Supplementary material: Greenhouse gas flux studies: An automated online system for gas emission measurements in aquatic environments

Nguyen Thanh Duc[1,4], Samuel Silverstein[2], Martin Wik[3], Patrick Crill[3], David Bastviken[4], Ruth K. Varner[1]

[1]Institute for the Study of Earth, Oceans and Space and Department of Earth Sciences, University of New Hampshire, Durham, 03824, USA
[2]Department of Physics, Stockholm University, 106 91, Sweden
[3]Department of Geological Sciences, Stockholm University, 106 91, Sweden
[4]Department of Thematic Studies - Environmental Change, Linköping University, 581 83, Sweden

Correspondence to: Nguyen Thanh Duc (thanh.duc.nguyen@liu.se)

This supporting information contains twenty two pages, two tables, and twelve figures

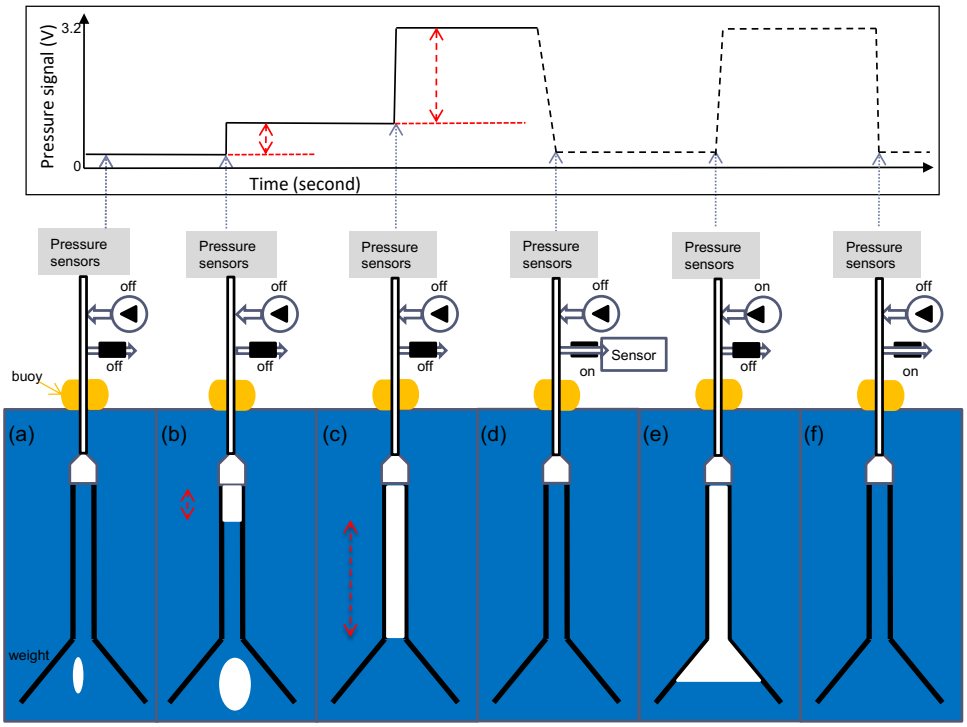

Figure S1: Simplified illustration of the submerged automatic bubble counter (ABC) cycle including (a) initiation, (b) trapping a small bubble, (c) trapping a big bubble, (d) releasing trapped gases into sensor chamber or sample vial array (not drawn here), (e) pumping air into sampler to refresh the system, (f) venting the system. The dashed lines illustrate pressure levels that are not related to new ebullition events. Note that the figure is not to scale. See the text for details.

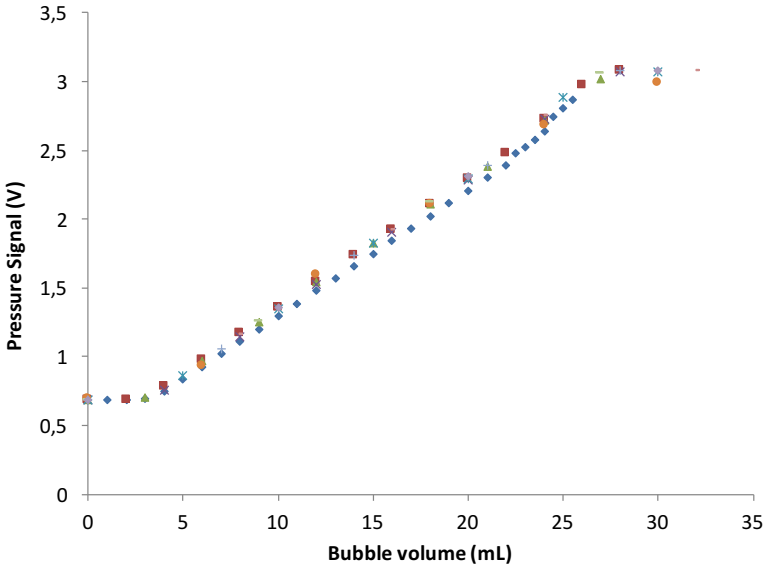

Figure S2: Pressure signal (V) response to bubble volume (mL). Bubbles were injected stepwise from 0.5 to 10 mL (presented in different symbols) under the bubble trap by 1, 5 and 10 mL syringes to generate the appropriate volumes. The test shows the linear response range of pressure sensor at bubble volumes from 5 to 28 mL.

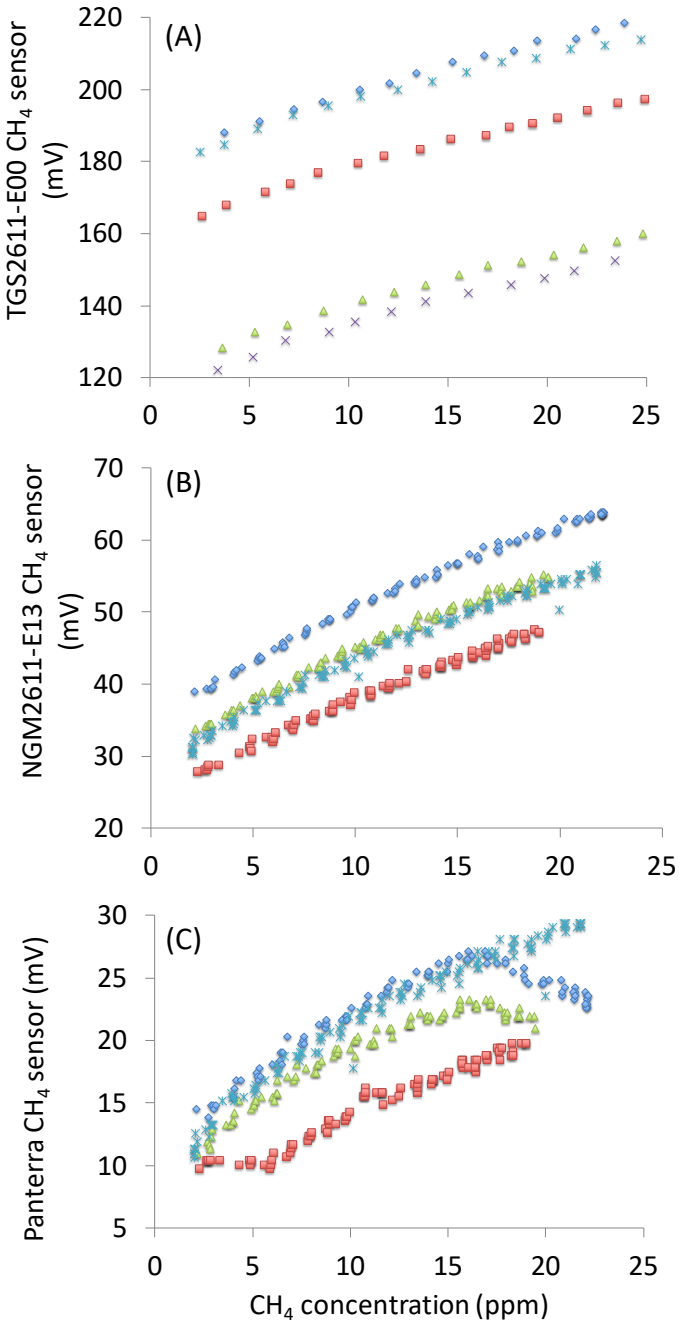

Figure S3: Response of three CH_4 sensors (a. TGS2611-E00, b. NGM2611-E13 and c. Panterra) to changing CH_4 concentrations and at different water temperatures: 10°C (squares), 15°C (asterisks), 20°C (triangles), and 30°C (diamonds).

5

S.1 Overview of AFC_ABC

The AFC_ABC system is controlled by a microcontroller-based datalogger board. The datalogger is pre-programmed with many functions to allow the AFC_ABC work as an independence system or semi-independence in a wireless network with mesh topology. These functions are called by setting an on-board DIP switch address. When the datalogger is powered on, its DIP switch address is read, then an assigned function will be activated as the following list:

Table S1: List of datalogger functions.

DIP switch address	Function	Describe
255	Standalone AFC_ABC	The AFC_ABC is configured from files in the SD card and data is logged on the SD card as in Duc et al (2013).
0	Controller	In a mesh network, this datalogger is used in the Host_controller on the lake shore. It sends request to clients and receive information from clients
1 - 10	Data relay driver	It is used in case that the line of sight from controller to clients are block. This is deployed to keep the topography of digimesh network.
11 - 50	AFC equipped discontinue collect gas samples _ABC	This mode is used when the accumulation period is shorter than 12 hours. A pair of samples including initial and end which are collected at the beginning and end of an accumulation period. A gap times between these collected pair samples can be set.
51 - 90	No sink AFC _ ABC	In ventilation step of this mode, air is blown into the chamber as in Figure S9.
91 - 126	AFC equipped discontinue collect gas samples _ABC equipped sensor box	In this mode, the accumulated gas in the bubble trap is vented into a sensor box include high range CH4 and CO2 sensors. The gas was hold in sensor box for 5 minutes to measure CH4, CO2 gas composition.

127 - 139	Test AFC sampling_ together with wireless communication	The AFC go through accumulation (1 minute), ventilation (5 minutes), closing (3 minutes), collect gas in 15 seconds into 9 vials.
140 – 179	AFC equipped continue collect gas samples _ABC	This sampling is as in description in Duc et al (2012)
180 - 192	Test no sink AFC_ together with wireless communication	The AFC go through accumulation (1 minute), ventilation (5 minutes), closing (3 minutes) to quickly test the open/close mechanism.
254	Test I2C communication with CO2 sensors	Test the CO2 sensors
253	Test switch mode	Test electronic switch of the power control board.
252	Test no sink AFC_ no network communication	As test setting with address 180 -192, but no host-controller needs
251	Test AFC sampling_ no network communication	As test setting with address 127 -139, but no host-controller needs.

In the program control AFC_ABC, the datalogger sends trigger signals to turn on/off electronic switches which control pump and valves. When pump, valves of either AFC or ABC does not connect, the datalogger still sends the trigger signal as being programmed, but no power is consumed. Therefore, users are free to choose either AFC or ABC or both. Beside control function, this datalogger have five analog signal inputs (for pressure sensor, Figaro CH4 sensors, temperature sensor LM35, and any type of analog sensor with Voltage output (< 3.3V) that operates on 5 or 10V, max operate current 500mA); and two I2C digital signal inputs for two CO2 sensors. These sensor sockets are independent, user can free to plug in the sensors or not. For the analog sensor, when there is no analog sensor in used, a jumper connecting ground line to data input needs to be pluged to send 0V to datalogger. For digital sensor, the datalogger will check if there are any CO2 sensor connection at the start up; if not, the datalogger will not send any measurement request in later. The analog sensor data can be read up to 5 data point per second, the data only log into the SD card or send to the host-controller in wireless network at the frequency set by user. The digital signal acquisition rate depends on the sensor working protocol. For the CO2 sensor in this study, the datalogger send the request to CO2 sensor to do measurement, then the datalogger send the request to CO2 sensor to send the measurement results of CO2 concentration, temperature and relative humidity. The highest measurement frequency of this CO2 sensor is every 20 seconds, but we observed a drift in CO2 concentration at this frequency over few hours measurement. In our wireless

network application, data acquisition rate is set one measurement per minute, this helps to maintain the communication protocol in this digimesh network (described in Wireless data transfer section).

The main electronic components of the AFC_ABC composes of one power control board, one datalogger board with DIP address set any value higher than 0, two diaphragm pumps, four of 3-way valves, one CH_4 sensor, one CO_2 sensor, one pressure sensor, an SD memory card for offline setup; and additional of one datalogger board with DIP address set to 0, one raspberry pi computer and two XBee pro S1 modules for wireless digimesh network. (The hardware design will be post or share via email for academic use after the manuscript accepted).

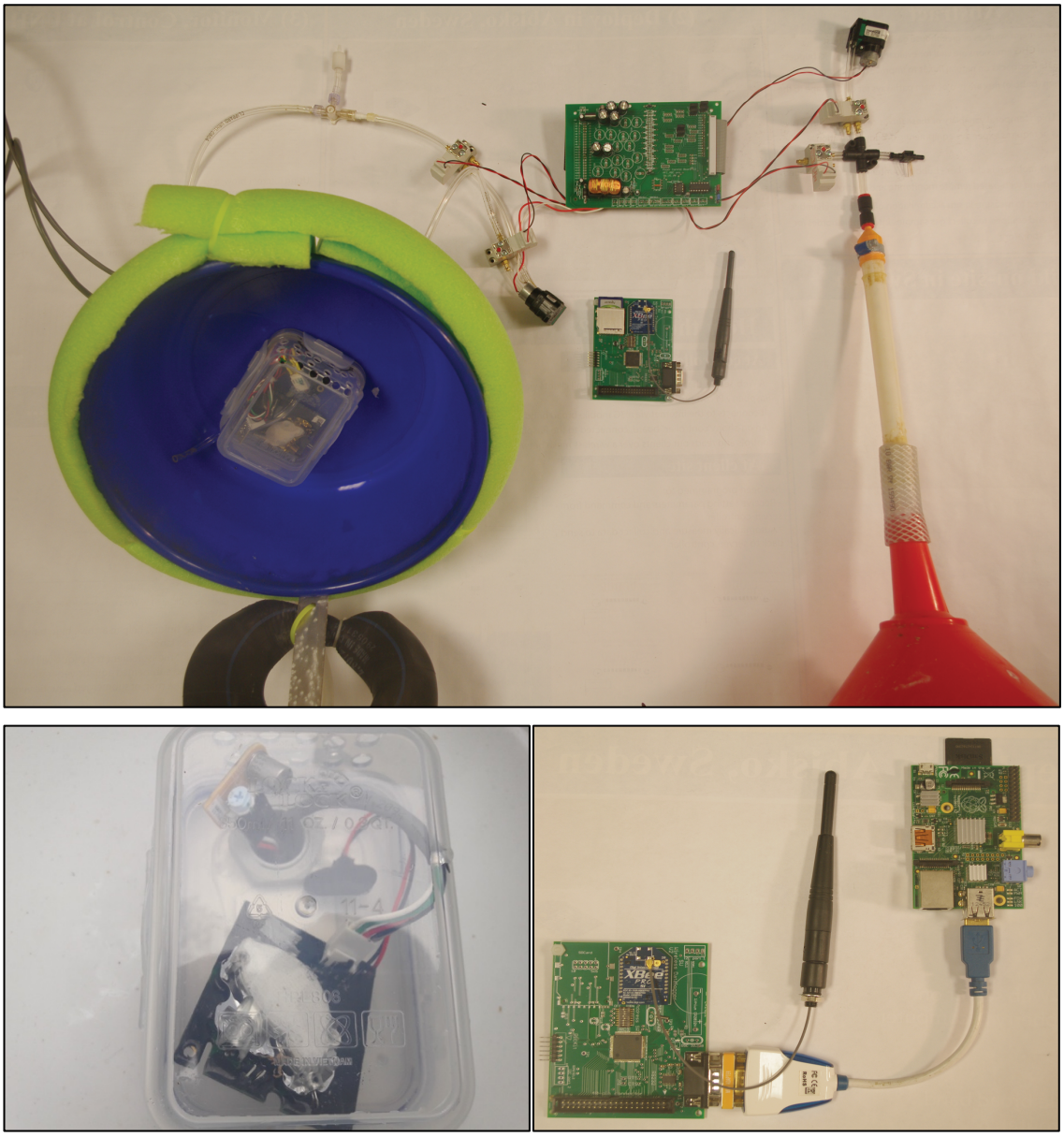

Figure S4: Complete electronic component set of AFC_ABC system. User can choose to operate either AFC or ABC or both and can either setup a wireless network or use it as a standalone unit with SD card.

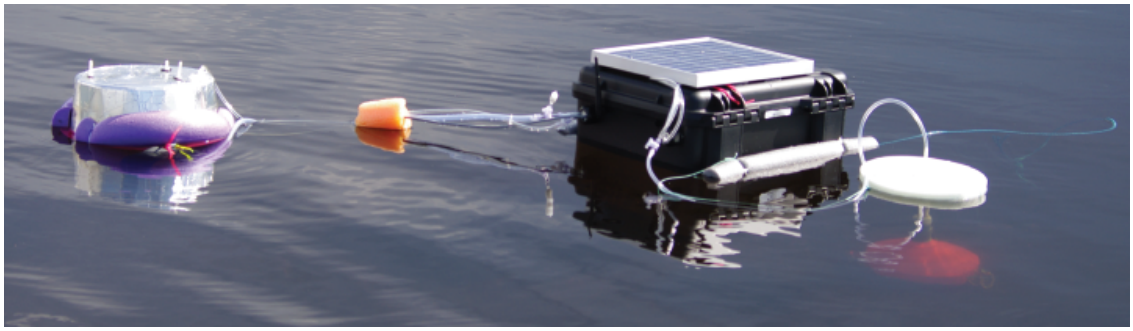

Figure S5: Example of AFC_ABC deployment in the lake.

S.1.1 Sensor coating

In our lab, we used two syringes (e.g. 5 mL and 1 mL syringes) to take the base and the hardener in ratio 3.8 : 1 pbv. 1 mL hardener was transferred to the 5 mL syringe containing the base using a 3-way stopcock. Then 5 mL syringe was closed and vortex for about 3 minutes to well mix these components. The mixture can be tested if they well mixed or not by connect and transfer the mixture from this 5 mL syringe to another empty 5 mL syringe. If the mixture is well mixed, there is some pressure or resistant when the syringe is pushed. An 18 GA needle is recommended to use for spreading the coating mixture on the sensor and around the connector.

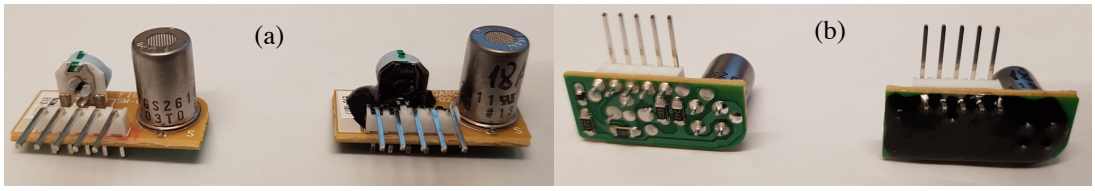

Figure S6: (a) Top view of CH4 sensor (left: non-coating, right: coating); (b) bottom view of CH4 sensor (left: non-coating, right: coating).

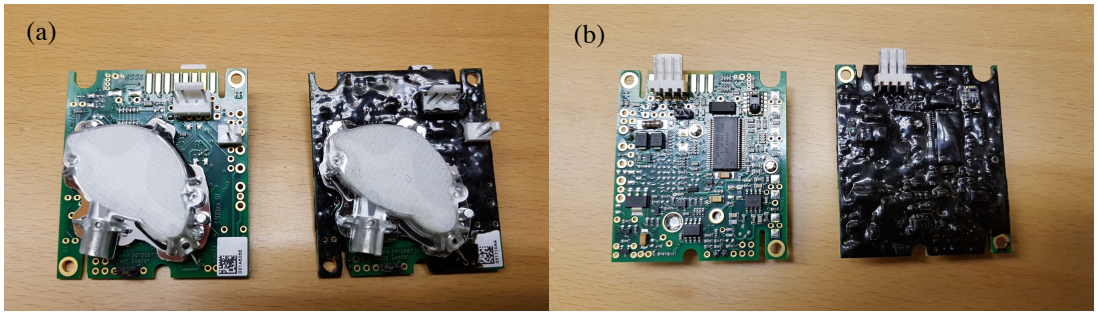

Figure S7: (a) Top view of CO2 sensor (left: non-coating, right: coating); (b) bottom view of CO2 sensor (left: non-coating, right: coating).

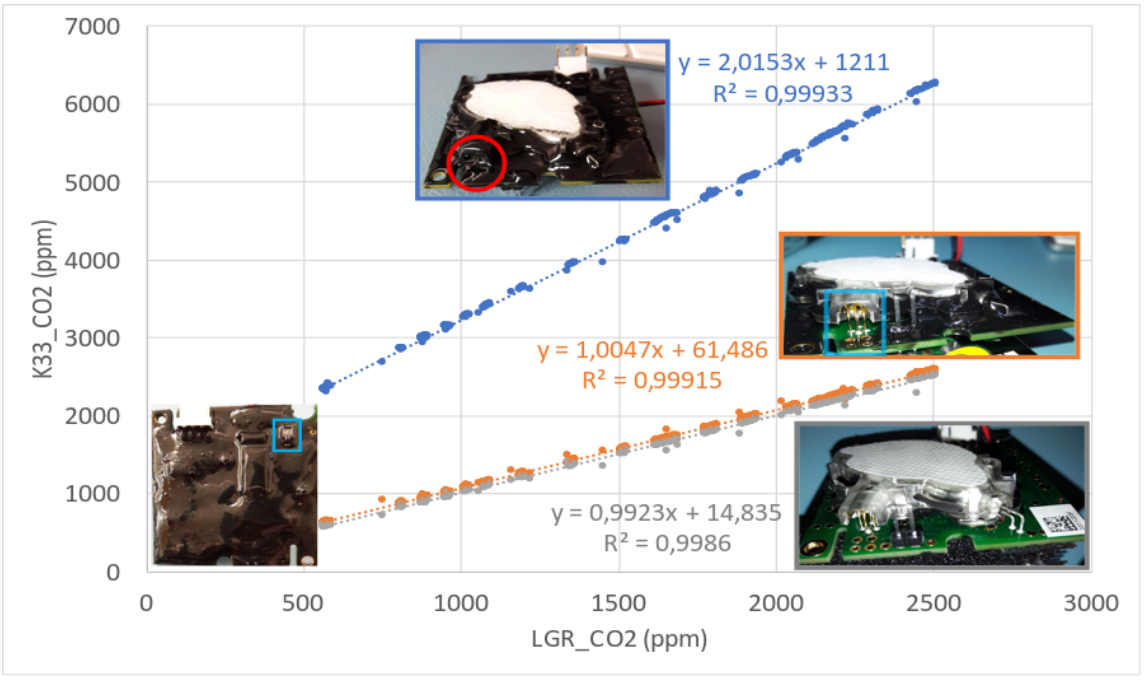

Figure S8: This figure shows calibrations curve of CO2 sensor with LGR analyzer. The grey line is from a non-coating CO2 sensor, the orange line is from a coating CO2 sensor but not at 4-pin detector (blue rectangle), the blue line is from a complete coating CO2 sensor even at the 4-pin detector (red circle). Strongly Recommend to leave this 4-pin detector non-contact with any coating material.

S.1.2 Important improve to prevent chamber sinking and flipping

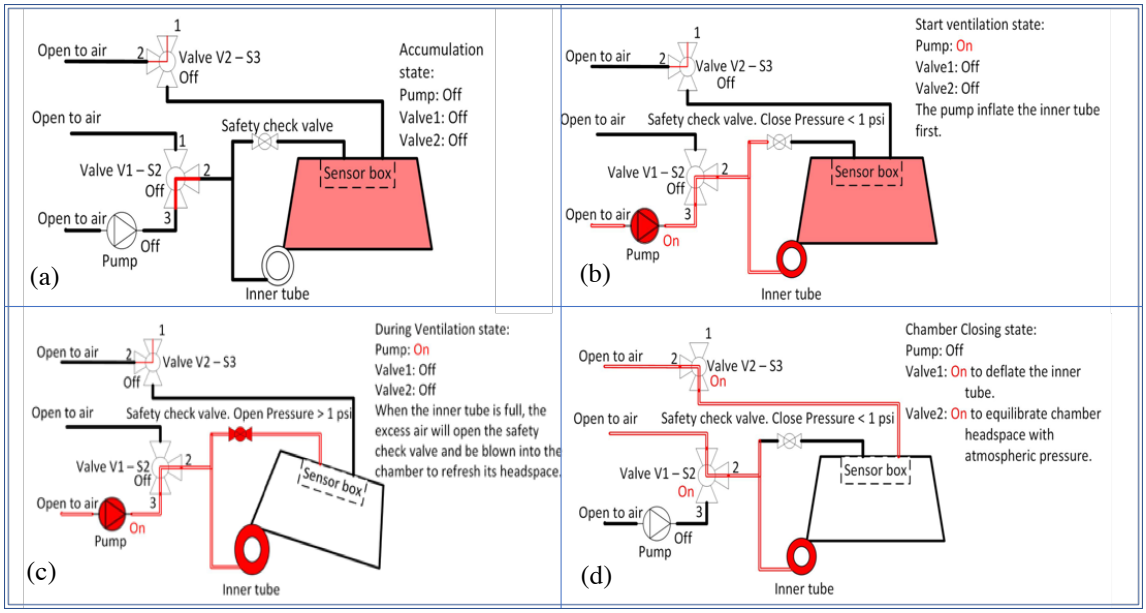

Figure S9: Simplified illustration of no sink AFC cycle including (a) accumulation state, (b) start ventilation state, (c) during ventilation state and (d) inner tube deflating to close the chamber for a new accumulation period. The figure is not to scale.

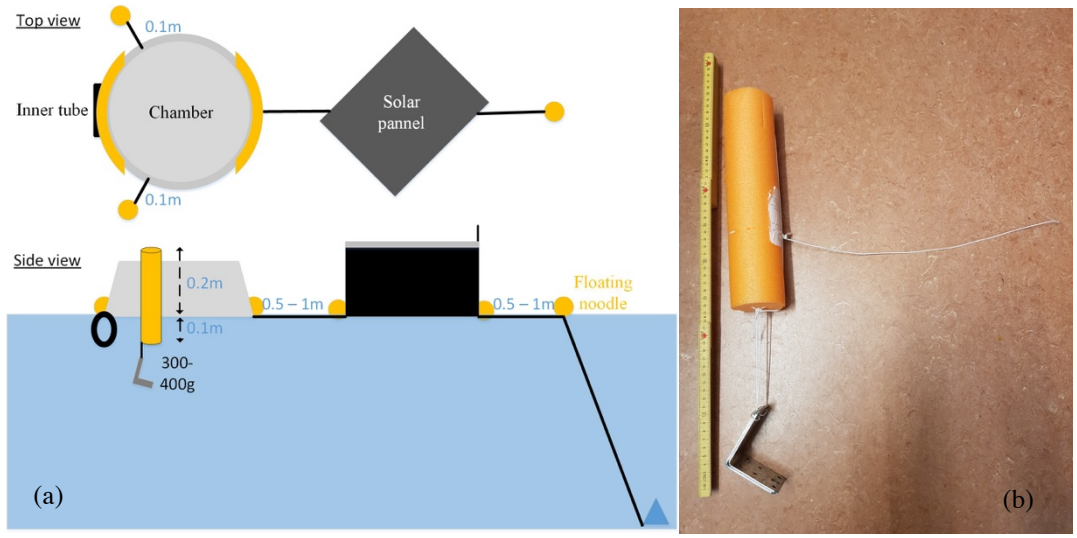

Figure S10: (a) Simplified illustration of preventing chamber flip by attached anti-flipping anchor to the chamber. (b) The anti-flipping anchor is made from 30 cm floating noodle and 300 - 400 g metal L shape. Two of these anchors are attached around the inner tube side (opening side in ventilation process) of the chamber with 10 cm string from the chamber. The 300-400 g anchor keeps 20 cm of floating noodle above water surface. This prevents the floating noodle move into the chamber when the wind pushes the chamber over the anti-flipping anchor.

S.2 A Wireless Control and Data Acquisition System using the DigiMesh protocol on Xbee modules

This section describes a data acquisition system for a network of semi-independent datalogger modules communicating based on embedded 16-bit microcontrollers connected via a wireless network with a mesh topology.

The network consists of a number of semi-independent data-taking nodes, called *clients*, that record triggered events and periodic environmental data. The clients in the network are controlled, monitored and read out by a designated *controller* that is in turn connected via serial cable to a conventional or embedded PC *host*. The controller discovers and monitors the status of active clients, and functions as a communications bridge between the clients and the host. Configuration and control commands from the PC can be sent to all or selected clients, and data collected by each client are read out and sent to the PC for storage.

S2.1 Hardware

Both the controller and monitors are implemented on a single hardware datalogger module. The module is based on a 16-bit microcontroller, the Microchip PIC24FJ128GA010, running a stand-alone code written in C and implemented in the Microchip MPLAB development environment. For simplicity in the field, all dataloggers have the same code, which includes both the controller and client functionalities. The module's address, manually encoded in an 8-pin DIP switch, determines which functionality is used.

The microcontroller has a large number of digital and analog I/O pins, along with a number of peripheral interfaces. Its electronic schema is shown in Figure S11. The interfaces used in the datalogger are described in the following sections:

Serial ports

The microcontroller has two RS232 serial interfaces with built-in hardware UARTs, designated U1 and U2.

Port U1 is connected to an external 9-pin DSUB connector through a MAX 3223 line driver/receiver for communicating with the host. The nominal rate is set at 57600 baud, with 8 data bits, no parity, and no hardware handshaking.

Port U2 is directly connected to the serial communications port of the XBEE wireless transceiver module (described below). The nominal communication rate is 9600 baud (may be raised later). The port itself is configured for 8 data bits and no parity, but a software-based handshake mode with the XBEE module is implemented using general-purpose I/O for RTS and CTS pins.

Serial programming interface (SPI) to SD card

Serial programming interface SPI2 on the microcontroller is interfaced with a SD card socket. A file I/O interface based on code by Lucio Di Jasio allows a local FAT16 file system with storage capacity up to 2 GB.

8-bit address switch

An 8-bit address switch is connected to eight digital I/O pins on the microcontroller. The microcontroller reads the switch settings to determine its system address. By convention, the lowest address (0x00) designates the controller, while higher addresses are for the clients and other functions.

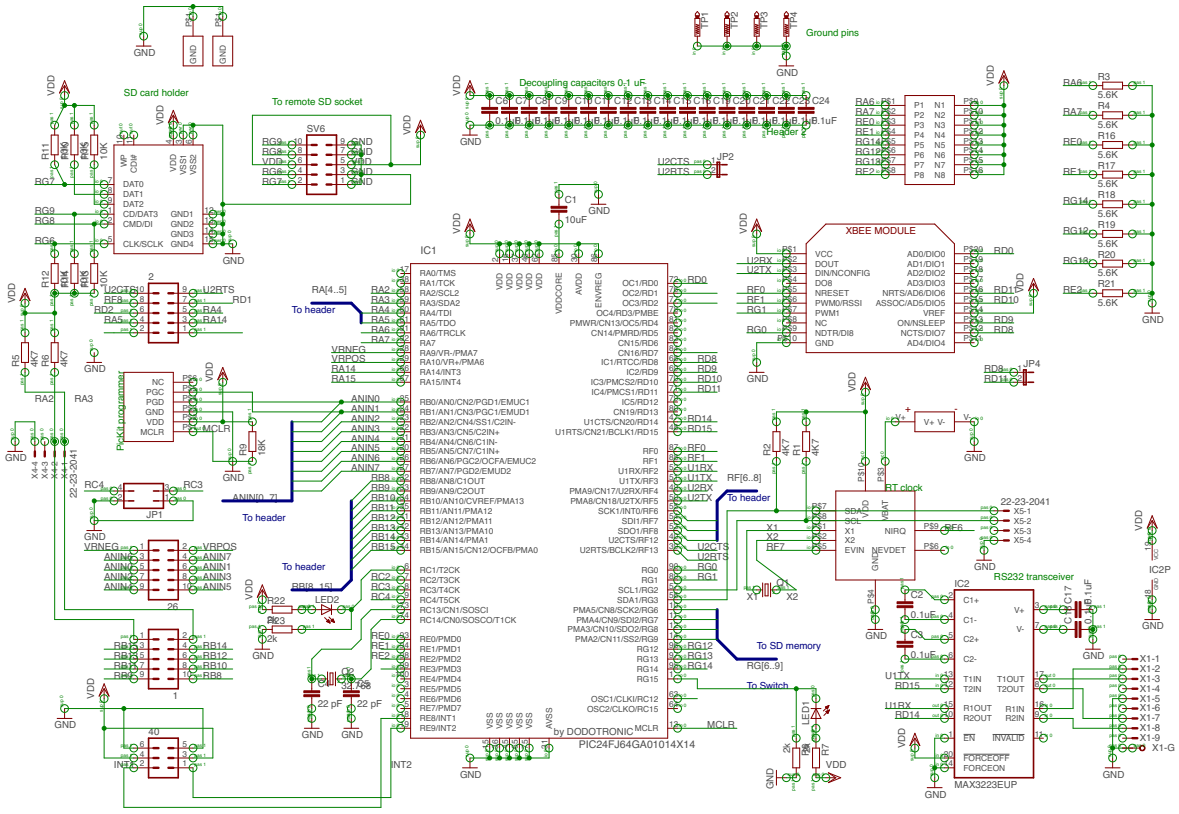

Figure S11: Electronic schema of the datalogger board.

S2.2 XBEE wireless module

Each datalogger module has a socket for an XBEE or XBEE Pro S1 wireless communication module, loaded with firmware that implements a wireless mesh topology using the DigiMesh wireless protocol. Documentation for the DigiMesh protocol can be found at the www.idigi.com website. A set of simplified commands to the XBEE (contained in xbeetestlib.c) has been developed to hide some of the complexity of the DigiMesh protocol from the user, but a broad overview outlined in the following section is useful to understand how the control and data acquisition system works.

S2.3 Digimesh networking features

The Digimesh networking protocol has several key features that make it particularly suitable for the data acquisition protocol described in this document. As a mesh network, there is a peer-to-peer architecture where all nodes are equal. This allows any board to act as a controller or client as needed, and no parent-child relationships need be explicitly set during deployment. The network is self-configuring, with the nodes discovering and creating network maps only as needed. It is also self-healing, allowing any node to enter or leave the network at any time without causing the network as a whole to fail. And the low-level protocol includes a set of acknowledgements that assure reliable delivery of data.

The DigiMesh protocol includes a so-called API mode, which provides a structured interface allowing one-to-many and many-to-many data communications between peers. Each node has a IEEE 64-bit network address identical to its unique hard-wired serial number. Peer-to-peer messages include both source and destination network addresses. Broadcast messages have no destination address, but the receiving nodes can extract the network address of the sending node from the message. Both broadcast and peer-to-peer messaging are used in the application.

S.3 Overview of the data acquisition system protocol

The data acquisition system is based on a set of command-response protocols, with wireless communications always initiated by the controller. To simplify the protocol, all serial communications are encoded in ASCII format, in order to avoid reserved characters in the message fields that could disrupt the API protocols.

S3.1 Serial protocol between host and controller

Communications between the host and controller are framed between a start and end delimiter. The start delimiter is the character '*' (hex 2A, dec 42), and the end delimiter is the character @, (hex 40, dec 64). Serial data not framed by these delimiters are ignored.

The command format begins with a start delimiter and a single-byte resource identifier. These are optionally followed by one or more arguments, and then a stop delimeter is issued:

*	Resource	Argument(s)	@
1 byte	1 byte		1 byte

The current PC-controller protocol must include the following operations, but there is room to expand the protocol as needed:

Resource (ASCII)	Name	Argument(s) (ASCII)	Response
1 (dec 49)	Discover new nodes	none	list of new nodes
2 (dec 50)	Poll all known nodes	address 00-FF	list of all known nodes, and their status
3 (dec 51)	Control virtual switch	address 00-FF	Start/Stop the AFC accumulation processes
4 (dec 52)	Read out node	address: 00-FF	formatted output from node's data buffer
5 (dec 53)	Set node register	address: 00-FF, register 00-FF, value 0000-FFFF	node address, register address and contents
6 (dec 54)	Read node register	address: 00-FF, register 00-FF	node address, register address and contents
7 (dec 55)	Set node time	address: 00-FF, time: YYMMddhhmmss	address: 00-FF, time: YYMMddhhmmss
8 (dec 56)	Read node time	none	address: 00-FF, time: YYMMddhhmmss
9 (dec 57)	Control Refresh sampling manifold	address: 00-FF	Activate a process to clean up a sample manifold with atmospheric air.

Communications are typically, but not necessarily, initiated by the PC. In some cases, more than one response may be returned for a single command. For instance, if a "Discover" command is issued, the controller can return several responses, one for each newly found node in the system.

5 S3.2 Controller wireless protocol

The role of the controller is to discover and monitor the status of all clients in the network, and to mediate commands and readout requests between the PC and the clients.

To keep track of the clients, the controller software uses an array that is indexed by the 8-bit system address. For each client address the 64-bit network address (if known) is stored, along with an integer "status" that indicates what is known about that client.

Before a client node is discovered, its default status is -1. Periodically the controller will broadcast a "Discover new nodes" (an ASCII '1') command throughout the network. Clients that are entering (or re-entering) the network will send a response containing their system address (00-FF) and their 64-bit network address. After discovery, the status of the client is set to 1, where it remains as long as the client continues to respond to commands.

For commands other than "Discover", the controller sends a text message addressed directly to each client and waits for the appropriate response. If a client does not respond within a given time-out period, the controller increments the client's status by one. After a given number of failed responses, the controller sets the status of the client to 0, indicating that the client has been removed from the network and no new commands should be sent to it until it responds to a new "Discover" broadcast.

S3.3 Client wireless protocol

The role of the client is to keep track of whether it is "known" by the controller, to respond to "Discover" broadcasts appropriately if it is not known, and to respond correctly to direct commands from the controller.

To participate in the network, the client must only keep track of the controller's 64-bit network address and its own internal "status". After a power-up or reset, the client begins with a status of -1. When a "Discover" broadcast is received, the client extracts the controller's network address from the message, sends a response with its own system and network addresses directly to the controller, and sets its own internal status value to 1, indicating that it is now "known".

Every time the controller receives a direct command from the controller, the client resets its status to 1, indicating that the controller is still aware of its presence. It is therefore important that every known client should receive at least one direct command from the controller between any two "Discover" broadcasts. If two or more consecutive "Discovers" are received without direct commands between them, the client increments its internal status to show that it has been "missed". After a given number of "misses", the client sets its status to 0, indicating that it is no longer a "known" client, and must respond to the next "Discover" command to rejoin the network.

A client should have the ability to remove itself from the network for other reasons, including some local failure, low voltages, etc. In this case the client should respond to a "poll" command with a special status code (F) to indicate its error status.

Available sensor data and status of client system will be sent to the controller after the client has established a well communication with controller.

S.4 Installing the host on Raspberry Pi

S4.1 Obtain or install a Raspbian OS filesystem on SD card

There are different ways to do this, depending on whether you work in Windows, MacOS or Linux. The latest version of Raspbian "wheezy" can be found at the following URL:

http://www.raspberrypi.org/downloads

Once you have downloaded the zip file, consult this page for instructions on setting up the file system on your SD card:

http://elinux.org/RPi_Easy_SD_Card_Setup

S4.2 Boot the Raspbian system for the first time and configure it

Insert the card in the Raspberry Pi, connect a keyboard, mouse, monitor and power. After boot-up you will see a Raspi-config menu. Here you need to do a couple of
things. Use arrow keys to navigate to a selection, then press "Return" and follow
instructions.

1. **expand_rootfs** This will expand the root partition so that your system uses all of the available storage in the SD card.
2. **configure_keyboard** If your keyboard is not a standard US English one, run this option to choose a suitable configuration
3. **change_pass** The standard password for the "pi" account is "raspberry".
Since this is general knowledge, it is important that you change this to
something more secure.
4. **change_timezone** Set the time zone to the one your experiment will use.
5. **ssh** Enable the ssh sever, so that you can securely access the RPi remotely.
6. **boot_behaviour** use this to have the RPi automatically start the desktop when
you boot.

When you are finished, use the arrow keys to navigate to “Finish”, and reboot the computer. It will come up again in desktop mode.

If you want to change any of these settings later, you can simply run the utility again by typing “raspi-config” from the command line.

S4.3 Downloading and installing new software

Like most Linux systems, Raspbian has a simple utility for downloading and installing new software and utilities. Here is an example for installing the popular “emacs” editor on your new system.

Begin by bringing up a new command line terminal. You can do this, for instance, by opening “LXTerminal” from the desktop.

In the command line, enter:

sudo apt-get install emacs

S4.4 Install and configure the lighttpd web server

First you need to set up a user that the web server can use:

sudo adduser www-data www-data

Next run the install, this will take a minute or two:

sudo apt-get install lighttpd

You can check that the server is installed correctly by typing:
ps -u www-data u

The main web server is now installed and can be checked by opening a browser and entering the address of your server and checking Lighttpd's default information page:

http://your.raspberry.pi.IP address

The root of your main directory is at /var/www. The host web tools require a data directory below this with the proper ownership and permissions, so enter the following:

```
sudo mkdir /var/www/data
sudo chmod 777 /var/www/data
```

set permissions for user group www-data

```
sudo chgrp -R www-data /var/www/
sudo chmod -R g+w /var/www/
```

Finally, configure lighttpd for cgi support, password protection, and directory listing in the /data directory. To do this, you will need to edit /etc/lighttpd/lighttpd.conf.

```
sudo emacs /etc/lighttpd/lighttpd.conf
```

Expand the server.modules list at the top to include the following modules:

```
server.modules = ( "mod_access", "mod_cgi", "mod_alias", "mod_compress", "mod_redirect", "mod_auth")
```

The DocumentRoot, which is the directory under which all your HTML files should exist, is set to /var/www/.

```
server.document-root        = "/var/www/"
```

Next, configure cgi support. For this example, you will install cgi files in the directory /usr/lib/cgi-bin. Add to the bottom of the configuration file:

```
# configure cgi support
$HTTP["url"] =~ "/cgi-bin/" {
```

```
cgi.assign = ("" => "")
}
alias.url += ( "/cgi-bin/" => "/var/www/cgi-bin/" )
```

Next, enable directory listing of the files in the /data directory. Add:

```
# allow directory listing of data files
$HTTP["url"] =~ "^/data($|/)" {dir-listing.activate = "enable" }
```

Finally we will add basic password protection for the cgi-bin directory. In this case, we will create a password file named ".lighttpdpassword" in the directory /home/lighttpd. This directory does not exist, so you need to create it as you did the data directory above.

```
sudo mkdir /home/lighttpd
```

In that directory, create the file .lighttpdpassword, and add one or more user names and passwords.

```
sudo emacs /home/lighttpd/.lighttpdpassword
```

This example line creates a user named "Your_user_nameXX" and password "Your_passwordXX":

```
Your_user_nameXX:Your_passwordXX
```

In the lighttpd configuration file, add the following:

```
# Password protect the cgi-bin directory
$HTTP["url"] =~ "^/cgi-bin" {
auth.backend       = "plain" auth.backend.plain.userfile = "/home/lighttpd/.lighttpdpassword"
```

```
auth.require = ( "/cgi-bin/" => (
"method" => "basic",
"realm" => "Password protected.",
"require" => "user=Your_user_nameXX")
)
}
```

There are more secure ways to do this, i.e. with encrypted passwords, but these are outside the scope of these instructions.

Reboot the system when you are finished to implement the webserver changes.

S4.5 Install the Host controller software and web interface

To install the host controller software, you will need to copy producer.zip to your user directory on the Raspberry Pi. This can be done with a secure FTP program, or from the command line of a Mac or Linux system:

```
scp producer.zip pi@(your.pi.IPaddress):~
```

In the Pi account, unzip the file:

```
unzip producer.zip
```

Go to the producer directory:

```
cd producer
```

When you are ready, you can compile all of the programs with the command:

```
./doit.sh
```

Congratulations, your host is fully installed and operational!

To start the producer, type the command:

./producer /dev/ttyUSB0

To run the host remotely, independent with terminal remote connection, tmux software is needed:

sudo apt-get install tmux

S4.6 Setup host-controller in outdoor environment

The host raspberry pi connects to a controller board as in Figure S5c. To access, mornitor and control the webserver in outdoor environment, an outdoor Wifi network is setup using a Wifi router. The raspberry pi is connected to the router via an ethernet cable. The raspberry pi will get an internal IP address from this router. This eth0 is shown by command:

ip -4 addr show eth0 | grep -oP '(?<=inet\s)\d+(\.\d+){3}'

Users use a browser software to access the web server
http://your.raspberry.pi.IP address

Log in locally or remotely as user “pi” and type the command:

ssh pi@your.raspberry.pi.IP address

Enter your raspberry pi password.

Right after login, update real date time:

date -s "Fri Aug 22 16:40:00 EDT 2014"

Go to the producer directory:
cd producer

Open tmux terminal shell:

tmux

To start the producer:

./producer /dev/ttyUSB0

Detach the tmux terminal:

CTRL + b d

Log out the remote access:

exit

Figure S12: Picture of a complete setup host-controller to be deployed in outdoor environment.