# Open-LIFU*

## Documentation for Developers


**Disclaimer**

This document is provided as a draft version of the user manual. All information contained herein, including but not limited to product features, specifications, and descriptions, is subject to change without notice. The contents should not be considered final, and Openwater makes no guarantees, express or implied, regarding the completeness, accuracy, or reliability of this information. The final published user manual will supersede this draft.

Document ID: ER-00015
Software Version: 1.19.0
Revision: A
Release Date: 5/07/2026
Authors: David Paribello, Peter Hollender, George Vigelette, Dan Blizinski

*The Open-LIFU platform is not yet cleared by the FDA and is intended for research purposes only. Not for commercial sale.

Introduction....4
Overview....4
Best Practices....4
Design Philosophy....5
Configuration and Change Management....5
Versioning....5
Verification & Validation....5
Documentation....5
Maintenance Strategy....6
Firmware Best Practices Guidelines....6
Software Best Practices Guidelines....6
License....7
Contribution Guidelines for Openwater....7
Code of Conduct....7
Getting Started....8
Making Contributions....8
Best Practices....8
Recognition of Contributions....8
Asking for Help....8
The Openwater Community....9
Community & Support....9
The Open-LIFU System....10
Open-LIFU Device Specifications....10
Open-LIFU Console Specifications....10
Transmit Module Specifications....11
Transducer Configurations and Specifications....12
Transmit Module Acoustic Output Specifications....13
Sonication Sequence Specifications....13
System Components....14
System Architecture....14
System Hardware....15
Console....15
Transducer....17
Coupling Pad....18
Additional Hardware Requirements....19
PC....19
Android Phone....19
Open-LIFU Software Development....19
Overview....19
Software Architecture....19
Open-LIFU Desktop Application....20

Open-LIFU Slicer Extension..... 21
Open-LIFU Python..... 21
Hardware interface SDK..... 22
Python Library Installation (Optional)..... 23
Embedded Firmware Layer..... 23
Local Data Storage..... 23
Android Application..... 25
Tools Used..... 25
Test & Validation Utilities..... 25
Transmit Modules..... 25
Transducer Logical Architecture..... 27
Transmit Module Wiring Diagram..... 28
Configured as first module (active lines Magenta)..... 28
Configured as middle module (active lines blue)..... 29
Configured as last module (active lines green)..... 29
Open-LIFU Hardware Development..... 30
Overview..... 30
CAD Files..... 30
Mechanical..... 30
Open-LIFU Mechanical Design Files..... 30
Electrical..... 30
Open-LIFU PCBs..... 31
Slicer Open-LIFU: Research Use Only (Advanced Users)..... 31
Install Slicer with the Open-LIFU Extension..... 31
Download Slicer..... 31
Install The SlicerOpenLIFU Extension..... 31
Slicer Modules..... 32
Data Management..... 33
Sonication Protocol..... 34
Pre-planning..... 34
Transducer Localization..... 35
Sonication Planning..... 37
Sonication..... 37

# Introduction

## Overview

The information provided in this document will enable users to start developing with Openwater's Open-LIFU[1] platform.

Open-LIFU (the System) is designed to emit programmable sequences of steered, Low Intensity Focused Ultrasound (LIFU). The System is composed of a console, a transducer, and a software program, and is supported by an Android mobile application. Through software configuration, users can use the System to generate sequences of LIFU pulses at chosen timings and amplitudes, steered using one or more transmit modules, each containing a 2D matrix array, located in the transducer. A deformable acoustic coupling pad provides a conformal fit to a variety of curved subject surfaces, while the headset-style transducer housing securely and comfortably holds the transducer in place. A photogrammetry-based transducer tracking system provides a method of locating the transducer position relative to an MRI coordinate system by matching rigid geometry from the MRI to the geometry reconstructed from photogrammetry.

The System is available in two different frequency variants, 155kHz and 400kHz, and in two form factors: a single module transducer for shallower targets, and a dual-module transducer for deeper targets. As an open-source platform, the system may be modified to achieve alternative customized form factors for specialized applications.

## Best Practices

Openwater is committed to meeting customer expectations by supplying high-quality, safe, and reliable products and services. Visit Openwater's GitHub page to read the Quality Statement.

Openwater recommends all developers follow open-source standards along with the appropriate regulatory standards for the region of development. Contact local authorities to get a complete list of local rules and regulations.

Open-LIFU software follows a **modular, iterative, and incremental development lifecycle** aligned with the principles of **IEC 62304**. Development is managed via:

- **Version-controlled repositories** (Git + GitHub)
- Feature-driven development with **issue tracking and pull requests**
- Milestone-based delivery and **release tagging**

Each major software component (firmware, FPGA logic, Python SDK, test tools) is maintained in its own repository.

[1] The Open-LIFU platform is not yet cleared by the FDA and is intended for research purposes only. Not for commercial sale.

## Design Philosophy

Openwater's designs and processes are governed by our Quality Management System and abide by these core principles:

- **Safety-First**: Built-in safety systems ensure safe operation within regulatory limits
- **Open Standards**: All product designs are released, enabling academic and commercial use with attribution
- **Modularity**: Component-based architecture allows customization of individual subsystems without redesigning the entire platform
- **Accessibility**: Designed to be manufacturable using standard fabrication methodologies and widely off-the-shelf components
- **Collaborative Iteration:** A transparent and safety-prioritized feedback loop allows the gathering of community input and contributions through an open-source platform

## Configuration and Change Management

- All source code and build tools are maintained in **Git repositories**
- Each commit is **tracked, reviewed**, and tagged
- **Issues and bugs** are logged in GitHub Issues with associated milestones
- **Change logs** are maintained per release
- Critical changes are subject to peer code review

## Versioning

Versioning follows **Semantic Versioning (SemVer)**:

MAJOR.MINOR.PATCH (e.g., v1.2.3)

- **MAJOR**: Breaking architectural/API changes
- **MINOR**: New backward-compatible features
- **PATCH**: Bug fixes or performance improvements

## Verification & Validation

- **Unit tests** implemented for embedded firmware and Python code
- **Manual and automated test procedures** executed on every release:
    - USB streaming verification
    - Histogram decoding accuracy
    - Frame sync and laser timing tests
    - Full-system functional regression tests
- Test logs, waveform captures, and timing measurements are archived for traceability

## Documentation

- Software is documented using:
    - In-code documentation
    - GitHub README files

    - Architecture/design documentation in Word and Markdown

- User documentation is provided for:
    - SDK usage
    - CLI tools and GUI
    - Developer onboarding

## Maintenance Strategy

- Bug reports and feature requests are triaged on GitHub
- Routine releases are issued quarterly or as needed
- Backward compatibility is maintained within major versions
- Deprecated features are clearly marked and scheduled for removal in future releases

## Firmware Best Practices Guidelines

- **Embedded Firmware** (STM32):
    - Developed in C, using STM32Cube HAL and CMSIS
    - MISRA C guidelines were applied where practical to improve safety, portability, and static analysis results
    - Code structured for:
        - **Readability** (clear naming, small functions, single responsibility)
        - **Safety** (defensive programming, bounds checking, explicit initialization)
        - **Testability** (hardware abstraction, minimal global state)
    - Additional Best Practices:
        - Use explicit-width types (uint32_t, int16_t) for all interfaces and registers
        - Avoid hidden side effects in macros; prefer static inline functions
        - Initialize all peripherals and state explicitly; avoid reliance on reset defaults
        - Separate driver, middleware, and application logic
        - Use assertions and error return codes for fault detection
        - Guard the ISR code for minimal execution time and deterministic behavior
        - Enable compiler warnings (-Wall -Wextra) and treat warnings as errors where feasible

## Software Best Practices Guidelines

- **Python SDK and GUI Tools**:
    - Compliant with **PEP8** and type-annotated
    - Fully type-annotated using Python typing
    - Modular design with clear separation of:
        - Hardware interface/transport
        - Data processing and analysis
        - GUI and user interaction
    - Tested on Python 3.10+, on Windows 11 and Linux
    - Additional Best Practices:
        - Use dataclasses or typed models for structured data
        - Avoid hardware access in GUI threads; use async or worker threads
        - Validate all external inputs and device responses
        - Centralize error handling and logging

- Write unit tests for protocol parsing and data processing
- Keep hardware interfaces mockable for offline testing
- Document public APIs and SDK usage with docstrings

### License

All Open-LIFU source code is released under the GNU Affero General Public License v3.0 (AGPLv3). Developers working with or building on Open-LIFU must understand the key obligations this license imposes.

Copyleft applies to network use: Unlike the standard GPL, the AGPLv3 extends the copyleft requirement to software used over a network. If you modify Open-LIFU and provide access to it as a service (e.g., via an API or web interface), you must also make your modified source code publicly available.

Derivative works must remain AGPLv3: Any modifications, extensions, or software that links against Open-LIFU code and is distributed or deployed must itself be released under the AGPLv3. You cannot relicense the code or incorporate it into a proprietary product.

Source code must be made available: When distributing AGPLv3-licensed software (in binary or source form), you must include or offer access to the complete corresponding source code, including any modifications you have made.

License and copyright notices must be preserved: All copies of the software must retain the original copyright notices, license text, and any notices that refer to the absence of warranty. The full license text is available in the LICENSE file in each repository and at https://www.gnu.org/licenses/agpl-3.0.html. Developers with questions about compliance or licensing exceptions should contact Openwater before integrating Open-LIFU into their own systems.

All Open-LIFU hardware designs are released under the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license. This license allows anyone to share and adapt the designs for any purpose, including commercially, as long as appropriate credit is given and any derivative works are distributed under the same license. The full license text is available at https://creativecommons.org/licenses/by-sa/4.0/

## Contribution Guidelines for Openwater

### Code of Conduct

Please read our Code of Conduct before contributing. We believe in a respectful and collaborative environment, and our Code of Conduct outlines our expectations for all participants.

## Getting Started

- Fork the Repository: Start by forking the repository to your GitHub account.
- Set Up Your Environment: Follow the setup instructions in the README to get your development environment ready.

## Making Contributions

- Issues:
    - Before creating new issues, please check if the issue already exists to avoid duplicates.
    - When creating an issue, provide a clear and detailed description, including steps to reproduce, if applicable.
- Pull Requests:
    - Branching: Create a new branch for each feature or fix.
    - Commit Messages: Write clear, concise commit messages with a brief description of the changes.
    - Code Standards: Ensure your code adheres to the project's coding standards and guidelines.
    - Testing: Add relevant tests for your changes and ensure all tests pass.
    - Documentation: Update the documentation to reflect your changes, if necessary.
    - Code Reviews: Once you submit a pull request, the project maintainers will review your changes. Be responsive to feedback and make necessary adjustments.

## Best Practices

- **Stay Updated**: Regularly sync your fork with the main repository to keep up with the latest changes.
- **Communicate**: If you are working on a particular issue, let others know so you do not overlap with their efforts. Use the project's communication channels for discussions.
- **Be Respectful**: Respect the time and effort of maintainers and fellow contributors.

## Recognition of Contributions

We value your contributions, and recognition will be given for each contribution. Contributors will be acknowledged in the project's documentation and release notes.

## Asking for Help

If you need help or have any questions, reach out to the community through

- Discord
- GitHub
- Email

# The Openwater Community

Have a question or bug, or just want to connect and get involved with the Openwater Community?

Visit Openwater's Community page here:
https://openwaterhealth.github.io/openwater-community/

## Community & Support

Have questions? Join our community Discord server to connect with other members, ask for help, and stay updated on project developments!

Visit: Discord Openwater

# The Open-LIFU System

The Open-LIFU hardware platform consists of three primary components:

1. **Console**: The main control unit containing high-voltage power supplies, timing controllers, and communication interfaces. The console generates and coordinates the electrical signals that drive the ultrasound transducer.
2. **Transducer**: A wearable headset-style housing containing:
    - One or more Transmit Modules, each containing a 2D matrix array of ultrasound elements (64 elements per module) and the electronics necessary to generate steered and focused beams from
    - Deformable acoustic coupling pad for conformant fit to curved anatomical surfaces
    - Unique geometric features to aid in spatial tracking and registration
3. **Supporting Hardware**:
    - An Android mobile device for capturing collections of photos used for photogrammetric reconstruction (“Photocollections”)
    - Connectivity between PC application and the console via USB
    - Optional water tank testing accessories

## Open-LIFU Device Specifications

### Open-LIFU Console Specifications

The console connects to a PC via a USB-C cable and contains a bipolar high voltage supply, capable of producing up to +/- 60V or +/- 65V[2], with a maximum power output of 110W for driving the ultrasound elements, while also supplying low voltage power and USB communication to the transmit electronics.

[2] Refer to the device serial number to determine which size power supply is installed in the console. 120 W power supply units are capable of +/- 60 V, whereas 180 W power supply units are capable of +/- 65 V.

*Table 1: Open-LIFU Console Specifications*

| | |
|---|---|
| Operating Voltage[3] | 100-240 VAC, 50-60 Hz, 1.2 Amps (120 W)<br>100-240 VAC, 50-60 Hz, 3.5 Amps (180 W) |
| High Voltage Output[3] | +/- 10-60 V (120 W)<br>+/- 10-65 V (180 W) |
| Connections | 1x USB-C (USB3.0)<br>1x 12-pin socket |
| Device Status | LED indicator |
| Dimensions (w x h x d) | 9.8 x 3.0 x 6.3 inches (250 x 75 x 160 mm) |
| Weight | 3 lbs (1.36 kg) |

## Transmit Module Specifications

*Table 2: Individual transmit module specifications.*

| | |
|---|---|
| Operating Voltage | 12 VDC, 1.2 Amps |
| High Voltage Input | +/- 0-100 V |
| Beamformer Clock | 10 MHz |
| Array Center Frequency | 155 kHz<br>400 kHz |
| Number of Elements | 64 |
| Array Dimensions | 40 x 40 mm |
| Element Size (w x l) | 4.7 x 4.7 mm |
| Element Pitch | 5 mm |
| Overall Dimensions (w x l x d) | 2.0 x 2.6 x 1.2 inches (52 x 67 x 30 mm) |
| Connections | 1x 12-pin socket (input)<br>1x 30-pin socket (input)<br>1x 30-pin socket (output) |
| Device Status | LED indicator |
| Weight | 155 kHz: 0.4 lbs (187 g)<br>400 kHz: 0.3 lbs (132 g) |

[3] Serial numbers 001 → 012 utilize the 120W power supply. Devices with a serial number starting at 013 and beyond use a 180W power supply. To determine which power supply is installed, refer to the serial number affixed to the bottom of the Open-LIFU console.

## Transducer Configurations and Specifications

Ultrasound pulses are produced from transmit modules located within the transducer housing. Integrated beamforming electronics take DC power from the console and generate the signals required for the piezoelectric arrays to emit the specified sequences of ultrasound pulses.

All transducer configurations are designed as wearable headsets, with a soft strap to go around the head, a soft disposable coupling pad, and padding to prevent the transducer from slipping. The outside of the transducer also has a curved faceplate with an embossed pattern to aid in localizing the transducer. The faceplate can be removed to access the transmit modules for service but should not be removed during normal use.

*NOTE: The transducer is not watertight. Never submerge the transducer in water. Doing so may lead to electric shock or damage the transducer.*

*NOTE: All Open-LIFU transducers have been factory calibrated. Reconfiguring or modifying any transducer immediately voids the factory calibration, requiring users to recharacterize the transducer to ensure it complies with all necessary application-specific requirements.*

Depending on the application, users may need a transducer with a Single (1x) or Double (2x) transmit module configuration. Certain applications may require more transmit modules or modules arranged differently; while such configurations are possible within the principles of the Open-LIFU architecture and generally supported by the software, the customization of Open-LIFU hardware is beyond the scope of this document.

### Transmit Module Acoustic Output Specifications

*Table 3: Open-LIFU 1x and 2x configuration specifications. Note these are representative values only. Exact values will vary based on the serialized transducers.*

| **Transducer Frequency** | **1x 155 kHz** | **1x 400 kHz** | **2x 400 kHz** |
|---|---|---|---|
| Peak P- @ 60V and (0, 0, 50)mm | 0.7 MPa | 1.1 MPa | 2.16 MPa |
| Derated P- ($P_{r.3}$) @ 60V | 0.68 MPa | 1.02 MPa | 2.01 MPa |
| Voltage | +/- 10-60 Vmax | +/- 10-60 Vmax | +/- 10-60 Vmax |
| Mechanical Index (MI) in water[4] @ 60V and (0, 0, 50)mm | 1.7 | 1.8 | 3.18 |
| Axial FWHM @ (0, 0, 50)mm | 34 mm | 31 mm | 15 mm |
| Transverse FWHM @ (0, 0, 50)mm | 8 mm | 4.5 mm | 4.7 mm (elevation)<br>2.4 mm (azimuth) |
| Axial steering range | 4 - 6 cm | 4 - 7 cm | 3 - 11 cm |
| Transverse steering range | +/- 1 cm | +/- 1.5 cm | +/- 2 cm |
| Pressure-voltage sensitivity @ (0, 0, 50) mm focus | 6 kPa/Vpp @ 30 mm | 9.9 kPa/Vpp @ 45 mm | 18 kPa/Vpp @ 50 mm |

# Sonication Sequence Specifications

*Table 4: Open-LIFU sonication sequence specifications.*

| **Category** | **Parameter** | **Description** | **Typical** | **Min** | **Max** |
|---|---|---|---|---|---|
| Pulse | Frequency (kHz) | Pulse center frequency | 155<br>Or<br>400 | N/A | N/A |

[4] Limited to 1.9 by the Open-LIFU software.

| Category | Parameter | Description | Typical | Min | Max |
|---|---|---|---|---|---|
| Pulse | Focal Pressure (kPa) | Peak negative pressure at target | 500 | 0 | 1200 |
| Pulse | Duration (ms) | Pulse duration | 5 | 0.01 | 100 |
| Sequence | PRI (ms) | Pulse repetition interval | 100 | 10 | 1000 |
| Sequence | Pulse count | Number of pulses per train | 300 | 1 | 1000 |
| Sequence | PTRI (s) | Pulse train repetition interval | 30 | 0 | 120 |
| Sequence | Train count | Number of pulse trains | 20 | 1 | Protocol Dependent |

# System Components

## System Architecture

The Open-LIFU system architecture is shown in Figure 1. A mobile device is used to help localize the transducer by capturing a series of images which are then reconstructed into a 3D mesh using local computer processing or cloud compute. The PC runs the Slicer Open-LIFU or Open-LIFU Desktop Application to plan, configure and run Sonications. The Application communicates with the console to configure the high voltage supply and Transducer to execute a Solution. The Transducer is placed on the subject using coupling gel and a disposable coupling pad.

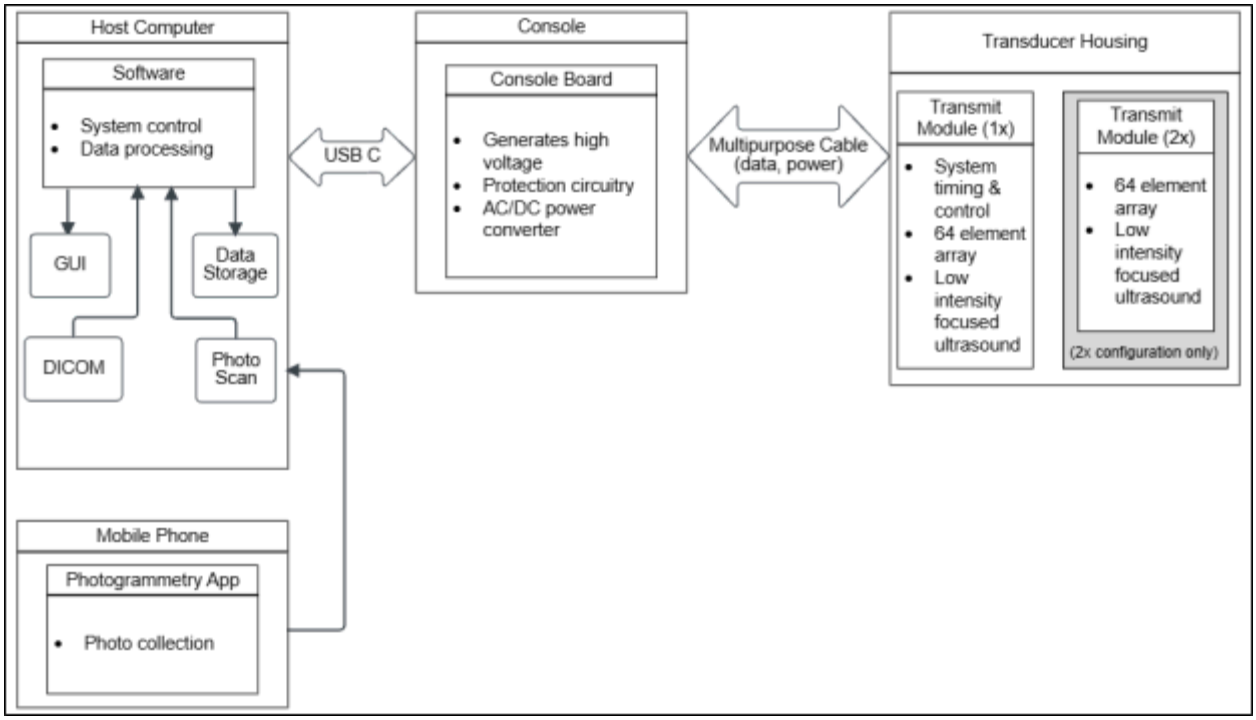



*Figure 1: System Architecture for a 1x and 2x transducer configuration.*

# System Hardware

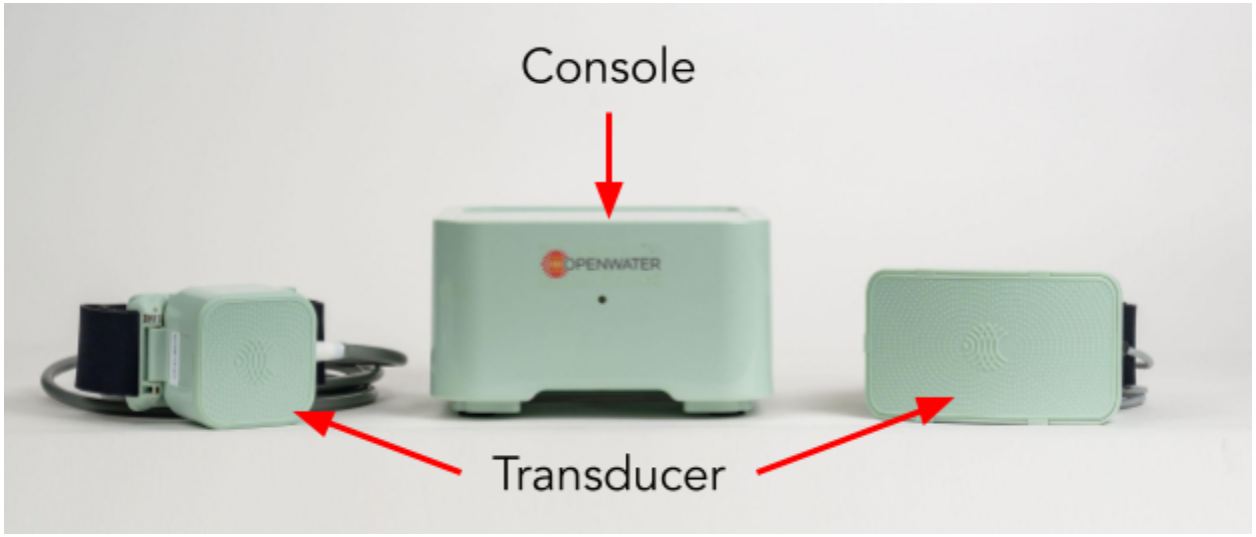



*Figure 2: The Open-LIFU 1x (left) and 2x (right) configuration are shown here in the image.*

## Console

The console connects to a PC via a USB-C cable and generates the positive and negative supply voltages used to drive the transducer.

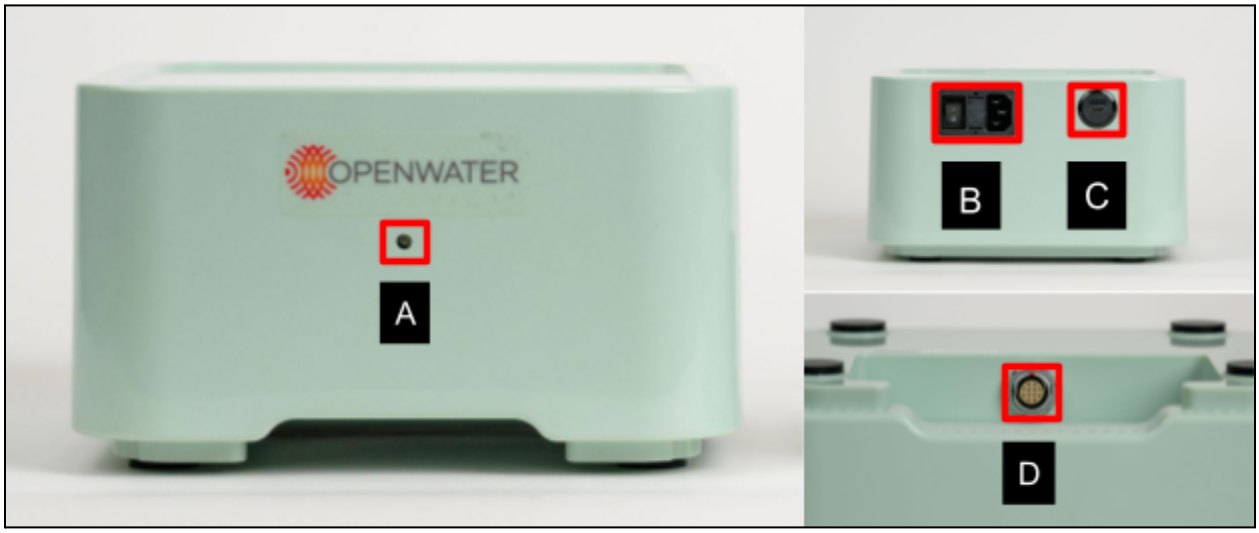



*Figure 3: Parts of the Console: (A) Status LED on front face of console, (B) Power switch and receptacle for the device, (C) USB port to connect to computer, (D) Transducer cable port.*

*Table 5: Console status LED indicator.*

| Color | Pattern | Meaning |
|---|---|---|
| **None** | Off | System is Off |
| **Green** | Solid | System is On and ready |
| **Blue** | Solid | System is On and sonicating |
| **Blue** | Pulsing | Error |

## Transducer

(a)

(b)

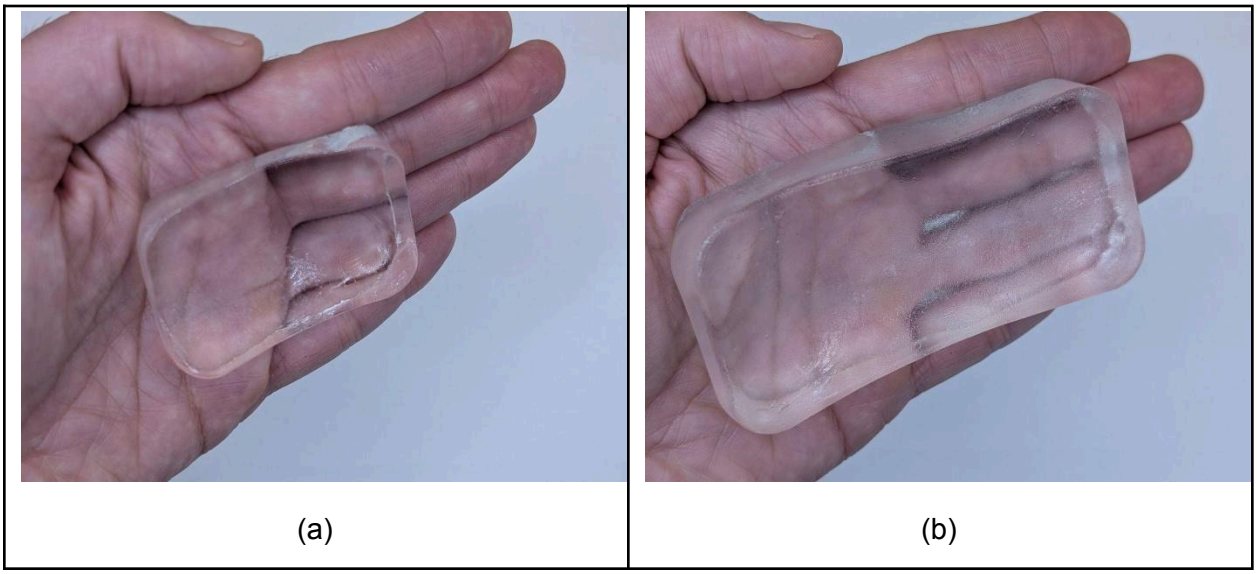
*Figure 4: The transducer uses a factory-configurable headset-style housing containing one or two transmit modules, e.g. 1x or 2x configuration (either 155 kHz or 400 kHz). a) the single transmit module transducer configuration ("1x"); b) the dual transmit module transducer configuration ("2x").*

NOTE: The transducer has an up-down orientation. Mechanical features prevent upside-down assembly, and a debossed arrow on the side of the transducer body indicates the "up" direction. The cable exits the transducer towards the subject's left ear when the transducer is worn on the forehead.

A soft padded foam frame on the transducer provides a comfortable, non-slip interface for when it is placed on the patient. The transducer also has a removable curved "veneer" attached to it and is used for transducer localization. Do not remove or damage the veneer.

NOTE: The transducer is not watertight. Never submerge the transducer in water. Doing so may lead to electric shock or damage the transducer.

## Coupling Pad



(a) (b)

*Figure 5: The coupling pad sits inside the transducer to help ensure good coupling is achieved between the transducer and patient. (a) the 1x coupling pad, (b) the 2x coupling pad.*

Coupling pads are non-sterile and composed of a water-based polymer hydrogel shaped to fit the faceted transducer faces and conform to curved surfaces (e.g., the forehead). The coupling pad is needed to:

- Eliminate air interfaces between ultrasound probes and skin
- Reduce acoustic impedance mismatch
- Improve transmission of ultrasound energy

*Table 6: Hydrogel coupling pad specifications.*

| | **Description** |
|---|---|
| Density | 1.12 ($10^3$ kg/$m^3$) |
| Speed of sound at $35^0$ C | 1520~1620 (m/s) |
| Hardness | 00 <40 |
| Peeling off strength | 220g |

NOTE: When using coupling pads, open packaging carefully at marked areas to avoid tearing or damaging the pad.

## Additional Hardware Requirements

### PC

A PC is required to download and run the Open-LIFU application.

**Recommended PC specifications:**

1. Graphics Card: NVIDIA CUDA 11+
2. Operating System: Windows 11+
3. Network card
    a. Recommended, but not required for installation
    b. Optional for Cloud Services
4. Storage: At least 5GB of free disk space

### Android Phone

To capture the photocollections, a mobile device with a camera is required for transducer localization. Official support is provided for Google Pixel phones, and the photogrammetry app has been tested on the Pixel 5, 7, 9, and 10. Additional phones that meet the minimum specifications should also work, e.g. Samsung Galaxy A16 or Motorola moto g, however full compatibility cannot be guaranteed. Developers are encouraged to visit Openwater's Discord for help with non-supported phones.

**Minimum phone specifications:**

1. Android phone running Android 14 or newer.
2. Dual Camera: 12.2 MP, f/1.7, 27mm (wide), 1/2.55", 1.4µm, dual pixel PDAF, OIS 16 MP, f/2.2, 117˚ (ultrawide), 1.0µm
3. Accelerometer, gyro

# Open-LIFU Software Development

## Overview

The Open-LIFU software stack comprises five coordinated layers: embedded firmware, a hardware sdk, a Python package of core sonication planning algorithms, a Slicer extension comprising the graphical UI modules, and a top-level executable desktop application. Each layer is modular and open-source, allowing for extensibility and integration, and the middle three layers can each be directly accessed through compatible programs for testing and development.

## Software Architecture

The software architecture is laid out as follows:

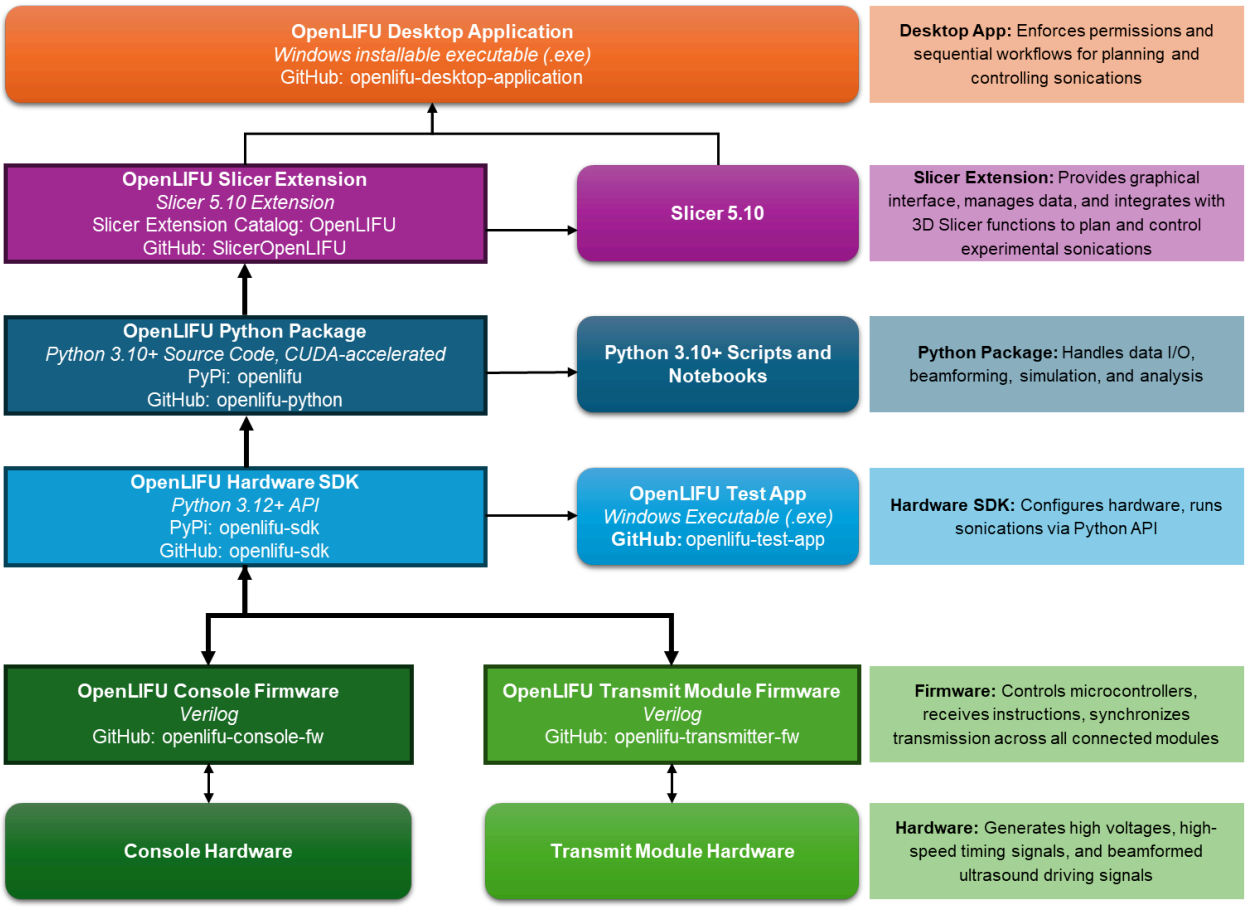



*Figure 6: The Open-LIFU software architecture is a five-layered stack comprising embedded firmware, a hardware SDK, a Python package for core sonication planning algorithms, a Slicer extension for graphical UI modules, and a top-level executable desktop application.*

## Open-LIFU Desktop Application

The Open-LIFU Desktop application is an executable application that provides a guided workflow for planning and executing sonications. This application assigns user-specific permissions and workflows and is the best way to prevent accidental system misconfiguration. This can be found in the openlifu-desktop-application repo on GitHub. The desktop application is a custom build of the open-source medical imaging program 3D Slicer with the Open-LIFU extension installed.

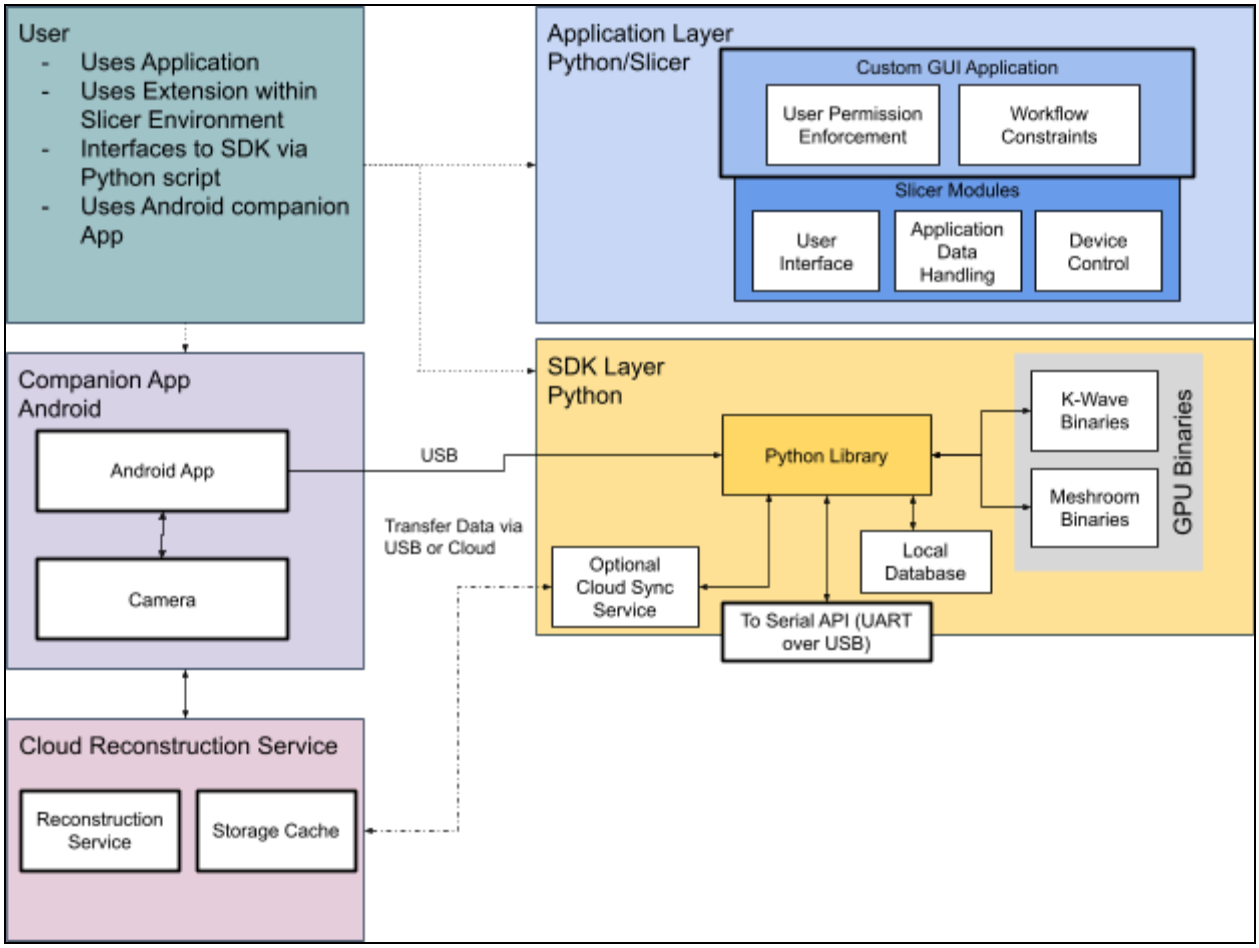



*Figure 7: The desktop application provides users with a guided workflow for planning and executing sonications.*

## Open-LIFU Slicer Extension

The graphical UI elements for configuring and controlling the system that the Desktop Application uses are found within the Open-LIFU Slicer Extension. For advanced users and developers, the underlying modules of this extension can be accessed directly from 3D Slicer by installing the Open-LIFU extension from the Extension Manager, or by using the SlicerOpenLIFU repository found on GitHub. This allows for the experimental modification of different steps in the sonication planning process and incorporation of external Slicer module functionality. Users must take care when using these modules, as many of the guardrails found in the Desktop application are not present or activated by default. In general, the 3D Slicer modules provide a graphical interface to functionality found in the lower-level Python package, but some graphically specific functions are only available in the modules.

## Open-LIFU Python

The core data classes and methods are all written in Python and are in the openlifu-python repository on GitHub. Some computationally intensive methods call GPU-accelerated binaries. These classes can be used directly by experienced developers to plan and control sonications and are fully customizable. Many of these classes are provided as abstract base classes with

one or more example-implemented subclasses, serving as templates for adding customized classes with more advanced and experimental features.

To get started, navigate to Open-LIFU GitHub repo and follow the installation instructions to install the necessary tools.

Python Package

- Repo: openlifu-python
- Framework: Python
- Function:
    - Orchestrates data management, beamforming computations, acoustic simulation, and field analysis
- Platform Support: Windows 11+, Linux
- Extensibility:
    - Open-source and modular (Python 3.10-3.12)
    - Integrates easily with NumPy, Pandas, Jupyter, and other scientific tools
    - API documentation can be found in openlifu-python/docs

Next, it is recommended to get familiar with and work through the different tutorials:

*Table 7: Summary of examples and code that users can use for getting started.*

| Repository | Description |
|---|---|
| Tutorials | Intro to the Open-LIFU python<br><br>Contains a series of interactive guides designed to demonstrate the functionality and usage of the openlifu library. This includes covering the basics of initializing the library and creating fundamental objects within the openlifu framework, demonstrating how the library connects to, queries, and interacts with databases, and generating analytical solutions and evaluating the results. |
| Test App | Python-based graphical user interface (GUI) designed for the validation and testing of Open-LIFU (Low-Intensity Focused Ultrasound) hardware. |

## Hardware interface SDK

While the Python package provides comprehensive planning tools, the hardware interface has been split into its own repository, so that applications requiring direct control of the hardware without the sonication planning tools can be built using only a lightweight interface layer. This layer runs on the PC and provides device control and data acquisition.

- Repo: openlifu-sdk
- Framework: Python
- Function:
    - Data management
    - Sonication pre-planning (“virtual fitting”)

    - Photogrammetric reconstruction using Meshroom
    - Beamforming computation and acoustic simulation using k-Wave
    - Beam field analysis
    - Hardware configuration and control
- Platform Support: Windows 11+, Linux
- Extensibility:
    - Open-source and modular (Python 3.10-3.12)
    - Integrates easily with NumPy, Pandas, Jupyter, and other scientific tools

### Python Library Installation (Optional)

Direct access to the underlying logic is available via the openlifu-sdk library.

1. Install Python (3.12)
2. Clone openlifu-sdk
3. Create a virtual environment

```
C:\Users\<uname>\AppData\Local\Programs\Python\Python311\python.exe -m venv env

.\env\Scripts\activate
```

4. Install dependencies

```
pip install -e .
```

## Embedded Firmware Layer

The Embedded Firmware Layer includes all firmware responsible for the low-level operation of the system, running on the console and transmit modules.
To work within this layer:

- **Development Environment:** The firmware is developed in C. Developers use **Visual Studio Code** to:
    - Download the project from Git.
    - Open the project within the Visual Studio Code.
    - Make necessary firmware modifications.
- **Technical Stack:** The firmware utilizes the **STM32Cube HAL** and **CMSIS** libraries.
- **Development Standards:** The code applies **MISRA C guidelines** where practical to enhance safety and portability. Code is specifically structured for:
    - **Readability** (clear naming, small functions, single responsibility).
    - **Safety** (defensive programming, bounds checking, and explicit initialization of peripherals and state).
    - **Testability** (hardware abstraction, minimal global state).

## Local Data Storage

In addition to the code is the representation of structured data on disk. While the Python code and Slicer modules can handle any data that is well-constructed into the appropriate class, a Database class provides convenient organization of and access to protocols, subjects, and

transducers, each with nested information. The system data is stored locally in a file tree on disk in a combination of JSON format for structured data and various other file formats for binary data.

The GUI application accesses the data from within such a database, while certain research applications may need to access data objects stored elsewhere via some of the lower levels of access.

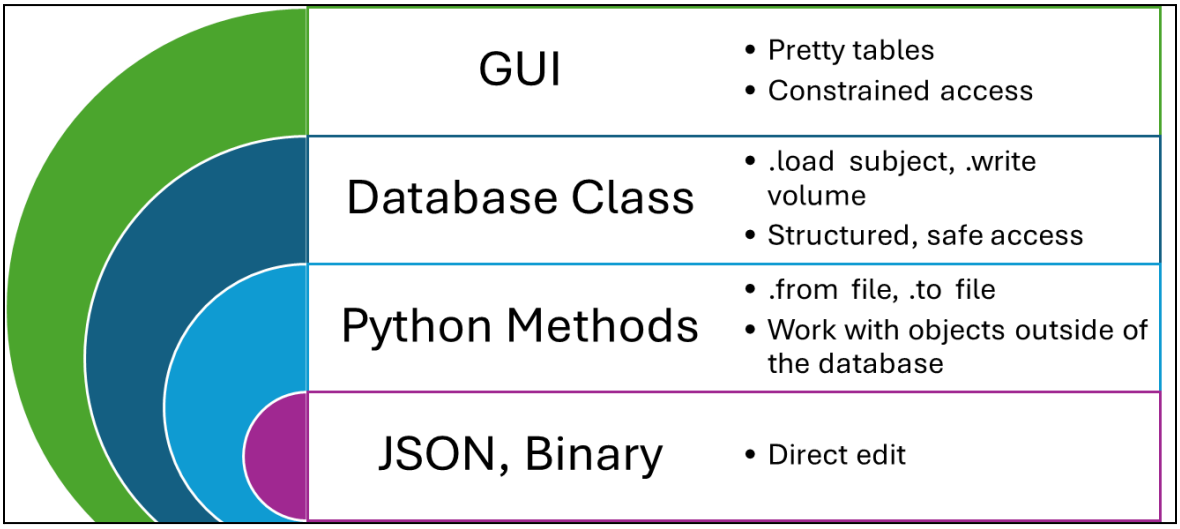



*Figure 8: Each layer is built on top of the other.*

An example of what the database file structure looks like is shown here (items with angle brackets represent instance IDs):

```
protocols/
  <pid>/
    <pid>.json
transducers/
  <tid>/
    <tid>.json
    <tid>.surf.obj
    <tid>.body.obj
subjects/
  <sid>/
    volumes/
      <vid>
        <vid>.json
        <vid>.nii
    sessions/
      <ssid>/
        photocollections/
        photoscans/
        solutions/
        runs/
users/
  <uid>/
    <uid>.json
```

### Cloud Database:

*The cloud database is an active roadmap feature and is in the process of being developed.*

### Android Application

Alongside the Open-LIFU application is a companion Android app used to complete the capture of photographs for photogrammetric reconstruction.

### Tools Used

| Category | Tools / Platforms |
|---|---|
| Version Control | Git + GitHub |
| CI/CD (optional) | GitHub Actions (under development) |
| IDEs | VSCode |
| Documentation | Markdown, docx |
| Testing | Oscilloscope, logic analyzers, unittest |

### Test & Validation Utilities

These tools are used in development, manufacturing, and QA workflows.

- Repo: openlifu-test-app
- Framework: Python + PyQt6
- Function:
    - Set high-voltage rails and read back the values
    - Turn on/off the transmit module(s)
    - Perform basic testing to ensure successful commands can be sent to and received by the transmitter
    - Toggle the trigger and change the target xyz coordinates to steer the beam
    - Upgrade firmware

## Transmit Modules

Depending on the configuration, there may be one or more transmit modules inside the transducer. These comprise a PCBA affixed to a PZT ultrasound transducer, encased in plastic, with a heatsink on the back.

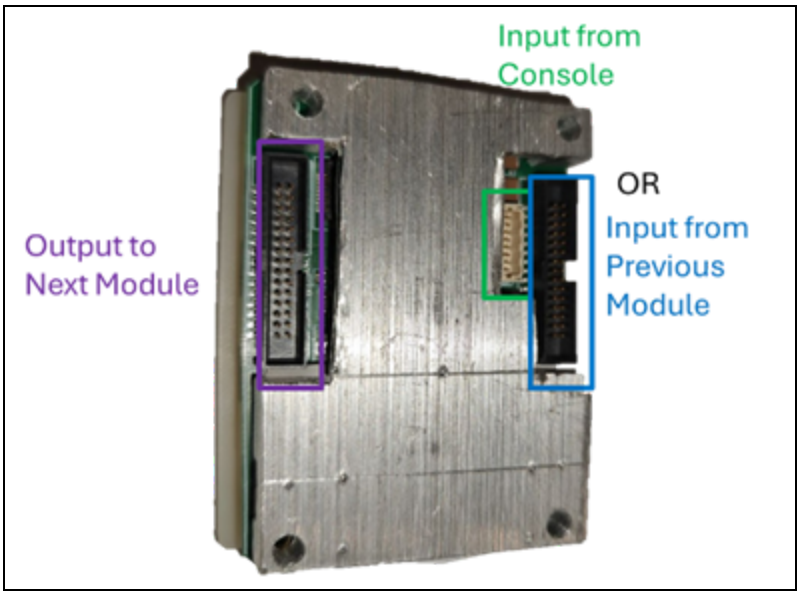



*Figure 9: Transmit module connection points.*

If reconfiguring the transducer, note that only the console input or the input from a previous transmit module should connect to one of the adjacent connectors; never both. Ribbon cables should be used to connect the output side of one transmit module to the input side of the next. Refer to the Water Tank Testing section prior to using a reconfigured transducer.

*NOTE: All Open-LIFU transducers and transmit modules are factory calibrated. Reconfiguring or modifying any transducer immediately voids the factory calibration, requiring users to recharacterize the transducer to ensure it complies with all necessary application-specific requirements.*

## Transducer Logical Architecture

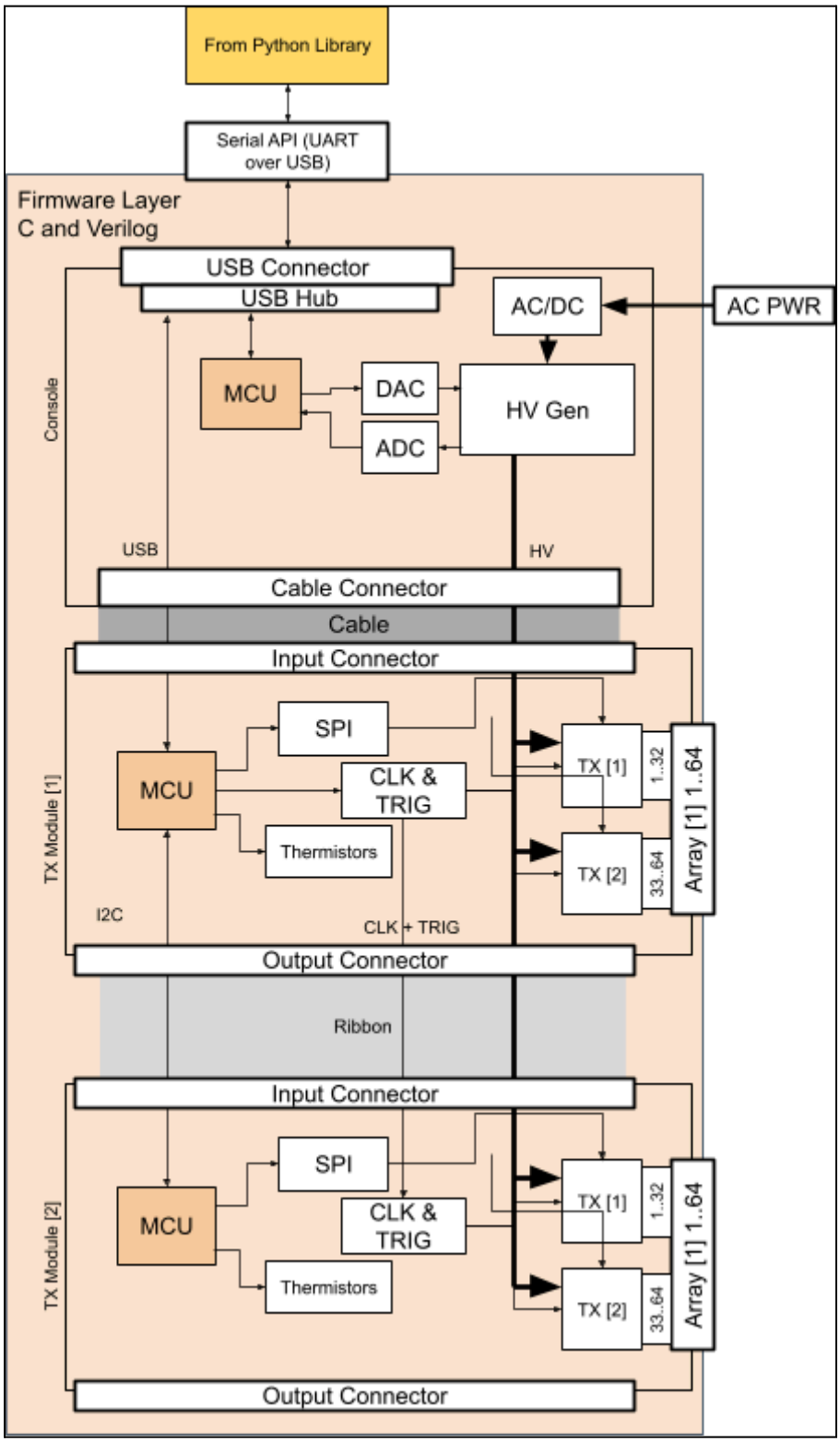



*Figure 10: Transducer signal diagram.*

## Transmit Module Wiring Diagram

While all transmit modules contain the same hardware, the console connects to only the first module in a chain via a custom cable. That module is connected via a parallel ribbon cable to any downstream modules, over which it passes commands via I2C and generates signals that are broadcast to all the modules for coordinating synchronized transmission of ultrasound. Downstream modules take input and output both via ribbon cable, allowing for near-arbitrary levels of daisy-chaining, if the HV power supply can provide sufficient power to all connected modules.

### Configured as first module (active lines Magenta)

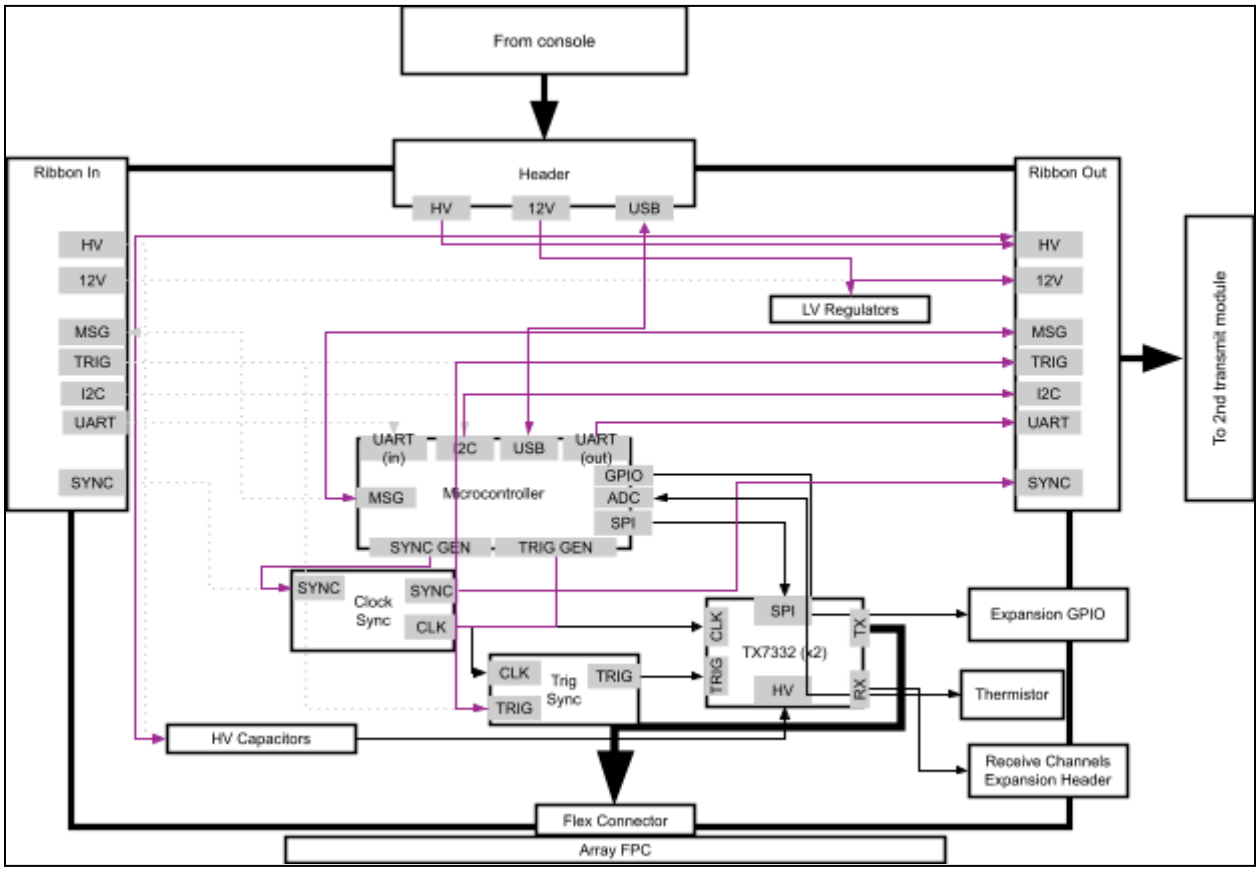



*Figure 11: A single transmit module wiring diagram.*

## Configured as middle module (active lines blue)

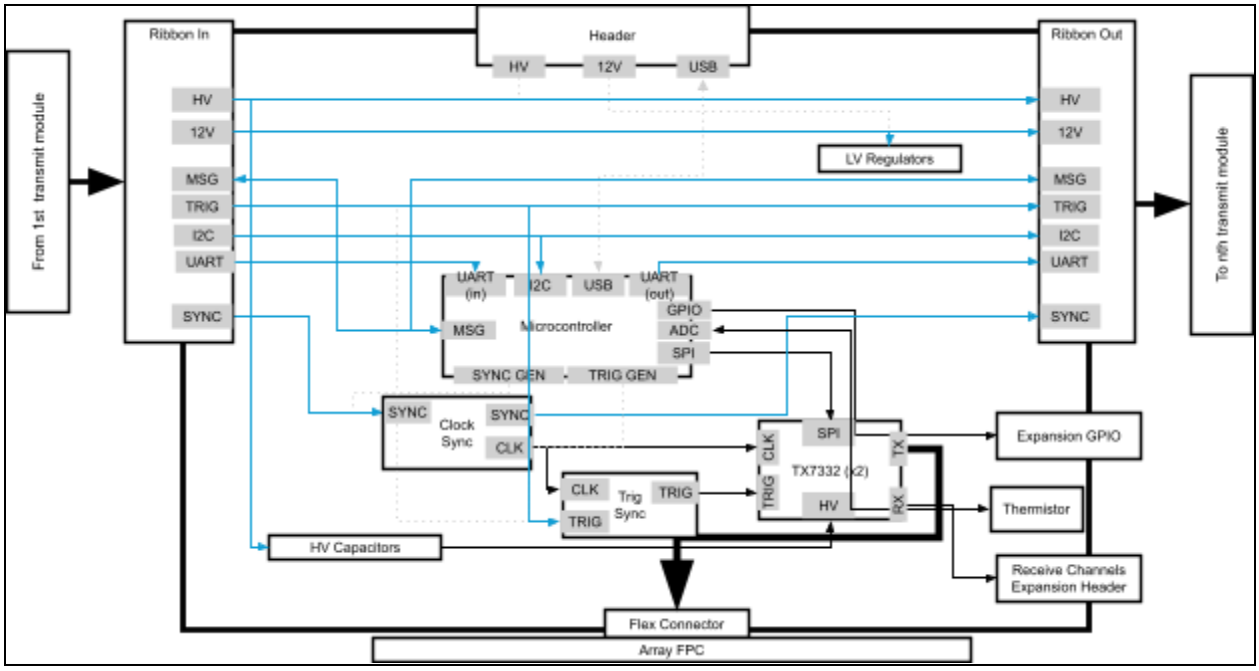



*Figure 12: Wiring diagram for daisy-chaining multiple transmit modules in series.*

## Configured as last module (active lines green)

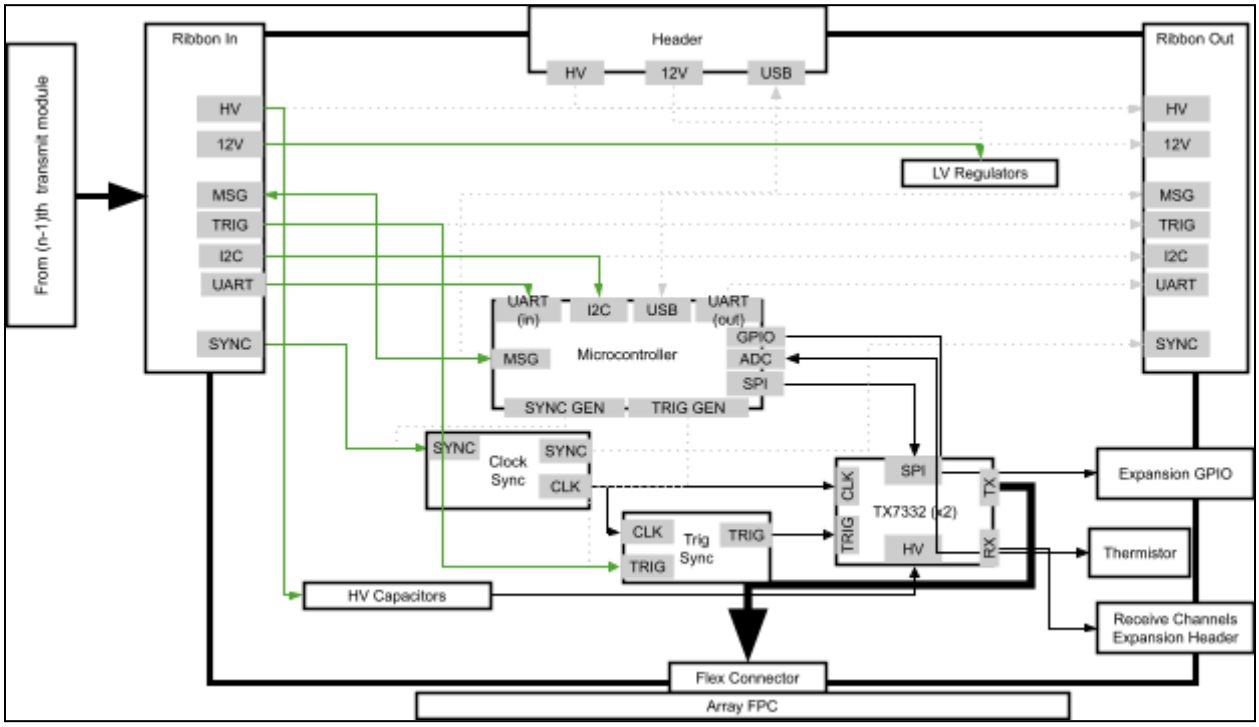



*Figure 13: Wiring diagram for the last transmit module in the series when daisy chaining more than one (1) together.*

# Open-LIFU Hardware Development

## Overview

All hardware developed using Openwater's base platform falls under Creative Commons - ShareAlike 4.0. The hardware is separated into the following 1x and 2x transmit module configuration CAD assemblies. The part numbers are provided in the tables below.

## CAD Files

Released design files for Open-LIFU are stored in the Mechanical and Electrical repositories on GitHub. Released CAD files can be downloaded by visiting openlifu-mechanical or openlifu-electrical.

## Mechanical

- Provides 3D mechanical parts and assembly models for Open-LIFU devices, including transducers and housings.
- Useful if users need physical dimensions, mechanical constraints, or to integrate with custom enclosures.
- Good for: Mechanical CAD & physical integration requirements.

### Open-LIFU Mechanical Design Files

*Table 8: Open-LIFU mechanical design files with their associated part numbers.*

| Part Name | Part Number |
|---|---|
| LIFU Console w/ Power Cable and USB Cable | 700-00012 |
| 1x Transducer Housing | 700-00018 |
| 2x Transducer Housing | 700-00021 |
| LIFU Transducer Strap | 700-00022 |
| Transmit Module | 700-00039 (158 Hz) or 700-00040 (400 Hz) |

### Electrical

- Contains PCB designs for the Open-LIFU platform (Console, Transmit modules, etc.).
- This is where users will find board-level schematics and layout sources for building or modifying hardware.

- Good for: Getting actual electronics hardware build files.

### Open-LIFU PCBs

*Table 9: Open-LIFU electrical design files with their associated part numbers.*

| Part Name | Part Number |
|---|---|
| Console Board | 710-00025 (Board Design)<br>720-00011(Schematic) |
| Transmit Board | 710-00015 (Board Design)<br>720-00001 (Schematic) |

# Slicer Open-LIFU: Research Use Only (Advanced Users)

## Install Slicer with the Open-LIFU Extension

To access a robust set of tools, install Slicer and the Open-LIFU extensions directly. This will allow the use of external slicer functions to modify the volumes/meshes/transformations, etc., used for sonication planning.

### Download Slicer

1. Download the latest stable release of Slicer from https://download.slicer.org/
2. **On Windows:** During installation, ensure that there are no spaces in the installation path

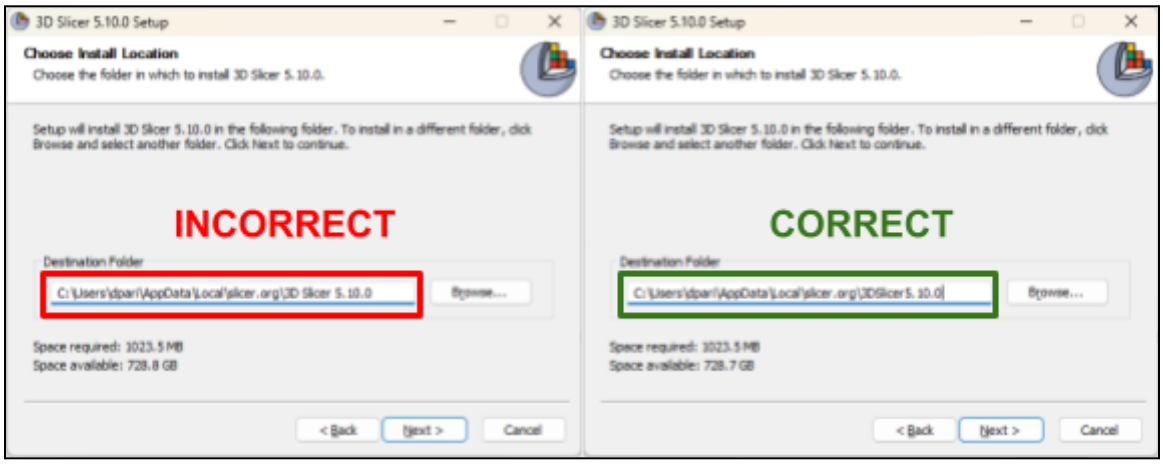



*Figure 14: Ensure no spaces are used when specifying the destination folder.*

### Install The SlicerOpenLIFU Extension

The SlicerOpenLIFU extension exists as an extension within Slicer itself.

1. Launch Slicer
2. Navigate to "View" in the top left corner.
3. Click on "Manage Extensions."
4. Type in "Open-LIFU" in the search bar. Locate the Open-LIFU extension and click "Install."

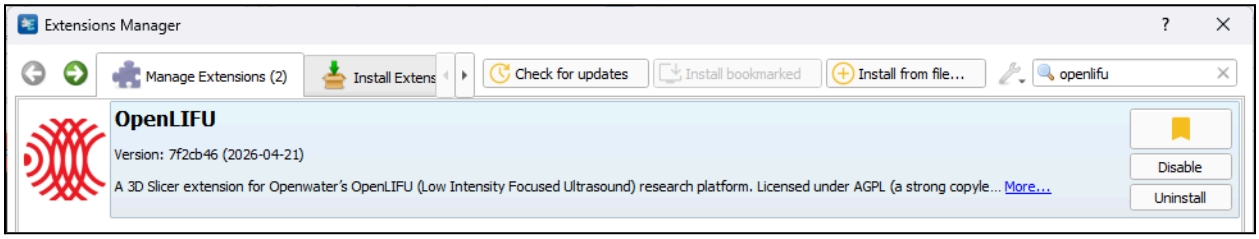



*Figure 15: Use 3D Slicer's extension manager to simplify the Slicer Open-LIFU installation.*

5. Once the installation is complete, restart the application for the extension to be enabled.
6. If an older version of the extension is needed, please go to the SlicerOpenLIFU repository and follow the instructions to download and install the extension. Note that users must first uninstall any previous versions of the Open-LIFU extension and restart the application before installing a new one.

To perform offline mesh reconstruction instead of using Openwater's available Cloud Services, users need to also install Meshroom and add it to the system PATH. Follow the instructions here to download and configure Meshroom for local photoscan reconstruction.

# Slicer Modules

*Only advanced users should use the Slicer modules directly, as fewer safety and permission restrictions apply to system configuration and operation. Because the underlying data objects are more explicitly exposed and mutable, it is possible to configure the system to operate outside of bounds for system safety, integrity, and performance.*

The Slicer Open-LIFU Extension exposes additional functionality to the user for each of the modules that is otherwise disabled in the Open-LIFU Desktop Application.

*Table 10: Overview of the Slicer Modules with accompanying details about each module.*

| Module | Description |
|---|---|
| Data Management | Allows users to setup and configure database directories for accessing user and subject data that is then used during a session. |
| Sonication Protocol | Users can create custom sonication protocols using the Protocol Configuration Wizard. The wizard can be used to create, and review or edit existing protocols. |
| Pre-Planning | Used for identifying and placing targets within a specific region of the body, i.e. the brain, and suggests transducer placement options to ensure the target lies within the focused beam pathway. |
| Transducer Localization | Uses photogrammetry to map the virtual transducer to the physical transducer by using corresponding anatomical landmarks to co-register an individual's pre-procedural MRI with their photoscan. |
| Sonication Planning | All necessary sonication inputs are used to compute the sonication solution and creates a visualization of the activation volume. The simulation helps to ensure the target lands within the focal spot. |
| Sonication | Monitors the progress and delivery of the full treatment to the prescribed target, usually consisting of multiple sonications. |

Users can navigate the Slicer modules using one of two workflow modes:

**Prescribed Workflow (Session)**

Loads a predefined session containing specific details about Sonication Protocols, Transducers, Volumes, and Targets. These parameters are locked throughout the workflow to ensure consistency and guide the user through a prescribed path.

**Open Workflow (No Session)**

Provides maximum flexibility by allowing users to input and modify data independently within each module. This mode is ideal for testing specific features, custom workflows, or iterative functionality where parameters may need to change between stages.

## Data Management

The Open-LIFU Data page introduces a section to view the data objects associated with a session. The user may define (create) a session by selecting a subject, sonication protocol, transducer, MRI volume, fiducial landmarks, and photoscan. These all then get saved to the session.

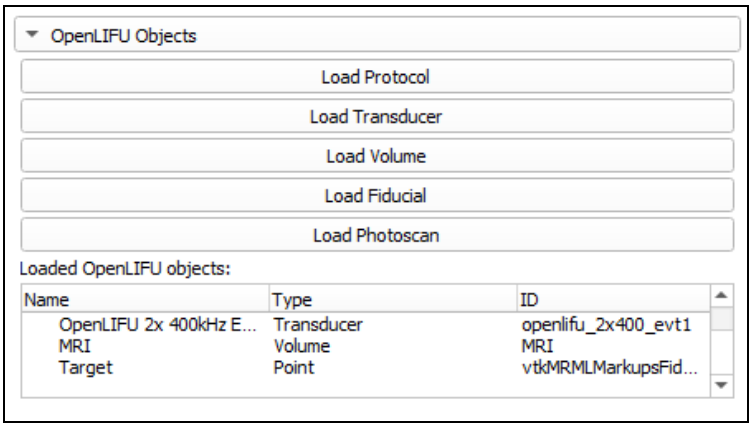



*Figure 16: User-defined data management inputs.*

# Sonication Protocol

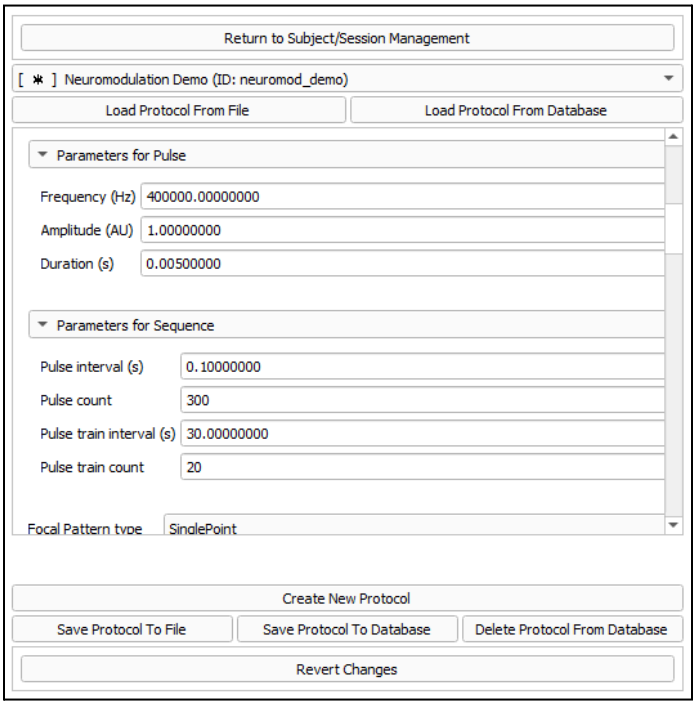



*Figure 17: Users can input sonication protocol parameters to customize protocols for a desired use case.*

The Sonication Protocol editor matches the functionality in the main application. No additional controls are exposed in the Slicer module.

# Pre-planning

The Pre-planning page serves as an interface to place/remove a target and identifies several appropriate transducer placement and orientation options based on the selected target location. When the user is not proceeding with a session, the input objects to the Virtual Fitting algorithm may all be selected among the available objects that were individually imported from the previous Open-LIFU Data page. Volumes can point to any volume imported into Slicer (allowing for compatibility with DICOM, nrrd, processed volumes, etc), and Targets can be imported as

any markup point file type. The protocol and transducer files are Open-LIFU-specific and must be loaded from specified files provided to the user.

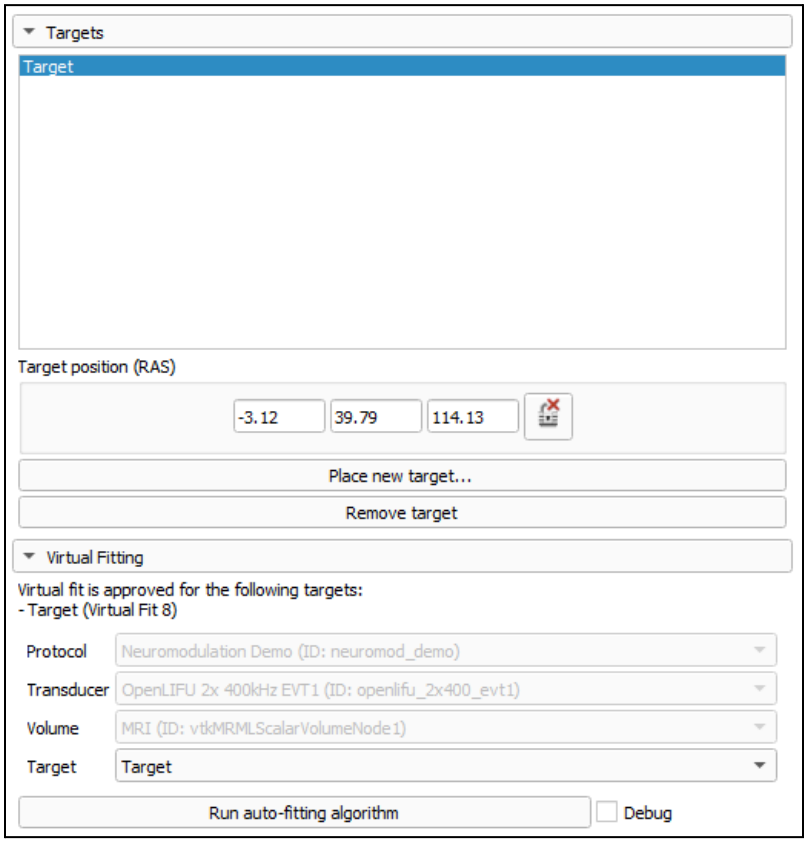



*Figure 18: Multiple targets can be placed and locked. The desired target can then be selected for determining the initial transducer placement.*

## Transducer Localization

The Transducer Localization module is the main interface for users to co-register a photogrammetry scan to a segmented volume, e.g. MRI. The user may undergo this process through multiple methods, including Online and Offline reconstruction. The user may also manually upload a photocollection or a photogrammetry scan from a local file system through the “Browse” button. The photogrammetry scan **MUST** be in an .obj file format. The manual upload feature in this section is native to Slicer Open-LIFU.

### Continuing WITH a session:

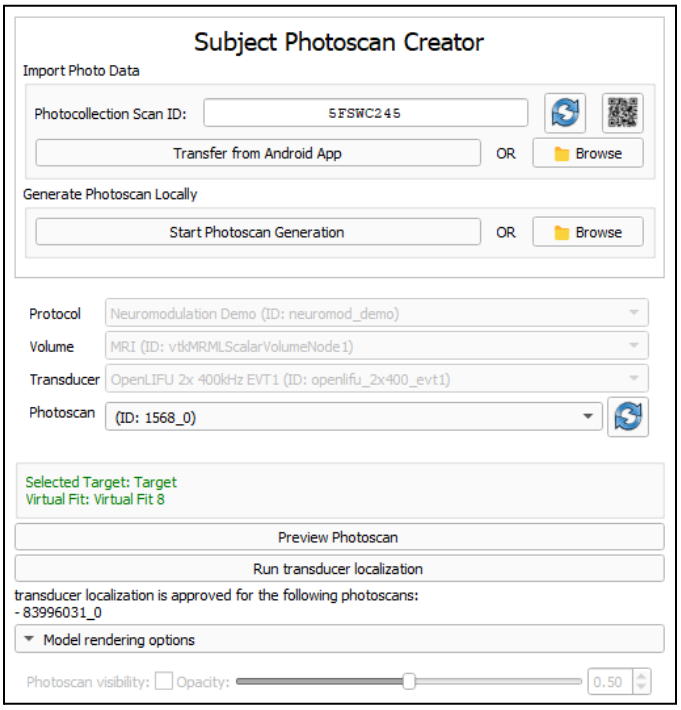



*Figure 19: Click the QR code and scan it using the photogrammetry phone app to streamline entering the data into the phone app.*

### Continuing WITHOUT a session:

**IF the user has decided to continue without selecting a session** and has manually uploaded multiple data objects individually in the “Open-LIFU Data” section, the user is able to select from those objects in this section. The user can instantly alternate between various imported data objects and test the transducer localization accordingly.

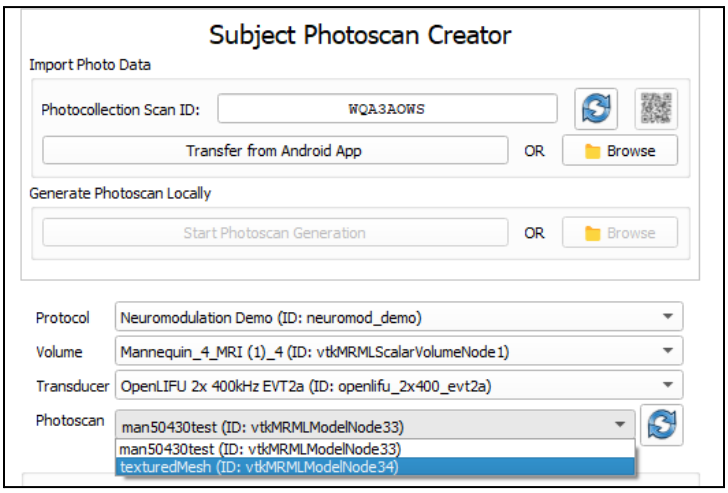



*Figure 20: Use this option when testing multiple session inputs at the same time.*

# Sonication Planning

The Sonication Planning module has fields to choose from, loaded Protocols, Transducers, Volumes, and Photoscans when running beamforming and simulation.

**Continuing With a Session:**

If **the user is using a session,** the data objects associated with that session will be present and greyed out in the data objects' designated sections. Once selected, the user may proceed with computing the sonication solution according to their selected data objects.

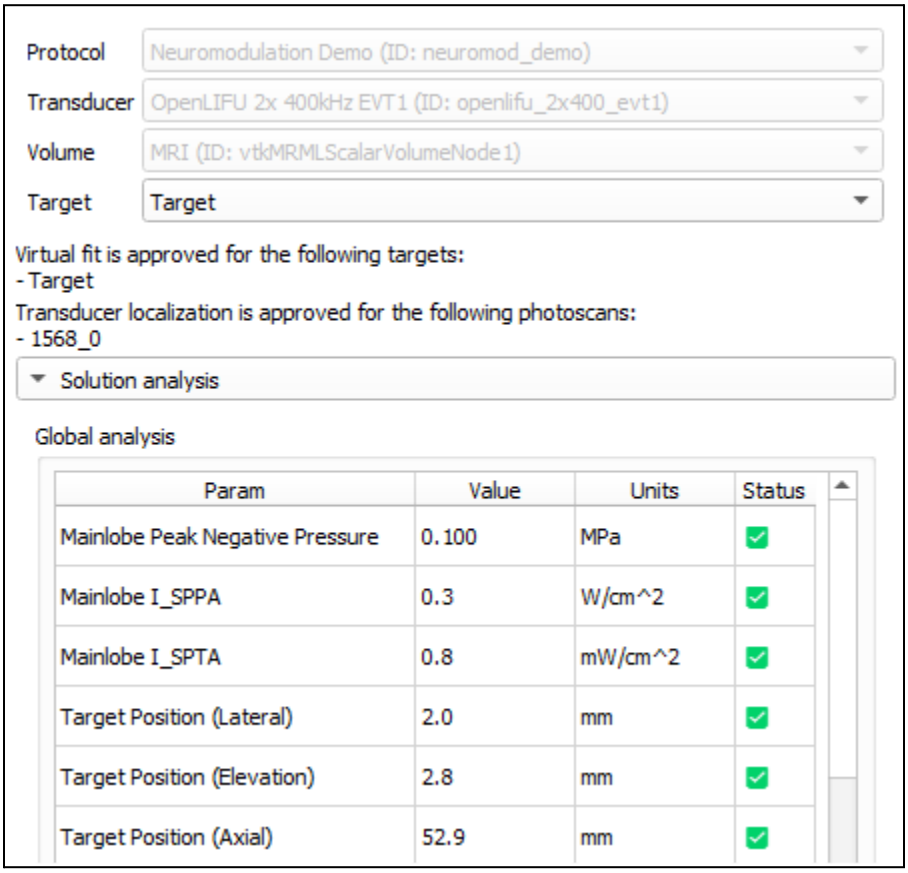



*Figure 21: When all parameters have a green checkbox in the status column then the sonication protocol has passed the system's internal checks.*

**Continuing Without a Session:**

**IF the user has decided to proceed without a session**, they may select from their previously uploaded data objects. This will allow the user to alternate between data objects and instantly test the sonication solution on various objects.

# Sonication

When the device is powered on and fully connected to the headset, the connection status will be indicated in the user interface.

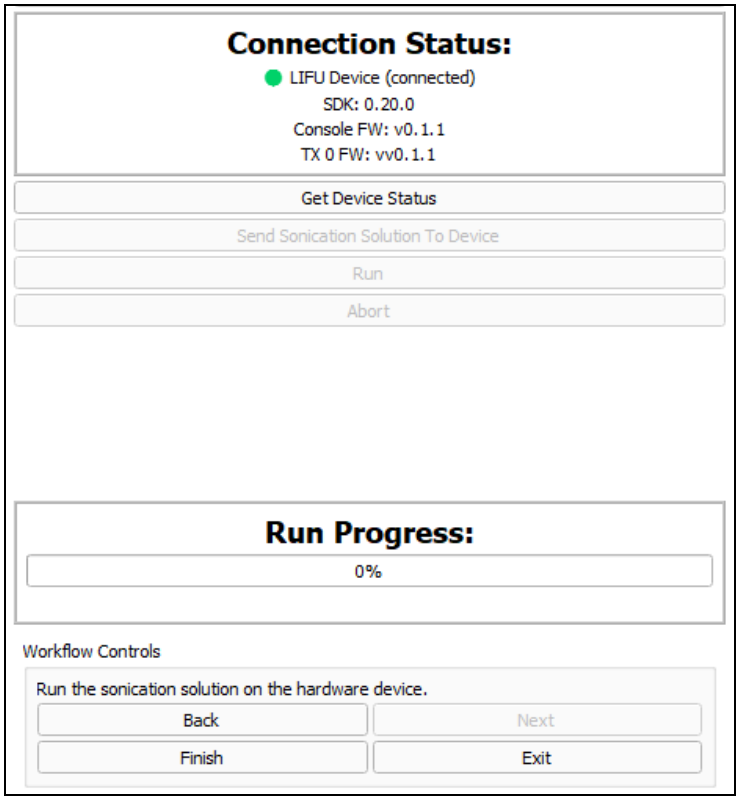



*Figure 22: When the device is connected and running, a progress bar will indicate when sonication is complete.*

## WARNING:

Running Slicer Open-LIFU does not restrict or prevent accidental sonications - a sonication solution can still be sent to the transducer even if the parameters fall outside the safe or acceptable zones. Users are responsible for knowing and understanding the different sonication protocol parameters, and the effects they may have on the subject.

The Sonication Module issues a warning for solutions that fail analysis checks, it does not always prevent execution[5]. Proceeding with these solutions poses a significant risk of hardware damage or subject injury and should not be attempted.

A progress bar tracks completion once the solution is sent to the device.

[5] If voltage or duty cycle values fall outside the predefined thresholds then the software will produce an error and the sonication solution **will not** be sent to the device.