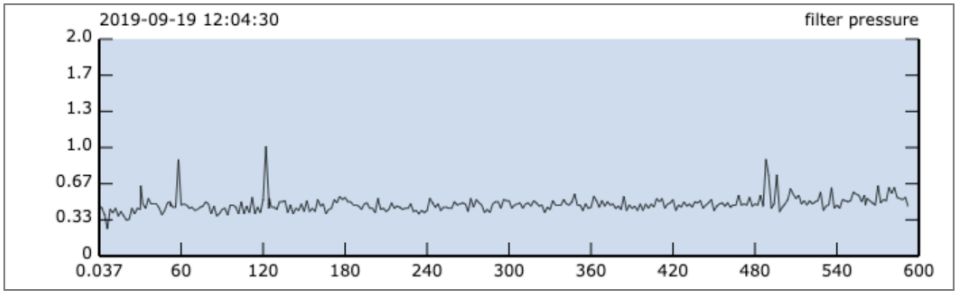


**Figure 13 Chart showing filter pressure**

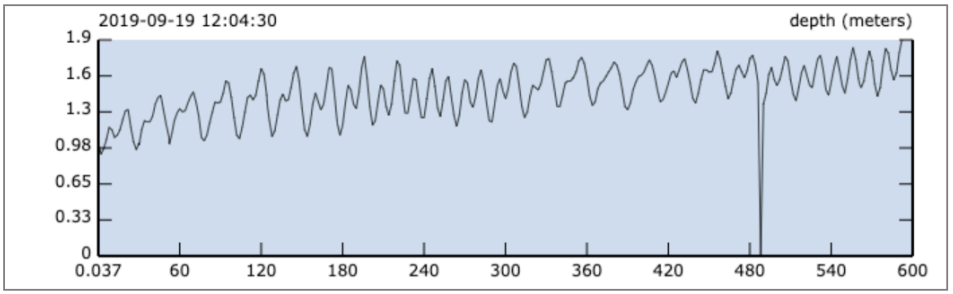


**Figure 14 Chart showing PHySS depth**

larger values may be observed if the pores in the filter become clogged (which happens naturally over time) or if the PHySS script uses higher pumping rates during filtered sample acquisition (in the example shown here, the pumping rate was 2 ml/m). The system can tolerate substantially higher rates (30 PSI is the maximum safe pressure), but any substantial increase observed during a deployment should be interpreted as an indication of a potential problem.

Figure 14 shows a typical plot of the depth of the PHySS in the water, for a unit that was mounted on a piling. Note the effect of the daily tidal variations. Also, observe the zero value shortly after time 480 hours. This is clearly a faulty reading, but appears to be just an isolated case. If it were

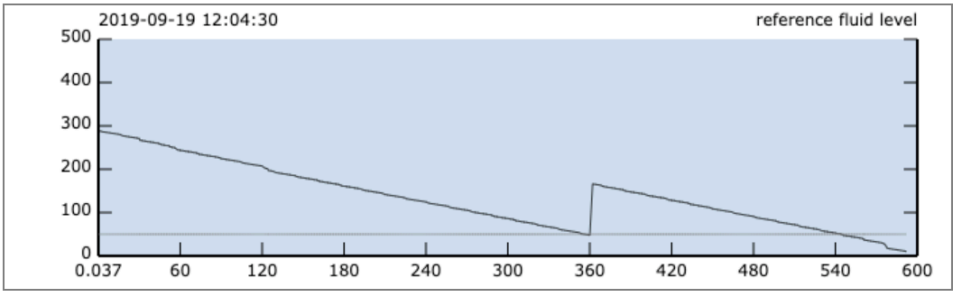


**Figure 15 Chart showing the reference fluid level**

to occur repeatedly it would be a cause for concern. In a recent deployment, one of the mesh screens that cover the inlet/outlet ports developed several holes large enough to allow large particles to get into the internal plumbing. These triggered spurious depth readings on alternate sample cycles, and this allowed the problem to be detected.

Figure 15 shows a plot of the reference fluid level for a deployed PHySS. When the level drops below 10 ml, the PHySS automatically suspends automated sampling. In this case, the reference fluid level was adjusted at about 360 hours, after the unit was pulled for maintenance and was observed to have more fluid remaining than was recorded by the PHySS. This is a fairly common occurrence, since the PHySS estimates how much fluid remains based on how much has been pumped. Because the pumping rates are not always accurate, poor estimates can lead to significant errors in the reference fluid level.

Figure 16 shows a plot of the spectrometer integration time. Note that the integration time has increased from about 31 ms to 40 ms over the 600 hours of this deployment. This is most likely due to the gradual accumulation of material on the inside of the spectrometer waveguide, and is not a particular cause for concern. However large sudden changes could indicate a problem.

Figure 17 shows a plot of the temperature inside the PHySS enclosure. This should typically stay in a narrow range between 30 and 35 degree Celsius.

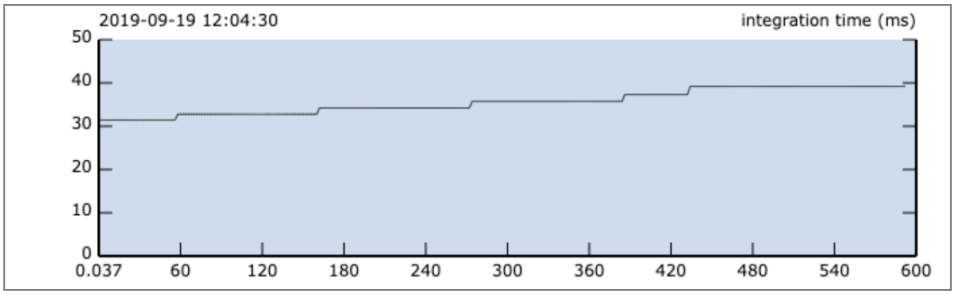


**Figure 16 Chart showing the spectrometer integration time**

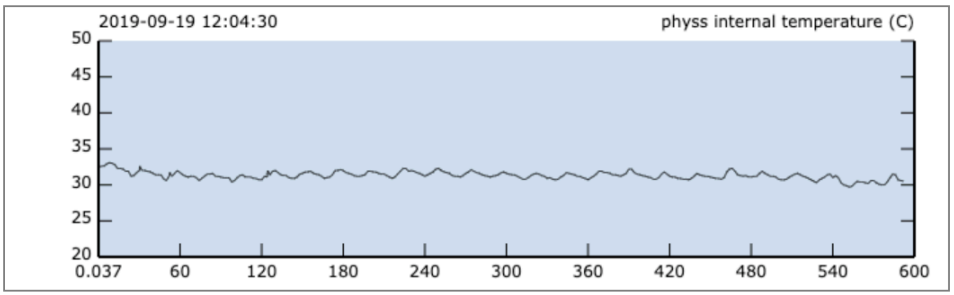


**Figure 17 Chart showing the PHySS temperature**

Substantially higher temperatures indicate that the system is overheating, most likely due to a failure in one of the electronic components. If the temperature exceeds 60 degrees, the system will shut down automatically, however any signfication deviation from the nominal range indicates a potential problem.

# 6 Trouble-Shooting

If you suspect that there is a problem with a deployed PHySS, the next step is to login to the unit and try to get a more complete understanding of the

problem. If the PHySS is operating in powerSave mode, you will need to wait for the start of the next sample cycle. So for example, if the current time is 13:00 GMT and the PHySS is operating on a 2 hour cycle, you'll need to wait until just after 14:00. To login, you will need to first open an SSH connection to coolcloud, as discussed in section 3. Then, enter localhost:28sn0 in your browser's URL window, where sn is the serial number of the unit you want to connect to. It does take the PHySS a couple minutes to startup and establish a connection to coolcloud, so do not expect to get connected right on the hour.

Once you have connected and logged into the PHySS, hit the *stop* button to suspend the automated sampling process. Then review the messages in the lower left panel to see if there if there are any anomalies. While the messages you see will depend on the details of the script you are using, there are some typical issues that you can look for. To illustrate, an excerpt from a typical sample cycle appears below.

```
starting cycle 76 at 2019-11-09 22:02:15 [INFO 271073.142]
starting filtered sample phase (aka cdom)
filteredSample(12.00ml, 3.00ml/m, 0.000, 0.000)
[DETAILS 271081.219]
getSpectrum(6) [DETAILS 271321.241]
spectrum: max=2253.4, maxWave=729.4, intensity440=2150.8
[DETAILS 271325.755]
getSpectrum(7) [DETAILS 271325.800]
spectrum: max=55463.7, maxWave=485.5, intensity440=38360.8
[DETAILS 271330.316]
```

The numerical parameter in the two `getSpectrum` operations is interpreted as three bits specifying the on/off state of the two lights and the shutter. So `getSpectrum(6)` specifies that the deuterium and tungsten lights are both on and the shutter is closed. This is used for obtaining a dark spectrum. The line following the *getSpectrum* operation reports the maximum light level

observed by the spectrometer, the wavelength at which that maximum light intensity is observed and the intensity at 440 nm. For dark spectra the light levels will typically be in the range of 2000 to 3000. If substantially larger values are observed, it implies that the shutter did not close as specified. The second *getSpectrum* operation shows a max light intensity of 55463.7, which is typical for a data spectrum obtained with the shutter open. If the value is substantially lower, it may indicate that one or both light sources failed to turn on, or that the shutter failed to open.

If you cannot identify the problem by simply reviewing the messages, you may need to run a few sample cycles manually while displaying more detailed log messages (using the log level menu). To do this, first turn off the powerSave mode in the config file and set the time between sample cycles to a small value (say 5-10 minutes). Then hit resume and observe the system as it runs through one or more sample cycles.

Sometimes when you attempt to connect to a PHySS, the *operations server* component will respond, and display the console screen, but the *data collector* component does not respond to commands. This typically means that the data collector has terminated for some reason. By selecting the "collector.stderr" item from the file menu, you can see additional error messages printed by the console, which may help you determine why the collector terminated.

If you are unable to connect to the system at all, the next step is to cycle power to it. In most deployments, a separate remotely operated switch is used to turn the power to the PHySS off and on. This switch can be activated by sending a text message. If this process succeeds, you should be able to connect to the PHySS within a couple minutes.

In situations where you cannot communicate with a PHySS using the control console, you may be able to communicate with it using SSH. This method requires some familiarity with the linux command line interface, but can be useful in some situations. To use this method, you will need to add

an additional SSH tunnel specification to your SSH configuration. If your local computer is a Mac or a unix/linux machine, do this by adding the following line to your SSH config file for each PHySS you want to be able to communicate with.

```
LocalForward 28sn2 localhost:28sn2
```

where "sn" is the serial number of the unit you need to communicate with. Note that the last digit of the port number must be 2. Once you have added this to your config file, you should be able to login to the the PHySS's main processor by typing

```
ssh -p 28sn2 guru@localhost
```

in a shell window. You will need the password for the *guru* account on the Beaglebone, in order to login. The system administrator can provide that to you, or can add your public key to the list of authorized keys for the account.

Similarly, if you are using putty, add the following to the list of tunnels in your coolcloud configuration and save it.

```
Source Port        Destination
   28sn2         localhost:28sn2
```

Now, open a connection to coolcloud using this configuration. Once this connection is open, click on the putty icon in the upper left corner of the window and select New Session. This will cause a new putty control window to open. Enter localhost into the Host Name text box and 28sn2 in the Port text box, then click the Open button. This will create a connection to the PHySS that passes through *coolcloud* using the configured SSH tunnels. You should see a linux login prompt in your putty connection window.

Once you have logged into the Beaglebone on the PHySS, you can determine if the data collector is running by typing

```
systemctl status collector
```

Similarly, you can determine if the operations server is running by typing

```
systemctl status opsServer
```

To start the operations server, type

```
systemctl start opsServer
```

Once this is running, you should be able to login using the control console and examine the error messages in the debug file and collector.stderr. Before starting the collector, you may want to first disable the *autoRun* mode, so that the collector does not start automated data collection when it starts up. If you don't do this, the collector may run into the same problem that caused it to terminate, leading it to terminate again. You should be able to disable *autoRun* by editing the config file using the control console. To start the data collector, type

```
systemctl start collector
```

# 7 Summary

# References


[1] Hails, A., C. Boyes, A. Boyes, R. D. Currier, K. Henderson, A. Kotlewski and G.J. Kirkpatrick "The Optical Phytoplankton Discriminator," *Proceedings of Oceans*, 2009. 2

[2] Kirkpatrick, Gary J., David F. Millie, Mark A. Moline, Steven E. Lohrenz and Oscar M. Schofield. "Automated, in-water determination of colored dissolved organic material and phytoplankton community structure using the optical phytoplankton discriminator," In *Proceedings SPIE 8030, Ocean Sensing and Monitoring III*, May 04, 2011. 2

[3] Molloy, Derek. *Exploring Beaglebone: Tools and Techniques for Building with Embedded Linux*, Wiley, 2014. 4

[4] Turner, Jonathan. "Using the PHYSS Data Analysis Console," Mote Marine Research Laboratory, Ocean Technology Group, 8/2019. 29

# Appendix

## A Operations Console Commands

This section lists all the commands that can be issued from the operations console. All commands are text strings that are sent from the console to the operations server on the PHySS. Some commands are carried out by the server directly, while others are forwarded to the data collector. These are listed separately in the following subsections.

Some commands are *restricted* and are only accepted if issued from the control console of the user who is currently logged in (called the *privileged user*). There is never more than one privileged user for a particular PHySS unit. The operations server implements the login mechanism and accepts restricted commands only from the privileged user. Commands that are purely informational are not restricted.

### A.1 Commands handled by operations server

The commands in this section are carried out by the operations server, in response to commands sent by the operations console in response to user interactions with the graphical user iterface.

- *login* user_name password. If no other user is currently logged in, and the user name and password are valid, the server responds with a "session accepted" reply containing a *session code*, which the console saves for inclusion in future command messages. This allows the server to distinguish commands sent by the privileged user from those sent by other users. If there is already a privileged user when a login message is received, the server responds with a "session in progress" reply, identifying the privileged user and the time remaining before that user will be automatically logged out (assuming no further activity on their part).

- *logout*. The logout command causes the privileged user to be logged out. The server responds with a "session terminated" reply. (restricted)

- *newpass* new_password. The new password replaces the password for the privileged user and a "new password accepted" reply is sent. (restricted)

- *read* file_name. The file name argument, may be any of the following:

```
collector.config
collector.script
collector.state
collector.debug
collector.stderr
opsConsole.stderr
```

- *write* file_name. The file name may be either `collector.config` or `collector.script`. (restricted)

- *clear*. Clears the contents of the files `collector.debug`, `collector.rawData`, `collector.stderr` and `opsConsole.stderr`. The normal reply is "cleared debug, rawData and stderr files". (restricted).

- *enableXfer*. Enables the transfer of the raw data file to the cloud server. The normal reply is "data transfer enabled." (restricted)

- *disableXfer*. Disables the transfer of raw data to the cloud server. The normal reply is "data transfer disabled." (restricted)

- *getXferStatus*. Used to retrieve the current on/off status of the data transfer mechanism. The normal reply is "xferStatus n" where $n = 0$ means "off", $n = 2$ means "on" and $n = 1$ means that the status is in transition (meaning that the operations server has changed the status but the data transfer sender has not yet acknowledged the change).

## A.2 Commands handled by the data collector

The commands listed below are relayed by the operations server to the data collector. Replies are similarly forwarded by the server back to the operations console. Most of these commands are issued in response to user interactions with the graphical user interface. They can also be entered manually into the operations console using the command text box at the bottom left of the console window. Each command consists of a command name, followed (optionally) by one or fields, with fields separated by spaces.

- *start*. Start execution of sample collection script. This is valid only when the data collector is first starting up. (restricted)
- *stop*. Suspend execution of sample collection script. This is valid only when the script is running. (restricted)
- *resume*. Resume sample collection from the start of the current sample cycle. This is valid only when the script is suspeded. (restricted)
- *restart*. Restart sample collection, setting sample cycle 1 number back to 1. (restricted)
- *quit*. Terminate data collection and shut down PHySS. (restricted)
- *samplePump* [ ((on rate) | off) | calibrateStart | calibrateFinish ]
  This command is used to turn the sample pump on off, change its rate and calibrate it. The rate parameter is specified in milliliters per minute. If no arguments are provided, the reply contains the current pump rate and the pump's maximum rate. After a *calibrateStart*, the only acceptable next command is a matching *calibrateFinish*.
  (restricted)
- *referencePump* [ ((on rate) | off) | calibrateStart | calibrateFinish ]
  Similar to *samplePump* command. (restricted)

- *reagent1Pump* [ ((on rate) | off) | calibrateStart | calibrateFinish ] Similar to *samplePump* command. (restricted)

- *reagent2Pump* [ ((on rate) | off) | calibrateStart | calibrateFinish ] Similar to *samplePump* command. (restricted)

- *referenceSupply* [ volume ]. If no argument is given the reply includes the current estimate for the amount of fluid in the reference supply reservoir. If the volume argument is specified (in milliliters), the specified value replaces the current estimate for the amount of fluid in the reference supply reservoir. (restricted).

- *reagent1Supply* [ volume ]. Similar to *referenceSupply* command. (restricted)

- *reagent2Supply* [ volume ]. Similar to *referenceSupply* command. (restricted)

- *filterValve* [ 0 | 1 ]. If no argument is included, the current valve state is returned in the reply. If an argument of 0 is included the valve state is changed to 0 (seawater is unfiltered). If an argument of 1 is included, the valve state is changed to 1. (restricted)

- *portValve* [0|1]. Similar to *filterValve* command. 0 specifies that the port valve is in the "straight-through" configuration, 1 specifies the "crossed" configuration. (restricted)

- *mixValves* [00|01|10|11] Similar to *filterValve* command. 11 specifies that the valves are to be configured so that both reagents can be mixed with the seawater sample, 00 specifies that neither reagent is to be mixed with the seawater sample, 10 specifies that reagent 1 should be mixed with the sample and 01 specifies that reagent 2 should be mnixed with the sample. (restricted)

- *lights* [bbb]. If no argument is included, the state of the lights is returned in the reply. The argument consists of three "bits". The first specifies the state of the deuterium light sources, the second specifies

the state of the tungsten light source and the third specifies the state of the shutter (restricted).

- *integrationTime* [time] If no argument is included, the integration time value is returned. If an argument is included, the specified integration time replaces the value currently configured for the spectrometer. (restricted)

- *spectrum* [bbb]. If no argument is included, a spectrum is acquired using the current configuration of the light source and light intensity values at 20 equally spaced wavelengths are returned. If an argument is given the three bits specify the light source configuration, as in the *lights* command. (restricted)

- *power* [bbb|on|off]. If no argument is specified, the PHySS power state is returned. If the argument is a three bit value, it specifies the desired power state. A argument of "on" is equivalent to 111, an argument of "off" is equivalent to 000. (restricted)

- *pressure*. Returns the pressure across the filter.

- *serialNumber*. Returns the serial number of the PHySS.

- *versionNumber*. Returns software version number

- *logLevel* [levelName]. If no argument is provided, returns the current logging level. If a valid log level argument is provided (TRACE,, DEBUG, DETAILS, INFO, WARNING, ERROR, FATAL), the log level is changed to the specified value. (restricted)

- *cycleNumber*. Returns the number of the current sample cycle.

- *reload* (script|config). If the argument is "script" then the script file is reloaded. If the argument is "config" then the config file is reloaded. (restricted)

- *check4faults* (0—1). If the argument is 0, fault checking by the Arduino is turned off. If the argument is 1, fault checking by the Arduino is

turned on. When fault checking is turned on, the Arduino checks for faulty conditions and shuts down they system automatically if a fatal fault is detected.

# B Script Statements

to be added

# C Configuration Variables

to be added

# D State File

The state file contains copies of data collector variables that must be retained across multiple executions of the data collector program. When the data collector is started, the file is read and used to initialize the data collector's internal variables. Whenever any of the internal variables is modified, a new copy of the state file is written out.

Here is an example of a state file with explanatory comments added.

```
cycleNumber = 11 # current cycle number

# max pump rate parameters, set by calibration procedure
samplePumpMaxRate = 6.25
referencePumpMaxRate = 7.14
```

```
# supply reservoir levels, modified by operator using gui, update
referenceSupplyLevel = 398

# depth sensor parameters, set by  calibration procedure
depthSensorOffset = 0
depthSensorScale = .0088

# spectrometer parameters, set by optimizeIntegrationTime script
integrationTime = 164

# dataStore state
currentIndex = 14523 # record index of next record to be output
deploymentIndex = 1341 # record index of most recent deployment r
recordMap = dark.14501 filtered.14506 concentrate.14513 unfiltere
# record indices of most recent spectra with specified labels
```

# E Raw Data File Format