Title
PolyWAG: Autonomous filtered water sampling for eDNA

Authors
Kai Roy * Riley Prince * Marc Belinga * John Selker * Chet Udell

Affiliations
(Insert Affiliations)

Corresponding author's email address and Twitter handle
Institutional email address preferred. If you have a Twitter handle, please add it here 'twitter: ....'

Abstract
Environmental DNA, eDNA, is an ideal way of researching aquatic environments to determine what species are present in an area the biodiversity of an area, and if any invasive or endangered species are present . Traditional sampling of environmental DNA (eDNA) consists of manually filtering water, which is labor and cost-intensive for remote locations. Furthermore, commercialized solutions are either expensive or require a field operator to function. We have built an eDNA capable of autonomous multi-sampling for a greatly reduced price compared to existing technologies. Our PolyWAG eDNA sampler system is a water sampling device that collects DNA samples via 47mm filter and provides a non-invasive, safe and autonomous means of eDNA collection. The sampler can hold 24 filters and is designed to be easily replaced and reusable. A browser application is used for real-time monitoring, scheduling tasks, and data logging for time, pressure, flow, and filtered volume. Additionally, the sampler design is openly published, modular and is constantly being tested to help us optimize our software and hardware to give us the best results. The 9-step sampling sequence helps reduce cross contamination significantly. Our machine can be deployed for an extended period. It is completely autonomous and costs around $6000.

Keywords
Environmental DNA * Sampling * Arduino * Data Logging

Specifications table

Hardware name	*PolyWAG*
Subject area	*Environmental, planetary and agricultural sciences*
Hardware type	*Field measurements and sensors*
Closest commercial analog	*Dartmouth Ocean Technologies' eDNA Sampler*
Open source license	CERN Open Hardware License GNU General Public License v3.0
Cost of hardware	*$3800 (Cost of just components)* *$6000 (Cost with labor included)*
Source file repository	*If you've uploaded your source files to an approved repository (OSF, Mendeley Data or Zenodo) write the DOI URL here. For exemple: http://doi.org/10.17605/OSF.IO/WGK7Q*

1. Hardware in context

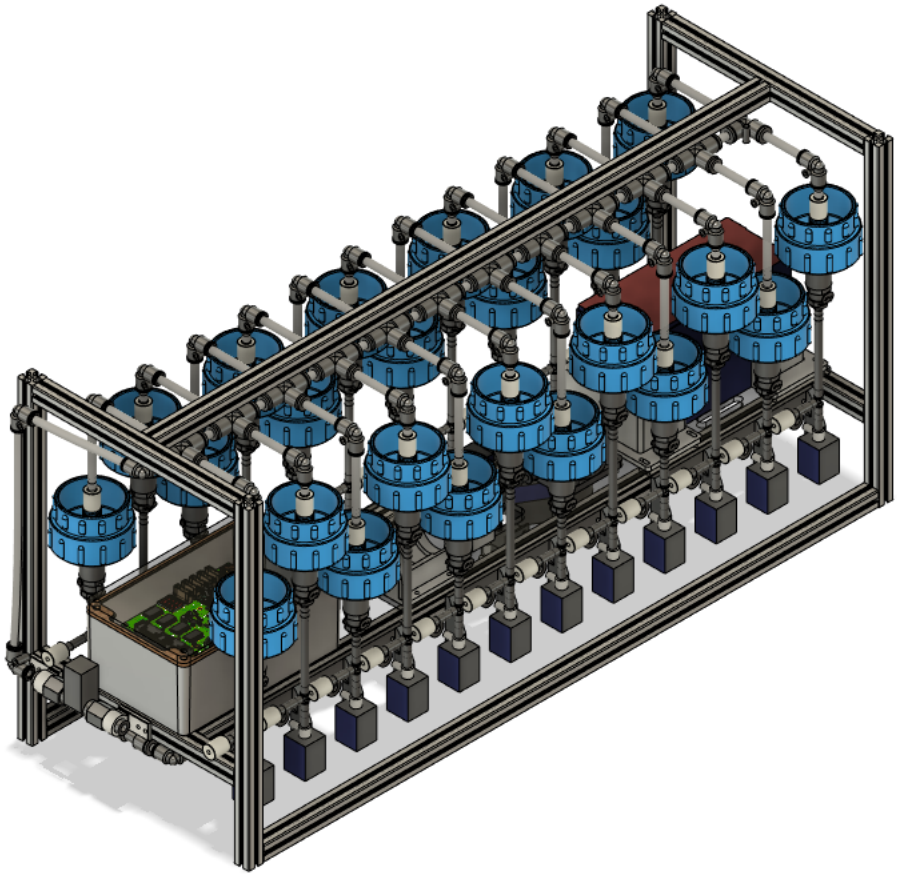
Environmental DNA (eDNA) is DNA derived from mucus, feces, gametes, and carcasses [1]. Many things can be learned once this DNA is put through sequencing. eDNA can be used to determine what species are present in an area, the biodiversity of an area, and if any invasive or endangered species are present [2]. eDNA sampling provides scientists and researchers a non-invasive, rapid, cost-effective and sensitive way to detect and quantify species in many environments.

Traditional sampling of environmental DNA consists of manually filtering water, often requiring one or more researchers to be on location for days or weeks [3]. The filtration process varies depending on the researcher, but it is common to pull a sample of water with a bottle and pour that water into a funnel containing a filter. This can be connected to a vacuum pump to expedite the filtering process. After the sampling process is completed, the filters need to be preserved and the setup cleaned to avoid cross contamination [3]. This process is labor intensive, cost intensive, and can be dangerous, especially for remote locations. While commercialized solutions to this problem exist, they either still require an operator to be on location or are very expensive. Smithroot's commercial solution offers a simplified process with additional data collection such as GPS location for a fair price, around $8,000[6]. A disadvantage of this solution is that it is not fully autonomous, still requiring an operator to be on location to use the device [4]. An alternative is the DOT Sampler which is a fully autonomous solution that is capable of multiple samples (20+ samples) and is also submersible but comes at a cost of $55,000 [5].

The solution designed by the OPEnS Lab is the middle ground of these two solutions. While it is not submersible (limiting its potential sampling environments), it is capable of autonomous, multi-sample operations for extended periods of time (approximately one month) for the cost of $6,000. The two core priorities for our design are its autonomous function and the cross-contamination. The autonomous function of the sampler is important for a handful of reasons. An autonomous system requires less researcher hours spent in the field. This has cost benefits from the reduced hours worked and safety benefits when sampling in hazardous environments.

2. Hardware description

The eDNA sampler we have developed is an autonomous multi-sampling device that collects eDNA samples from water via 47mm filter holders and provides a non-invasive, safe, and autonomous means of DNA collection. The sampler can hold 24 of these filter housing and are designed to be easily replaced and reusable. The sampler is controlled by a custom logic board with an Adafruit M0 Feather Wi-Fi microcontroller loaded with a webserver to act as the interface for the sampler's operations. This webserver hosts a browser application which is used for real-time monitoring, scheduling tasks, and data logging for time, pressure, temperature, flow, and sample volume. This data is located stored onto an SD Card for later data analysis.

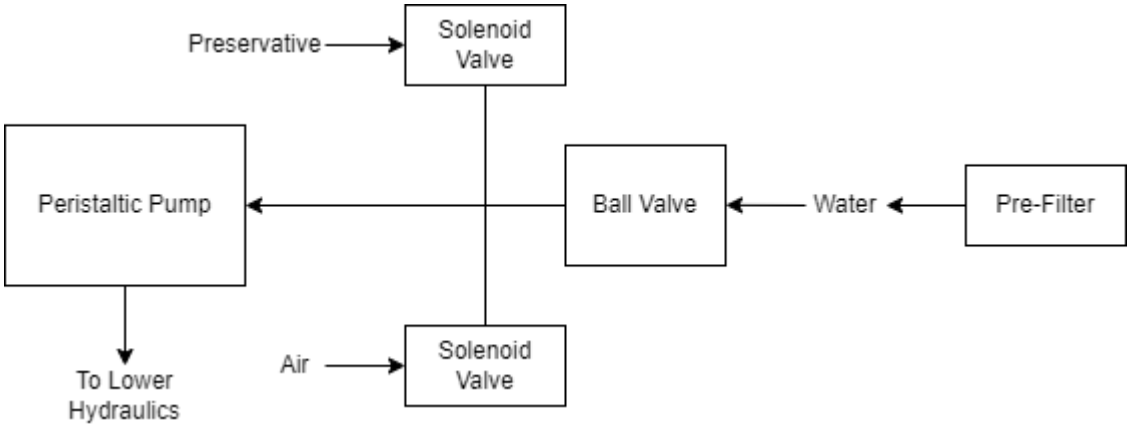


2.1 Hydraulics

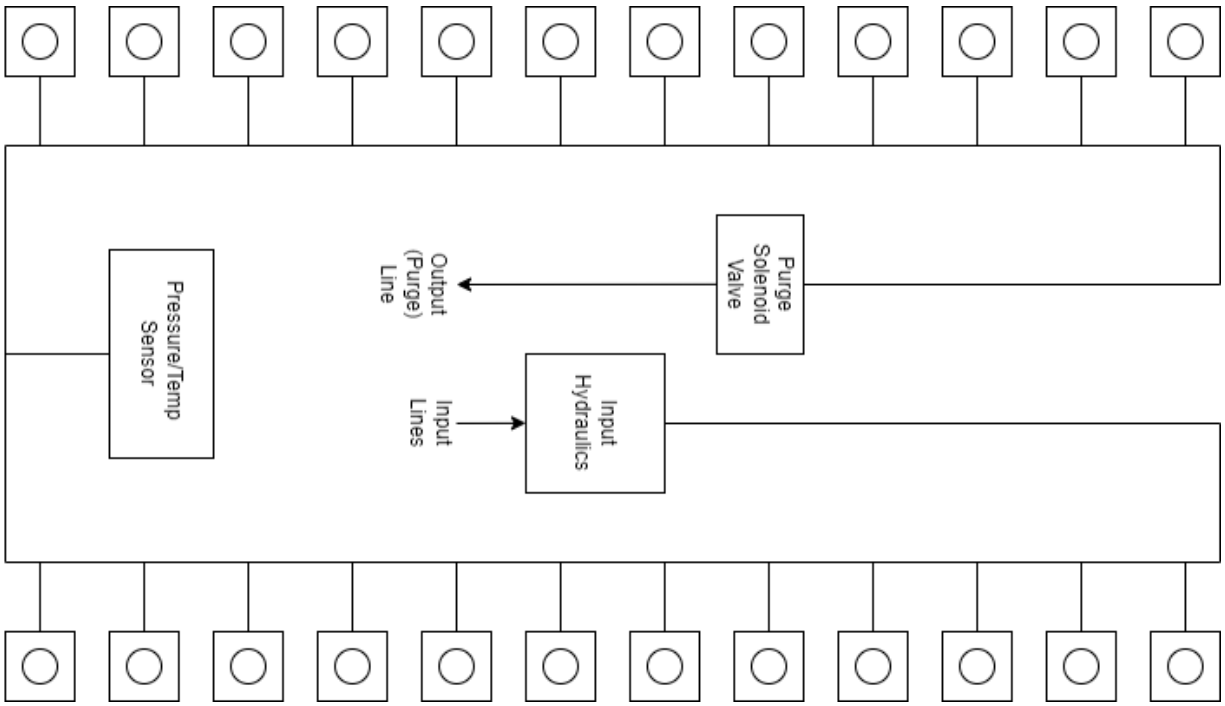
The hydraulics of the sampler can be roughly split into the following sections:

- The Pump and Inputs
- The Lower Hydraulics
- The Filters
- The Upper Hydraulics and Outputs

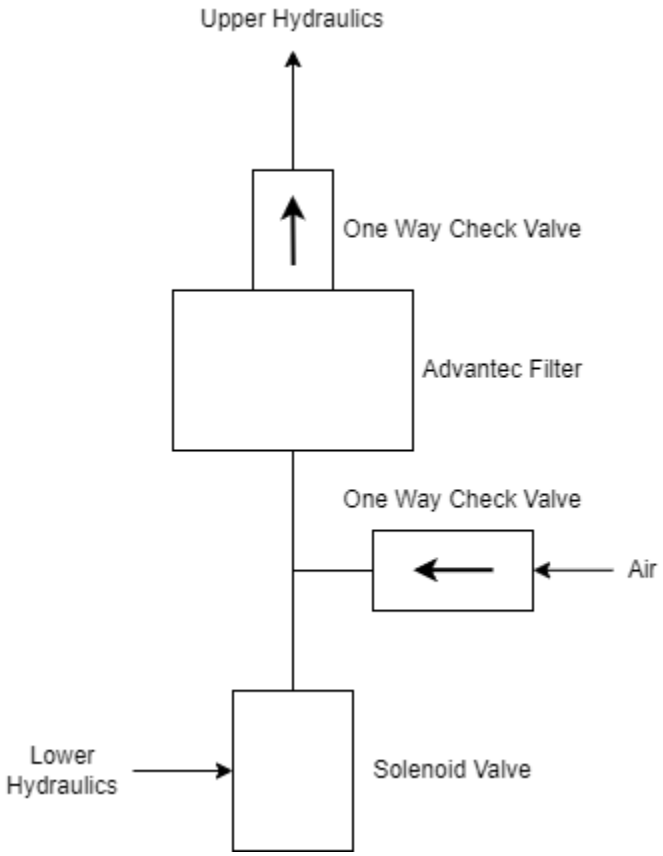
There are three inputs into the sampler: one for air, one for preservative, and one for water. The preservative input is connected to a hydration bladder where the preservative of choice can be stored. The water input has a prefilter at the front end of the tube to prevent debris from entering the sampler. Three valves are used to control the flow from these inputs with the air preservative being regulated by a solenoid valve and the water being controlled by a ball valve. These three valves connect into a single tube connected to the input of the peristaltic pump. The pump is capable of 400ml/min of flow under ideal conditions. The output of the pump connects directly into the Lower Hydraulic Rail.



The Lower Hydraulic Rail consists of 24 solenoid valves connected parallel to each other which controls which filter liquid flows through. The valves are split into two sets, one on each side of the sampler. In between these two sets is a M32JM-000105-100PG pressure and temperature sensors. The temperature is logged for later use and the pressure is used for monitoring, stopping an operation if the pressure exceeds a certain margin. At the end of the Lower Hydraulic Rail is another solenoid valve which allows for the lower hydraulics to be purged of their current contents when necessary.

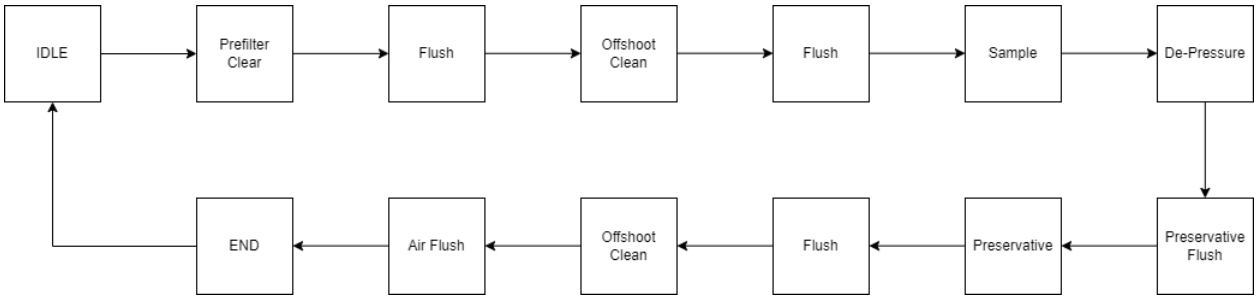


After the solenoid valve there is a tee connection that goes to a one-way check valve and a modified Advantec filter. The one-way check valve allows air into the solenoid valve that opens when the pump runs backwards. The Advantec filter is modified with a CPC quick disconnect and a one-way check valve. The one-way check valve is connected to the Upper Hydraulics and is used to prevent liquid from going backwards through the filter. The Upper Hydraulics simply connects the output of all the filters to one central line that goes through a flow meter and out of the sampler.



2.2 Sampling Procedure

Having worked on multiple iterations of the sampler, we have decided to go with a 13-step sampling sequence that helps reduce cross contamination significantly. This sequence can be split into 9 unique steps: Idle, Prefilter Clear, Flush, Offshoot Clean, De-pressure, Sample, Preservative Flush, Preservative, Air Flush, and End.



The Idle state is the default state of the sampler. This is where the sampler waits for a signal from the RTC to move to the first/next state of the Sampling Sequence. If the sampler is not in sleep mode, this is when a client would interact with the UI to do a handful of tasks such as setting up a Sampling Schedule or using the other task utilities. If the sampler is in sleep mode, then only the RTC and supporting circuits are powered. This means there is no way to interact with the sampler without exiting sleep mode.

Once the RTC sends the signal to start a sample procedure, the sampler enters the Prefilter Clear (PC) state. In this state, the purge and input ball valve are opened, and the pump is run in the backwards direction. This will allow for air to flow from the purge and out the input line. This is used to clear the prefilter of anything that might be clogging it, such as accumulated debris. This state runs for X seconds, before moving onto the next state.

The Flush state and the proceeding Offshoot Clean (OC) state are used to prepare the lower hydraulics before the Sample state. The Flush state starts with the purge valve and the ball valving opening, then the motor starts to run in the forward direction. This fills the lower hydraulics with sample liquid and clears out/dilutes and liquid that remained from previous sample. The Flush state runs for the time specified when the Sampling Schedule is created. We recommend a Flush time of 6 minutes.

The OC state closes the purge valve and opens the filter valve (for the filter which is about to be used. The pump runs backwards for a few seconds. This clears anything that might be in the tube between the valve and the filter (what we refer to as the offshoot). The Flush state is run one more time before moving to the Sample state.

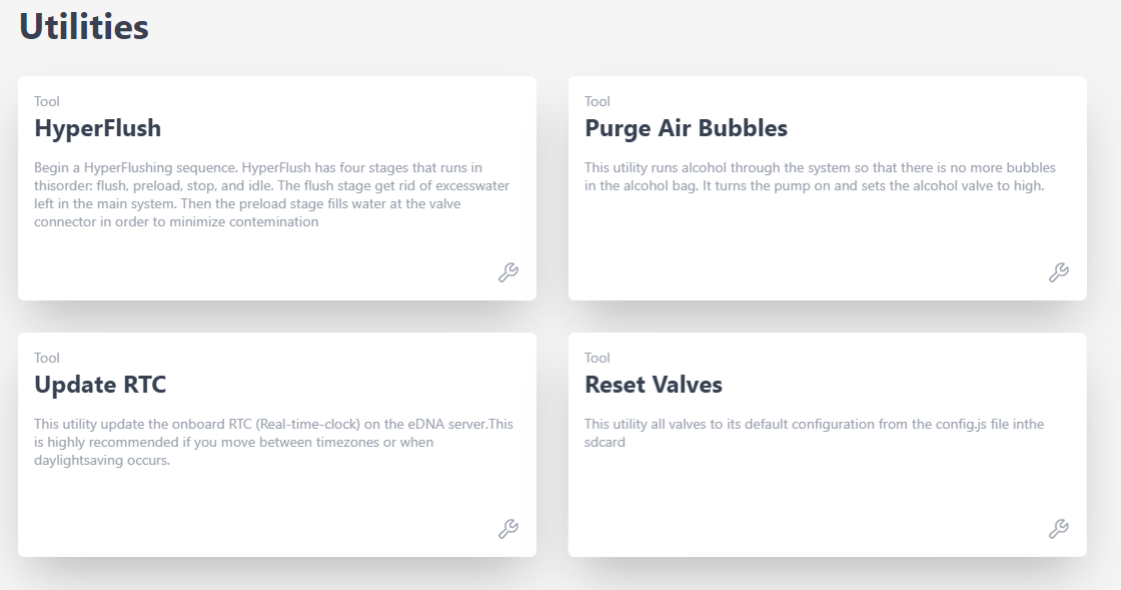
Like the name suggests, the Sample state is where the system pushes the sample water through the filter. This is done by opening the Filter and Ball Valve and running the pump in the forward direction. The system moves to the next state when the target Sample Volume is reached. This volume is measured by a Flow Meter on the filter output line. Ideally, the state is terminated when the target volume is reached. There is an additional condition that will end the Sample state and that is the Sample Time. This time cutoff was added since the filter clogs and the flow rate decrease rapidly during the sample process. To prevent the sample state running for too long, the time limit was implemented. Both conditions are set during task scheduling. Since the pressure greatly increases due to the clogged filter, the de-pressure state is used to reduce the pressure in the lower hydraulics to ensure that the valves can operate consistently.

The Preservative Flush (PF) and Preservative (P) states are the next states in the sequence after the de-pressure state. The PF state is nearly identical to the Flush state except the Preservative input valve is used instead of the ball valve. The goal of this state is to saturate the lower hydraulics with preservative, preventing additional sample water that may have been stored in the lower hydraulics from going through the filter. If this water was allowed through the filter, then the Sample Volume would be inaccurate by the end of the sequence. The P state is like the Sample state except preservative is the input fluid instead of sample water. This state runs for a time specified by the user during scheduling.

After the P state, another Flush and OC state runs to purge the leftover preservative in the lower hydraulics. After these two states, an Air Flush (AF) state is run which is identical to the Flush and PF states but uses the air valve as the input instead of the other two inputs. This ensures that any liquid that is in the lower hydraulics is purged.

After the AF state occurs, the system sets an RTC alarm for the time of the next sample. The system then moves into Idle and if the system was in sleep mode, then the system will go into its low power state.

2.3 Utilities



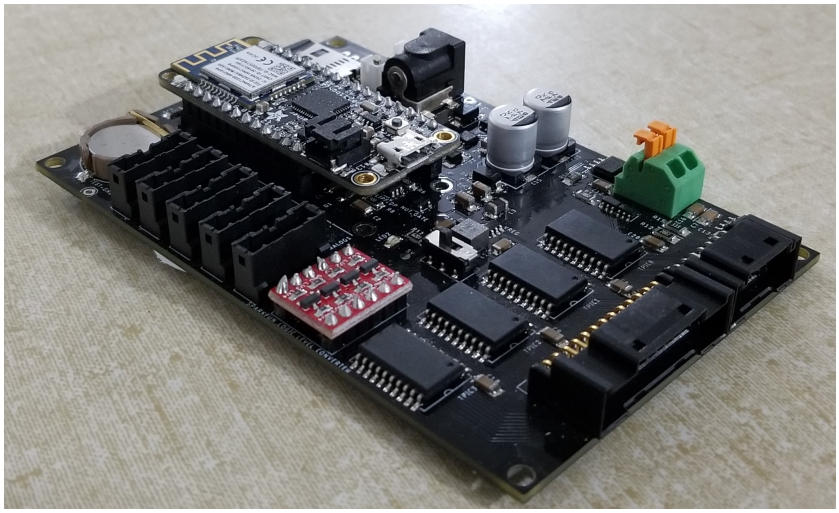
The HyperFlush utility runs water through every filter sequentially for a few seconds per filter. This is mainly used for cleaning out the system after a sample task (i.e., a set of 24 samples) to prevent any unwanted cross contamination. This utility can also be used to test the basic functionality of the sampler, as nearly every component is activated during this sequence.

The Preservative Air Purge (PAP) utility turns the pump on and opens the alcohol valve for 10 seconds. This runs some alcohol through the system and removes air bubbles from the alcohol bag. Often it helps to use this utility multiple times and to tilt the Preservative Bladder so that the air is near the port.

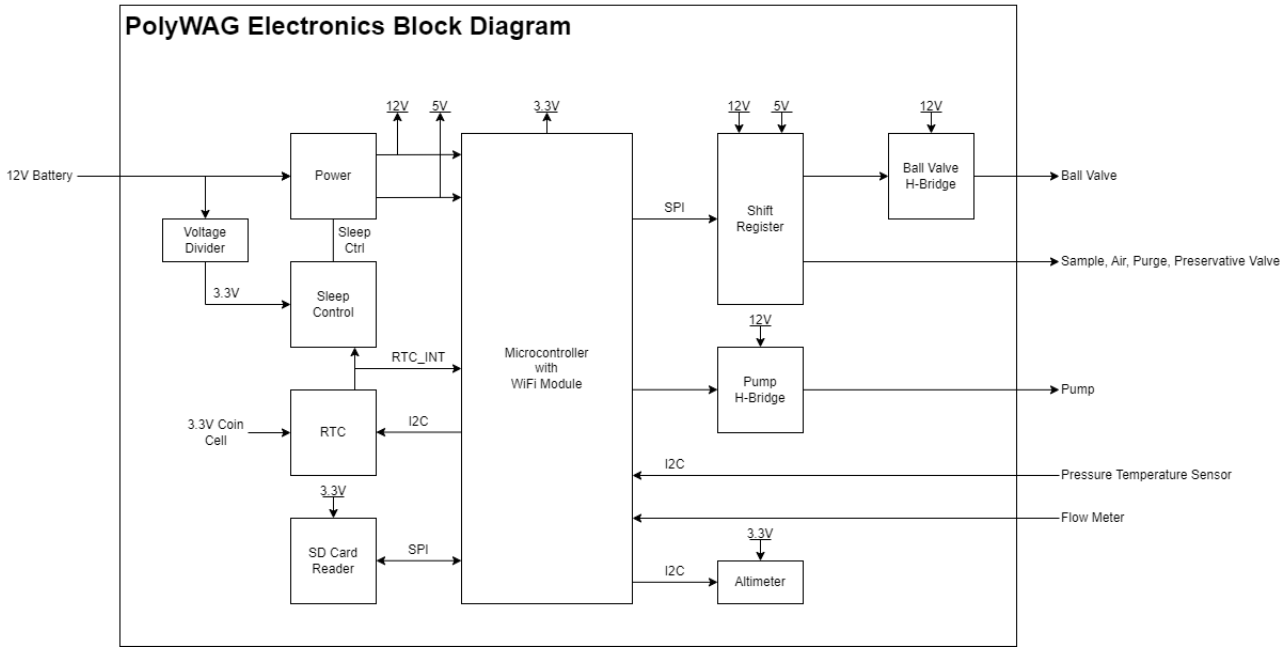
The Update RTC utility is needed to make sure that the time on the sampler matches your local time, so scheduling a task will remain accurate. Whenever the system is fully depowered (ie the battery is removed), or when new code is uploaded to the microcontroller, the RTC will need to be updated. It is also recommended that the RTC is updated when there is a daylight-saving change, or when you move between time zones.

The Reset Valves Utility is used when valves have been sampled that you want to be sampled again. This is required since the system 'locks' the filter valves when they have been used in a sample, this prevents samples from being corrupted accidentally. The code does not let you sample a valve multiple times without being reset to prevent messing up a sample. It is important to note that this utility will reset all valves, not a specific one.

2.4 Electronics



The PolyWAG Sampler is designed with a custom electronics control board that can be split into 8-10 blocks with an Adafruit Feather M0 at its core. These blocks consist of the microcontroller/Wifi Block, Power, RTC, and sleep control blocks, and the output blocks consisting of the Shift Register, Pump, and Ball-Valve H-Bridge Blocks.



The power block consists of a reverse polarity current (RPC) circuit and a voltage regulator circuit. The RPC Circuit was added to protect the 12V battery from current flowing backwards through the system. While the battery has its own protection circuits, they lock the battery in the case of a short and need to be reset using the battery charger. The RPC circuit was added to prevent any "permanent" power loss while in the field. The voltage regulator circuit is a 12V to 5V regulator with an enable pin that connects to the sleep control circuit. This is used to save power during long term deployments.

The RTC and sleep control circuit are used to keep track of time and to save power respectively. The sleep control circuit controls the output of the power circuit and is constantly being power by a simple voltage divider circuit. It is basically a Flip Flop circuit that is reset when the RTC triggers an interrupt. The RTC circuit is used to keep track of the time between samples and is powered by a coin cell while power is off. This allows it to keep accurate track of time and signals an interrupt when its internal alarm is triggered. This interrupt

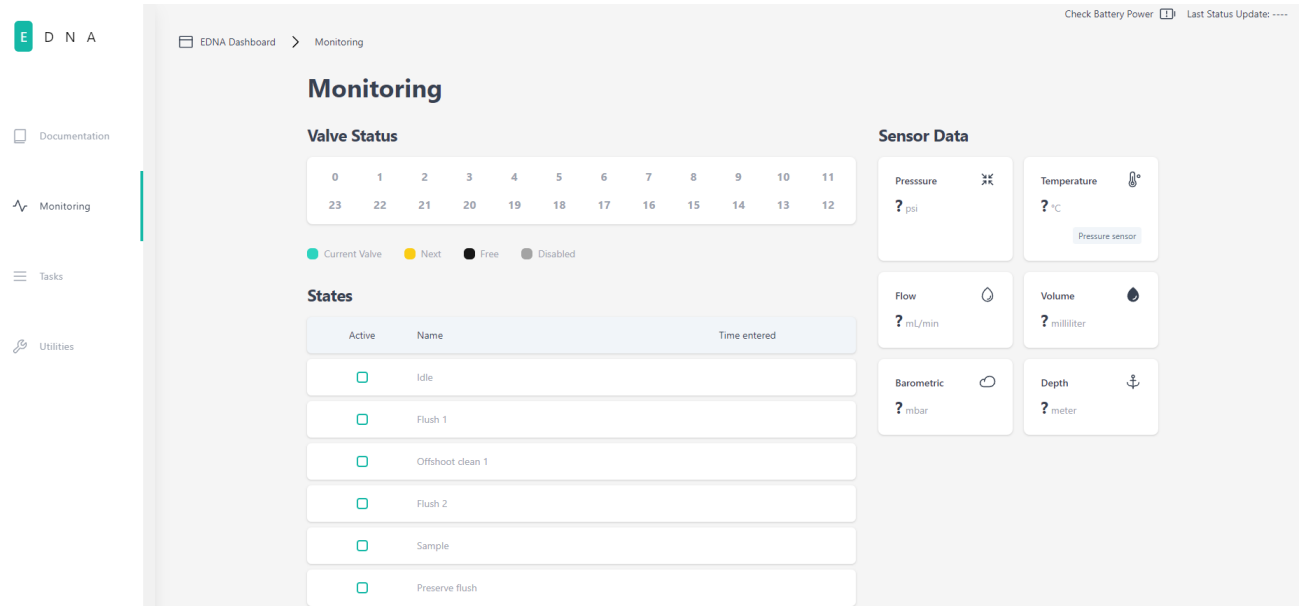
is used to both turn power back on and to inform the microcontroller that it is time for a sample. If noise causes the sleep control circuit to reactivate power, the microcontroller will see that the RTC did not trigger the interrupt and will fall back into power saving mode.

The shift register circuit consists of 4 8-bit shift registers connect to the microcontroller via SPI. The shift registers are pull-down style shift registers where the 'output' pins are pulled to ground. This allows the shift registers to control devices that use higher logic voltages. This allows us to control the 27 12V solenoid valves with a 5V IC. The shift registers are also used to control the H-bridge for the Ball valve. The H-bridge for the pump is controlled directly by the microcontroller itself.

The board contains an SD Card circuit for datalogging purposes. The data is logged every second and includes the current state, time, and data from the sensors. The sensors include an in-line pressure temperature sensors for monitoring the lower hydraulic line and a flow meter out the output for measuring volume.

The microcontroller of choice is an Adafruit Feather M0 WiFi. The WiFi version of the Feather M0 was chosen as the user interface requires the feather to host a webserver.

2.5 User Interface



PolyWAG Sampler hosts a webserver that can be connected to via a browser. This acts as the user interface for the system. There are three main sections that make up the user interface: monitoring, tasks, and utilities. The monitoring page displays the data from the sensors, the current state of the sampling procedure, and information on the sampling valves such as the current valve being sampled, and which valves are locked and unlocked. The utilities page is used to activate the utilities mentioned earlier. The tasks page is where sampling tasks are created. Multiple tasks can be created, and each task is saved in memory for later modification and use. This page is also where tasks can be scheduled for sampling. Each task contains the information on which valves are being used as well as for how long each state occurs.

2.6 Task Configuration

Design files

Your design files should be editable - see OSHWA's open source definition of 'Documentation' for further details. You must then either:

- *Upload your design files to one of the three approved online repositories - Mendeley Data (instructions), the Open Science Framework (instructions) or Zenodo (instructions). We recommend this option as the repositories support versioning of files.*
- *Upload your design files as supplementary materials (e.g., CAD files, videos. . .) to Hardware X's online editorial system when you submit your manuscript.*
- *Include your design files in the body of the manuscript (e.g., as figures).*

CAD files: You are encouraged to use free and open source software packages for creating the files. For CAD files, OpenSCAD, FreeCAD, or Blender are encouraged, but, if these are not available, we accept source files from proprietary CAD packages, such as Autocad or Solidworks, and other drawing packages.

3D printing: Supplementary files that facilitate digital replication of the devices are encouraged; for example, STL files for 3D printing components. We recommend uploading CAD files to the NIH 3D Print Exchange as Custom Labware and then entering the link here.

Electronics: PCB layouts and other electronics design files can be uploaded to the Open Hardware Repository or other repositories or as supplementary materials.

Software and firmware: All software files used in the design and operation of the hardware should be included in the repository. Provide a description of the software and firmware and use extensive comments in the code.

3. Design files summary

Complete a separate row for each design file associated with your hardware (including the primary design files). Any empty rows should be deleted.

Design filename	File type	Open source license	Location of the file
For example: Design file 1	*e.g. CAD file, figures, videos*	*All designs must be submitted under an open hardware license. Enter the corresponding open source license for the file.*	*Either enter the URL for the repository or the sentence: "Available with the article".*
. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .

For each design file listed in the summary table above, include a short description of the file below (just one or two sentences per design file).

Bill of materials

Given the number of materials required to build a PolyWAG Sampler, The BOM will be located in an external file and can be found here

4. Build instructions

Given the complexity of the PolyWAG Sampler, a Build Instructions section of sufficient detail would be near a hundred pages long. This is why an external build guide document will be linked for those who are interested in knowing how one of these samplers are assembled. This file can be located here

5. **Operation instructions**

6. **Validation and characterization**

Ethics statements

The work does not use any human or animal subjects.

CRediT author statement

Kai Roy: Project administration, Hardware(Electrical), Validation, Writing- Original draft
Riley Prince: Project administration, Methodology, Validation, Data curation, Writing- Original draft
Marc Belinga: Software, Investigation, Writing- Original draft
Torrey Menne: Conceptualization, Hardware
Bao Nguyen: Project administration, Conceptualization, Hardware(Electrical)
Nikhil Wandhekar: Project administration, Hardware
Nathan Jesudason: Software, Validation
Kawin Pechetratanapanit: Conceptualization, Software
John Selker: Funding acquisition, Supervision
Chet Udell: Funding acquisition, Supervision
Cara Walter: Funding acquisition, Supervision, Resources

Auther: Contribution.

Acknowledgements

All contributors who do not meet the criteria for authorship should be listed in an acknowledgments section.

In addition, please list any funding sources in this section. List funding sources in this standard way to facilitate compliance to funder's requirements:

Funding: This work was supported by the National Institutes of Health [grant numbers xxxx, yyyy]; the Bill & Melinda Gates Foundation, Seattle, WA [grant number zzzz]; and the United States Institutes of Peace [grant number aaaa].

It is not necessary to include detailed descriptions on the program or type of grants and awards. When funding is from a block grant or other resources available to a university, college, or other research institution, submit the name of the institute or organization that provided the funding.

If no funding has been provided for the research, please include the following sentence:

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

If relevant, you should include a reference to the original publication of the hardware you customized and a reference to the repository in which your design files are published. Other references can be included, as required; for example, references that put your device in context in the literature. For more information on the reference format in HardwareX please see the Guide for Authors.