# ER-00014-RevA Open-Motion_ Documentation for Developers

Last Modified : 04/20/2026 09:49:03 AM

## Change Approval History

| | |
|---|---|
| **Change Number** | ECO-000233 |
| **Effective Date** | 05/01/2026 09:12:40 AM |
| **Approval Date** | 05/01/2026 09:12:40 AM |
| **Approvers** | George Vigelette (Principal Software Engineer), Christopher Bush* (Senior Staff Quality Engineer), David Paribello, Madhumita Srikanth (Senior Quality Engineer) |
| **Associated Items** | From ER-00014 (A - In Production) to ER-00014 (A - In Production) |

* An asterisk next to an approver's name indicates they proxy approved a change.

# Open-Motion*

Documentation for Developers


Document ID: ER-00014
Revision: A
Release Date: 04/15/2026
Authors: David Paribello, Ethan Head, George Vigelette, Dan Blizinski

**RESEARCH USE ONLY – WARRANTY VOID IF ALTERED**

This device and documentation are intended strictly for laboratory research and development; clinical or diagnostic use is prohibited. CAUTION: Disassembly may expose the user to hazardous Class 3 laser radiation, capable of causing permanent eye injury. A one-year limited warranty applies only to manufacturer's defects; any modification, disassembly, or installation of unauthorized software/firmware that alters device performance immediately voids all warranties. By building, creating, or modifying this hardware or software, you assume all risks. Openwater disclaims all liability for any harm, injury, or damages incurred, and the user agrees to indemnify Openwater against any claims arising from such modifications.

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

# Introduction

## Overview

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

The Open-Motion system is a modular, open-source platform designed for research use that leverages low-intensity near-infrared light to measure blood flow, blood volume, and micro-motions deep within tissue. The system consists of a console and one or more sensor modules, each equipped with 8 camera sensors and a processing engine for on-board image processing, significantly reducing data transfer requirements. The Open-Motion console houses a laser system, safety circuits, and a sensor module hub, allowing for a standard PC to interface with the system over USB. The platform is highly configurable, supporting multiple sensor modules for diverse applications, and is controlled by PC software running on Windows, macOS, and Linux.

Open-Motion is fully open-source, with firmware, FPGA logic, and host-side software available under permissive and reciprocal licenses. This empowers researchers and developers to customize or extend the system for novel applications, develop their own acquisition and analysis pipelines, and integrate it into other imaging or biosignal frameworks. The software development kits (SDKs), and Applications are implemented in Python, providing an accessible and well-documented interface for control, data acquisition, and visualization.

The platform offers flexibility, portability, and cost-effectiveness while adhering to safety and compliance standards. The device is not FDA-cleared and is intended solely for investigational use by qualified researchers.

## 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 that all developers follow open-source standards, along with the appropriate regulatory standards for the region in which they develop. Contact your local authorities to get a complete list of local rules and regulations.

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

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

- **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.

## 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 under Creative Commons ShareAlike 4.0 license, 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**
- Code review is required to merge pull requests
- Releases in main will be tagged with the version numbers
- **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
    - 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 ISR code for minimal execution time and deterministic behavior
        - Enable compiler warnings (-Wall -Wextra) and treat warnings as errors where feasible
- **FPGA Logic**:
    - Written in **Verilog**
    - Uses consistent and documented:
        - Module interfaces
        - Clock and reset conventions
        - Internal signal naming
    - Additional Best Practices:
        - Use synchronous design practices; avoid gated clocks
        - Clearly define clock domains and crossing mechanisms (FIFOs, synchronizers)

- Prefer non-blocking assignments in sequential logic
- Parameterize widths, depths, and timing values
- Include reset behavior for all stateful elements
- Separate control logic, datapath, and interfaces
- Include assertions or simulation checks for protocol correctness
- Maintain synthesizable-only code in RTL (no simulation-only constructs)

### 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 3.12
  - 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

# 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 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, 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're working on a particular issue, let others know so you don't 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, feel free to reach out to the community through

- Discord
- GitHub
- support@openwater.cc

## 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-Motion System

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

1. **Console**: The Console includes a Laser, electronics, and optics, a power cable, and an on/off switch on the back. The Console should never be opened as it could expose the user to harmful laser radiation and electrical hazards.
2. **Sensor Module(s)**: Wearable sensor modules containing:
    - An array of camera sensors (configurable through software)

## Open-Motion Specifications

### Open-Motion Console

*Table 1: Open-Motion Console Specifications.*

| | |
|---|---|
| Operating Voltage | 100-240VAC, 50-60 Hz, 0.5 Amps |
| Connections | 1x USB-C (USB3.0)<br><br>1x SMA trigger (input)<br>1x SMA sync (output)<br><br>2x electrical ports (left / right)<br>2x optical ports (left / right) |
| 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 | 3lbs (1.36 kg) |

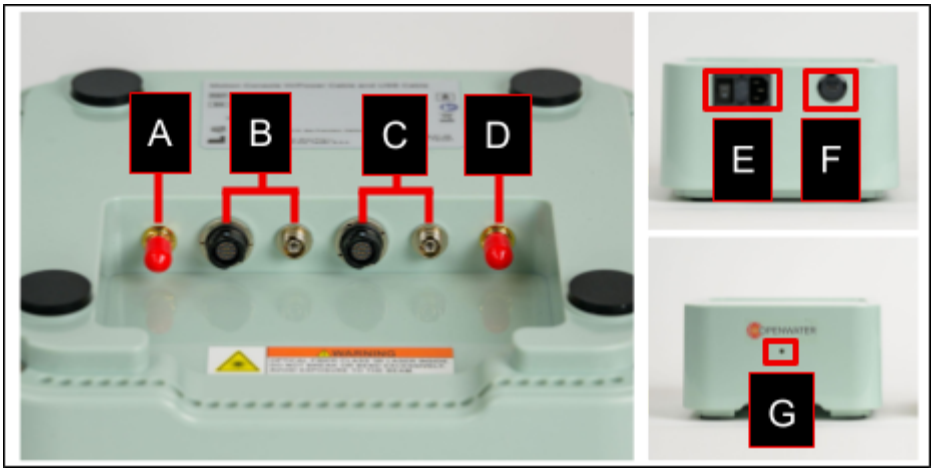


*Figure 1: Parts of the Console: (A) SMA trigger output port for synchronizing other devices with Open-Motion data, (B) Left Sensor Module electrical and optical ports, (C) Right Sensor Module electrical and optical ports, (D) SMA trigger input port for synchronizing Open-Motion with other devices (in development), (E) Power switch for the device, (F) USB port to connect to computer, (G) Status LED on front of Console.*

## Open-Motion Sensor Module

*Table 2: Specifications for a single Open-Motion Sensor Module. Each device uses two modules.*

| | |
|---|---|
| Image Sensors | 1/3.52" CMOS |
| Resolution | 1920 x 1280 |
| Frame Rate | 40 Hz |
| Pixel Size | 2.2 µm x 2.2 µm |
| Dimensions (w x h x d) | 2.1 x 1.6 x 2.5 inches (52.5 x 39.4 x 64.1 mm) |
| Cable Length | 6’ (2 m) |
| Weight | 0.5 lbs (0.2 kg) |

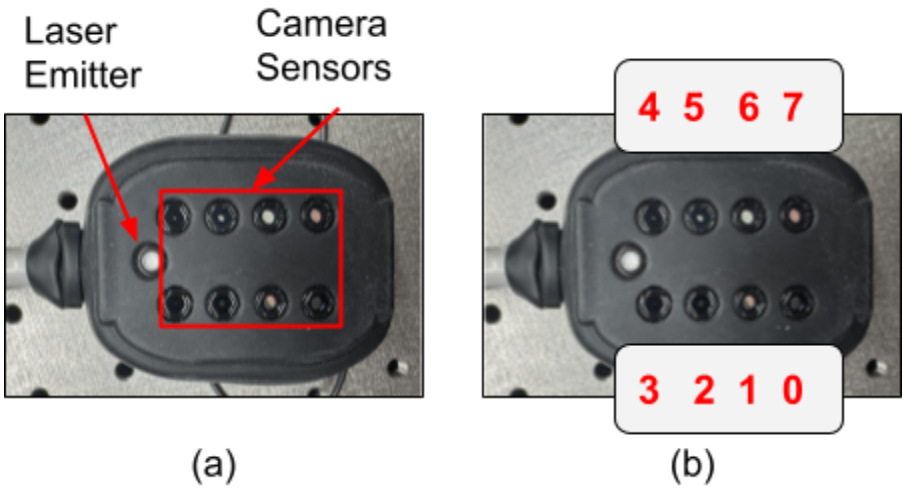


*Figure 2: a) Parts of the Sensor Module. b) Identifies how the cameras are ordered in the software.*

## Laser

*Table 3: Open-Motion Laser Specifications.*

| | |
|---|---|
| Laser Classification | Class 1 |
| Wavelength | 795 nm |
| Pulse Duration | 250-1000 µs |
| Pulse Repetition Rate | 40 Hz |
| Average Power | 12 mW |
| Energy per Pulse (at the delivery fiber tip) | 300-500 µJ |

# System Architecture

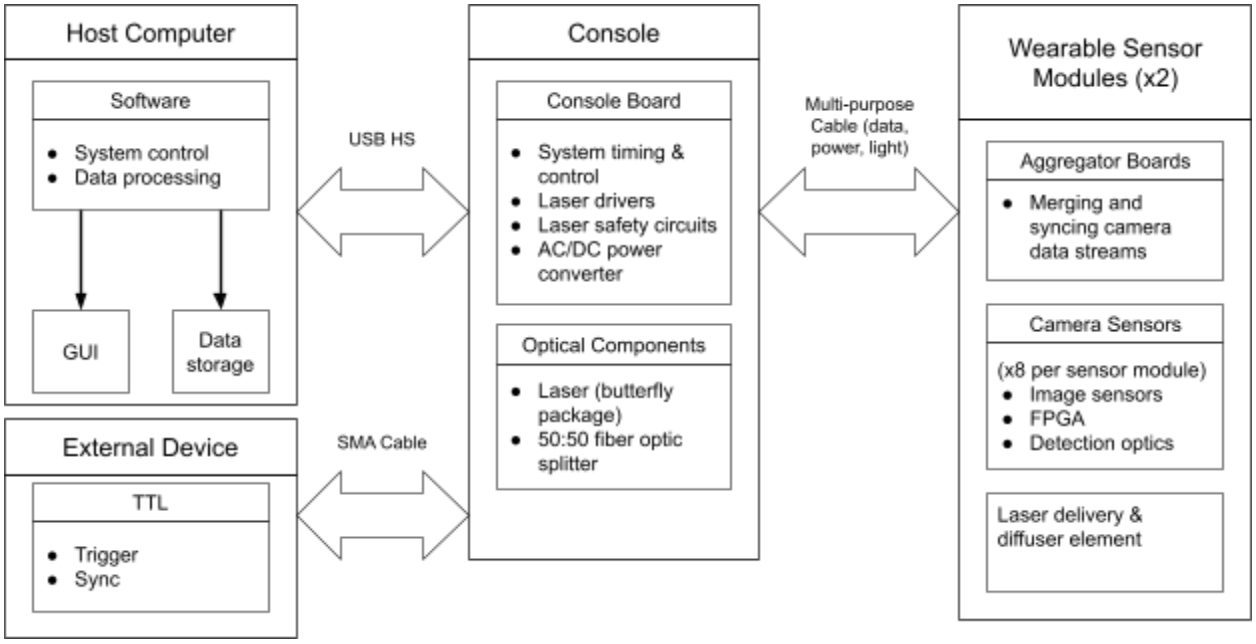


*Figure 3: General system architecture for the Open-Motion device. Note that the USB cable must be a USB A □ C and can only be sourced from Openwater (PN 100-00049).*

# System Signals

## Open-Motion Console

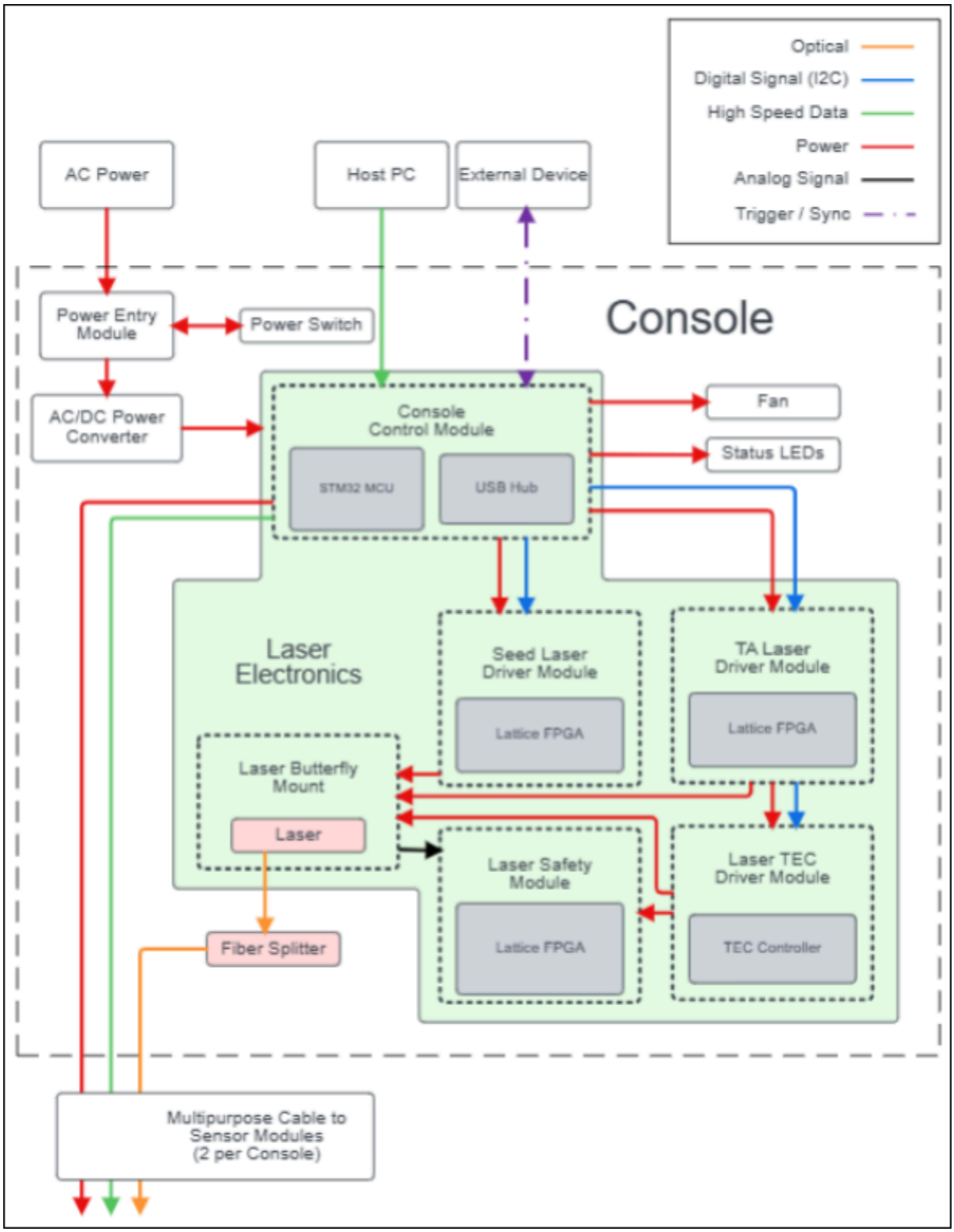


*Figure 4: Signal pathways within the Open-Motion console.*

## Open-Motion Sensor Module

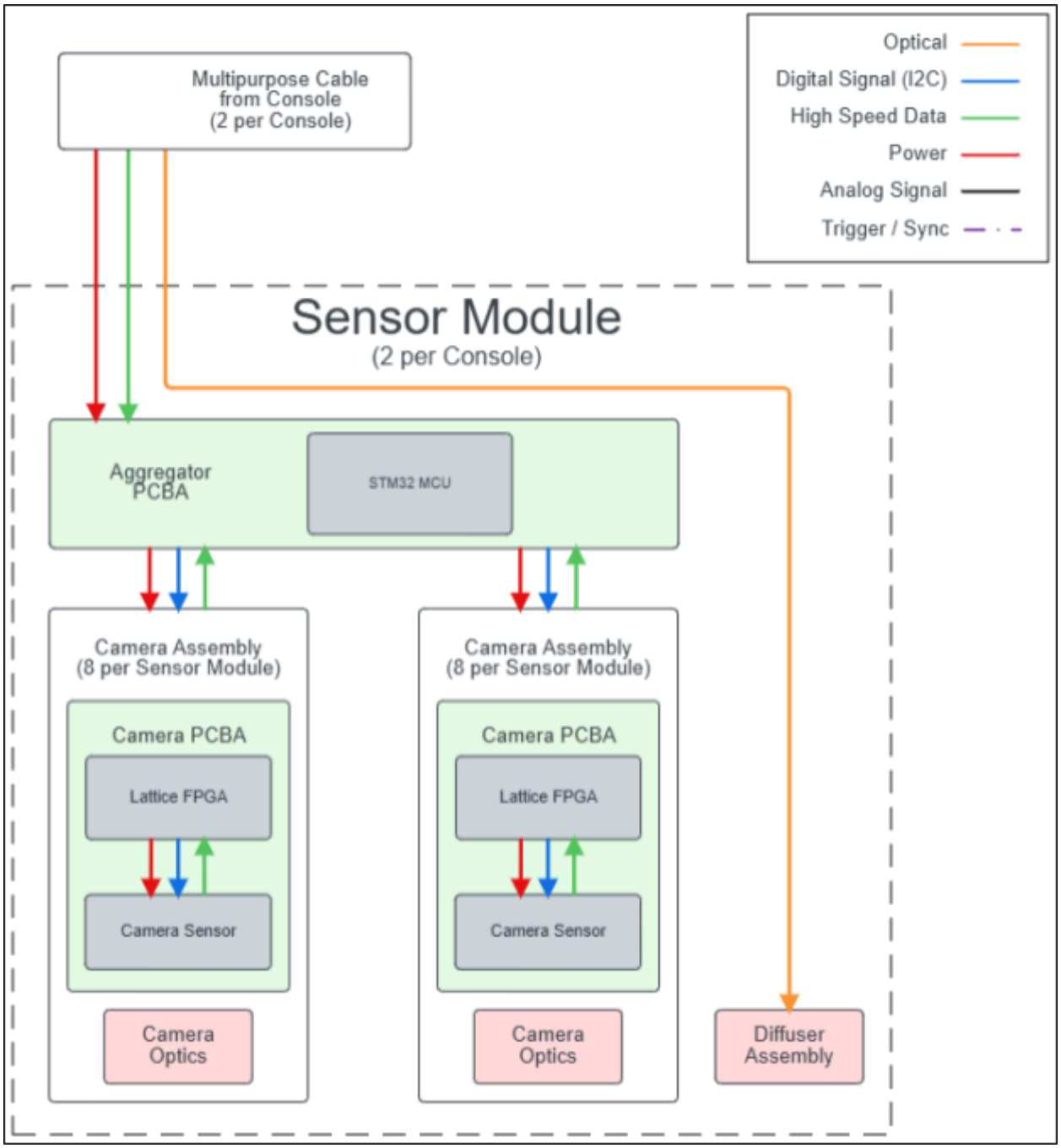


*Figure 5: Signal pathways within the Open-Motion sensor modules.*

# Open-Motion Software Development

## Overview

The Open-Motion software stack comprises three coordinated layers: embedded firmware, host-side software, and test/development utilities. Each layer is modular and open-source, allowing for extensibility and integration.

All software developed on Openwater's main branch is licensed under the AGPL.

For the latest released version of the Open-Motion application, visit Openwater's GitHub repo, openwater-bloodflow-app .

### Tech Stack

**Front End:**
PyQT v6.8

**Middle Layer:**
openmotion-sdk
SDK Documentation

**Back End:**
Microcontroller on sensor modules and console
Serial COM port protocol
USB 2.0 HS
PySerial v3.5
libUSB v3.3.1
PyUSB v1.3.1

**Databases:**
*The database is an active roadmap feature and is currently being developed.*

**Tools:**

| Category | Tools / Platforms |
|---|---|
| Version Control | Git + GitHub |
| CI/CD (optional) | GitHub Actions (under development) |
| IDEs | STM32CubeIDE |
| Documentation | Markdown, docx |
| Testing | Oscilloscope, logic analyzers, unit test |

## Database Architecture

*The database is an active roadmap feature and is currently being developed.*

# Software Architecture

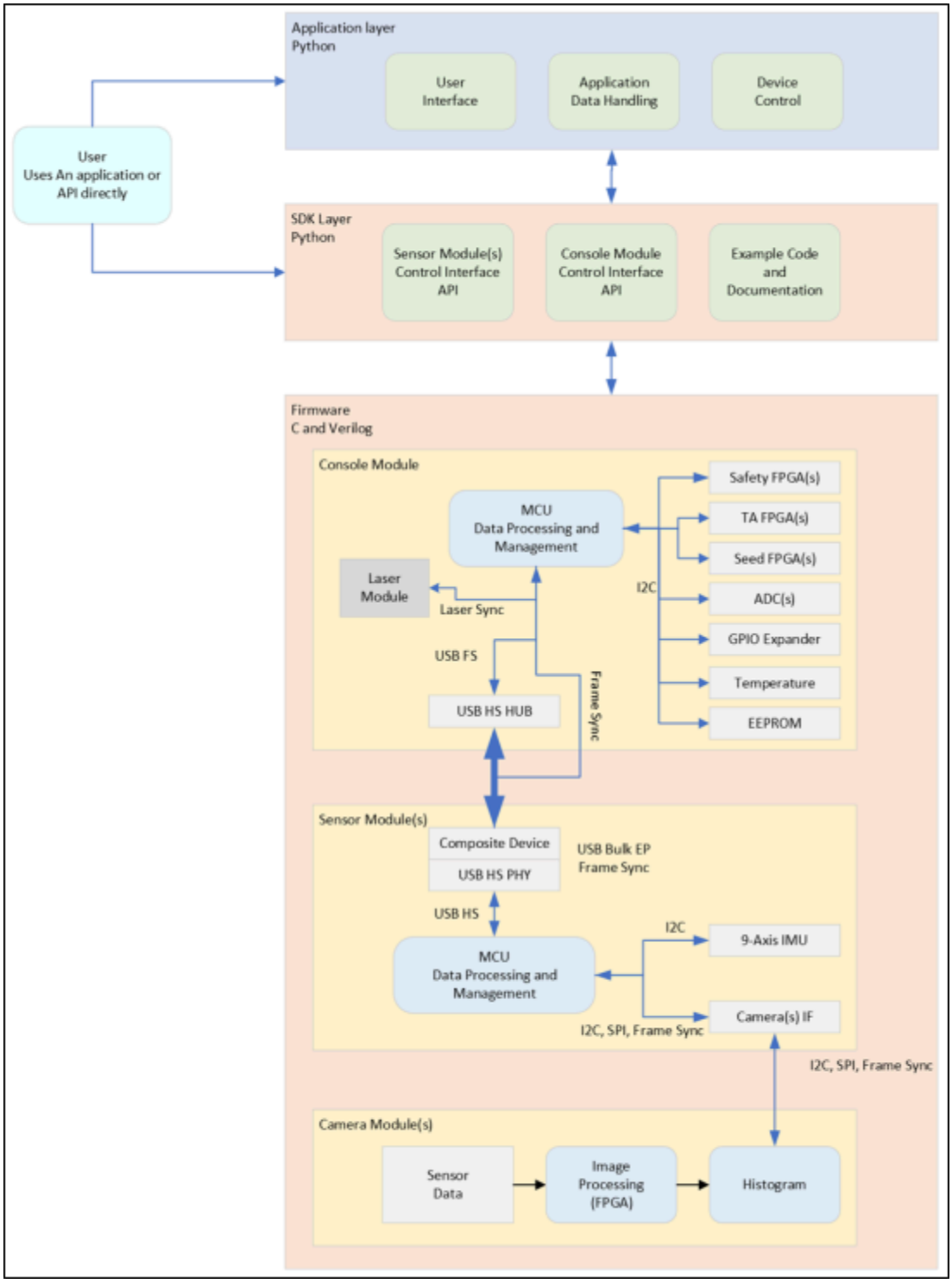


*Figure 6: Open-Motion software architecture diagram.*

# Embedded Firmware Layer

This layer includes all firmware running on hardware devices (camera, aggregator, console).

**Camera Module FPGA Firmware**

- Hardware: Lattice FPGA
- Repo: openmotion-camera-fpga
- Function: Converts raw 2MP grayscale image frames into 1024-bin histograms at 40 FPS
- Interface: Outputs histogram data via SPI
- Customizable: RTL (Verilog) can be modified for alternative processing methods

**Aggregator Board Firmware**

- Hardware: STM32 MCU
- Repo: openmotion-sensor-fw
- Function:
    - Receives histogram data from 8 camera SPI inputs
    - Aggregates and transmits data over USB HS
    - Provides USB control endpoints for config/status
- Interface: Outputs aggregated data and exposes a control/monitoring interface over USB

**Console Firmware**

- Hardware: STM32 MCU + USB Hub + Laser Control FPGAs
- Repo: openmotion-console-fw, motion-safety-fpga, motion-seed-fpga, motion-ta-fpga
- Function:
    - Generates FSIN sync pulses
    - Triggers laser pulses with programmable delay
    - Routes Aggregator data to the host PC
    - Controls power, fan, and safety interlocks
    - Provides a monitoring and Control interface for the entire system
- Interface: Outputs aggregated data and exposes a control/monitoring interface over USB

### Sensor Module Diagram

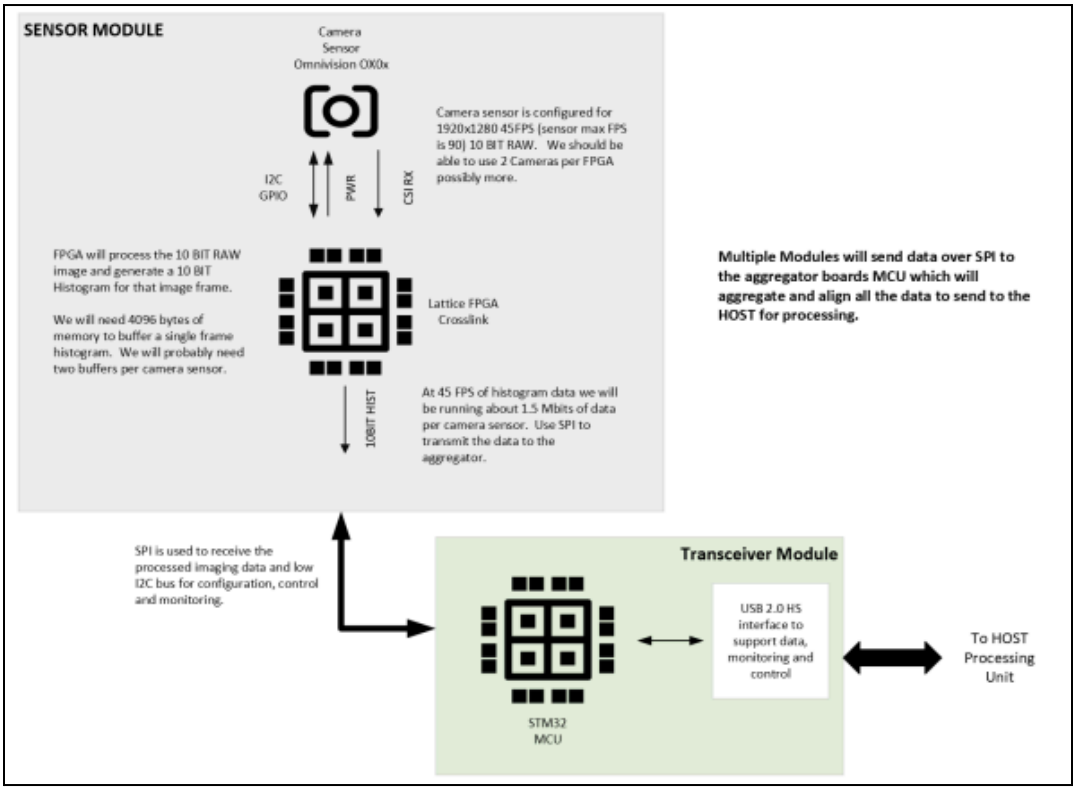


*Figure 7: General schematic of the sensor module.*

## Host Software Layer

This layer runs on the PC and provides device control, data acquisition, and visualization.

**Python SDK**

- Repo: openmotion-sdk
- Function:
  - Enumerates and connects to aggregators/console
  - Reads and parses 16-camera histogram streams
  - Computes blood flow and volume metrics
  - Provides real-time visualization and logging
- Platform Support:
  - Current: Windows 11+
  - Future Development: macOS 12+, Linux
- Extensibility:
  - Open-source and modular (based on Python 3.12+)
  - Integrates easily with NumPy, Pandas, Jupyter, and other scientific tools
  - Example test scripts can be found in the openmotion-sdk
  - API documentation can be found in openmotion-sdk/docs

## Test & Validation Utilities Layer

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

**Test Engineering Application**

- Repo: openmotion-test-app
- Framework: Python + PyQt6
- Function:
    - Live preview of histogram data per camera
    - Validate USB/SPI communication
    - Inspect hardware status (version, ID, sync)
    - Automate production tests

# External Device Connectivity

The Open-Motion device has Trigger and Sync capabilities that can be enabled via the SDK functions. When the system is turned on, it outputs a 40 Hz square wave that is triggered by the camera capture.

Syncing Open-Motion with an incoming TTL signal is an active roadmap feature and is currently being developed.

# Open-Motion 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 one and two-sensor module CAD assemblies. The part numbers and quantities are provided in the tables below.

## CAD Files

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

## Mechanical

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

### Sensor Module Assembly

*Table 4: Top-level mechanical assemblies for the Open-Motion device.*

| Part Name | Part Number |
|---|---|
| Motion Console W/Power Cable and USB Cable | 700-00013 |
| Motion Head Strap, Arm/Leg Strap, Torso Strap | 700-00029 |
| Sensor Module with Cable (console to Sensor Module) and Protective Cover | 700-00031 |

## Electronics

- Contains PCB designs for the Open-Motion platform (Console and sensor modules).
- This is where you will find board-level schematics and layout sources for building or modifying hardware.
- Good for: Getting actual electronics hardware build files.

### Open-Motion PCBs

*Table 5: Electronic board designs for the Open-Motion device.*

| Part Name | Part Number |
|---|---|
| Console Board | 710-00024 (Board Design) – IN DESIGN<br>720-00010 (Schematic) – IN DESIGN |
| Camera Board | 710-00021 (Board Design) – IN DESIGN<br>720-00007 (Schematic) – IN DESIGN |
| Aggregator Board | 710-00019 (Board Design)<br>720-00005 (Schematic) |