CyVital Final Design Document

SD-May25 Group 22

Meng Lu

Sajan Patel, Jay Patel, Chuck Mallek, Ty Beresford, Daniel Karpov

sdmay25-22@iastate.edu

https://sdmay25-22.sd.ece.iastate.edu/

Executive Summary

Cy-Vital addresses the limited access to affordable, effective biomedical education tools at Iowa State. BME 3500 is a brand new course at Iowa State. BME 3500 students require hands-on experience with biomedical signals. Existing training systems are often prohibitively expensive and outdated, leaving students without adequate hands-on experience using real-world biomedical devices. We aim to fix these problems and allow Cy-vital to give students the experience they need to be successful in the biomedical world. The key design requirements for Cy-Vital include accurate data acquisition from physiological sensors, real-time data visualization, and modularity to allow seamless integration of various sensors. The system must be user-friendly and follow the educational objectives.

The design integrates three main components:

1. **Data Acquisition Unit (DAQ):** Converts analog signals from sensors into digital data and transmits it to a computer.
2. **Physiological Sensors:** ECG, Pulse oximeter, and EMG sensors to measure heart rate, blood O2 levels, and muscle response.
3. **Graphical User Interface (GUI):** Displays real-time data, enabling students to analyze physiological signals interactively.

The system employs technologies like Python for GUI development, Analog Discovery 3 DAQ, and digital filtering for noise reduction. This modular approach allows scalability and ease of use for students and instructors. Progress to date includes successful testing of ECG sensors, initial DAQ integration, and a functional Python-based program to display data. We have prototyped a GUI and implemented basic signal-processing algorithms. Challenges, such as ensuring clean signal acquisition and constant ground connection, are constantly considered. As students, we have a unique perspective as one of our users will be

other students. This allows us to develop the product with the student user in mind and make sure that we adhere to this user. We are building a foundation with the base software that will allow us to implement convenient user interaction with the other two users.

The next steps involve refining the GUI, improving signal processing algorithms, and integrating additional sensors. The team will also focus on system-level testing and user feedback to enhance functionality and ensure the platform meets educational objectives.

Learning Summary

Development Standards & Practices Used

- IEEE 1073.4.1a-1999 - IEEE Standard for Medical Device Communications - Physical Layer Interface - Cable Connected
    - This IEEE standard describes a physical interface for the interconnection of computers and medical devices. The physical and electrical characteristics of the connector and signals necessary to exchange digital information between cable-connected medical devices and host computer systems are specified for this standard. This standard was implemented for all analog and digital sensor data communication and data collection.
- IEEE 1228-1994 - IEEE Standard for Software Safety Plans
    - This IEEE standard specifies the acceptable requirements for the content of a software safety plan. This pertains to software safety plans for the development and maintenance of critical safety software. This standard requires that the plan be prepared in the context of system safety. This standard was used in practice during

code reviews and design standards to ensure that all safety critical software components were tested to ensure and maintain an acceptable level of safety.

- IEEE 1016-1998 - IEEE Recommended Practice for Software Design Descriptions
    - This IEEE standard describes the part of the software design descriptions (SDDs). An SDD is a representation of the software system that is used as the main medium for communication. This is applicable to paper documents, databases, design description languages, or other means. This standard was applied when documenting our software functionality and design.

Summary of Requirements

- Each sensor method must be implemented and integrated onto the main circuit board

- Sensor analysis software must be written in a language familiar to the BME students prior to BME 3500

- A graphical interface must be developed to allow students to collect data and analyze output plots from their signal processing functions

- A lab manual must be written to accompany the students to guide them through the implementation of of each data analysis and signal processing method for each sensor

Applicable Courses from Iowa State University Curriculum

- EE 201, 230
- CPRE 288, 281
- COMS 227, 228, 309

New Skills/Knowledge acquired that was not taught in courses

ADC

Waveforms

I2C

Arduino platform

Soldering

Breadboard basics

Python GUI integration

Software API

Table of Contents

1. Introduction

1.1 Problem Statement

The current access to affordable and effective healthcare education is very limited. Most training equipment is thousands of dollars, the software is decades old and difficult to operate. This creates a disconnect between theory and practice, leaving students without a solid understanding of how biomedical devices work in real-world situations. As healthcare technology develops, there is an urgent need for workers to understand theory as well as have the ability to work with the equipment to develop new solutions.

Our project, Cy-Vital, addresses this problem by creating a hands-on learning environment for students studying biomedical engineering. The goal of this project is to show students how to engage directly with physiological devices such as heart rate, O2 level, and muscle sensors as well as understand how to read and interpret the data given to them once they test the sensors on a live volunteer. The broader issue we aim to address is making affordable and effective tools that allow students to experiment safely. Without tools like this, students miss valuable opportunities to explore medical technologies and experience building effective medical devices.

Our project is designed to solve these issues by integrating the physiological sensors into a single, easy-to-use system. Students will be able to measure, record, and analyze real-world data using the provided sensors and their own developed programs. This helps the students gain firsthand experience building basic medical devices based on the content they learn in lecture. Our projects aim to equip future biomedical engineers and empower them to innovate in ways that will allow them to design, engineer, and program medical sensors for analysis using our platform.

1.2 Intended Users

User 1: Biomedical Engineering Students

CyVital has the intention of reducing overall overhead while using physiological devices and its sensors. CyVital is bridging software between the hardware (the sensors) and the computer itself. This allows the Biomedical Engineering students to properly read and analyze data returned by the sensors.

Biomedical Engineering students need to have a proper understanding of statistical data returned from any physiological sensor, however, it is not their job to navigate a confusing, outdated existing software. With advancements made in technology and overall reduced runtimes, it should be expected that the software can properly conform to today's standards. However, based on anecdotal evidence, our mentor is suggesting a cheaper, simpler yet more efficient solution.

The benefits of using CyVital can be categorized as such:

1. The fluidity of the program
    a. Color toning
        i. Colors are appealing and reveal data in priority
        ii. The graphical user interface makes "sense" and does not obstruct data analysis
    b. Increased framerate
        i. Reduce bugginess and frame rate drops
            1. Impose data caching when possible to reduce data transfer costs
            2. Update libraries to meet new standards
    c. Increase overall modularity
        i. Remove product-specific cables

1. Replace cables with USB-C to increase overall data transfer rates and allow for complete plug-and-play modularity
2. Allow code to be open-sourced so necessary personal modifications can be made

Despite requiring an absolutely large rewrite of the existing program, the basic premise of the project remains the same: increase user interaction satisfaction and decrease overall overhead from existing software.

As part of the learning process, students will be tasked with filling in functions and methods within the code that powers the interaction with the physiological sensors (such as heart rate, reaction time, O2, and muscle sensors). These coding exercises will be designed to be incremental and hands-on, giving students the opportunity to learn by doing. Here's how the project will encourage students to engage with the programming aspect:

1. Students will begin by exploring the pre-written code that works with the sensors. This will give them an idea of how the data is captured and processed from the sensors.

```
def _estimate_heart_rate(self, ecg_signal):
    """Detects R-peaks and estimates heart rate."""
    filtered_ecg = signal.filtfilt(self.b, self.a, ecg_signal)
    peaks, _ = signal.find_peaks(filtered_ecg,
                                 distance=self.data.sample_rate//2.5,
                                 height=np.mean(filtered_ecg) + 0.5*np.std(filtered_ecg))

    if len(peaks) > 1:
        rr_intervals = np.diff(peaks) / self.data.sample_rate
        return int(60.0 / np.mean(rr_intervals))
    return None
```

Ex. Student would analyze and re-write any portion of code for any sensor (ECG, EMG, Pulse Oxygen and Reaction Time)

2. As they work through the code, they will find “gaps” in the methods that are not yet fully implemented. These gaps will be made to push students to figure out how to complete the code. For example, a student might need to write a function that reads data from a heart rate sensor and displays it.
3. Students will run their program on live subjects, which allows them to observe how their code interacts with the real-world data. The iterative process of coding and refining, ensure students learn along the way
4. By filling in the code themselves, the students will have to learn the underlying principles of the medical sensors, data analysis, and sensor calibration. This approach will help the students get a deeper understanding of how the sensors work.

User 2: Lab Teaching Assistant

These Lab TAs will be running each lab for the bioengineering students. The TAs will need to have a great understanding of the software that is being used as well as how the equipment works. The lab TAs will need to be able to introduce the lab to the students in a clear and concise manner that would make sense to the bioengineering students. No lab should also be too hard for the students to do nor too long for them to complete within the lab time.

Needs Statements:

- Need to be able to give clear and concise instructions
- Need to be able to help with the lab
- Need to be able to do the labs themselves
- Need to be able to troubleshoot sensors

How TAs might derive value from Cy-Vital

- Deepen their understanding on bio-instrumentation

    - The TAs will have to be able to help students often with the labs and this will help solidify/improve their understanding on the topic
- Improve Teaching and communication skills
    - The TAs are going to have to guide students through the labs and help them with any questions they have
- Collaboration
    - TAs are going to have to work closely with the professor of the course in order to choose the best labs week to week and make sure that the labs align with what is being taught in the lectures.

User 3: Course Professor

The course professor is responsible for administering and overseeing the course curriculum and making sure that students are meeting specific learning objectives needed for the BME course. The professor will have a deep understanding of the theoretical and practical skills the students need to learn. Professors will also be in charge of evaluating the effectiveness of the lab, managing TAs, and guiding students through the course

Needs statements:

- Needs to make sure lab tools and activities align with course objectives
- Need the hardware and software solution to be both reliable and intuitive to use
- Need to be able to easily interpret and grade student data collected from the labs

Benefits to the professor:

- The CyVital system is both a hardware and software platform which allows for a simplified way to conduct student labs

- By using our solution, the professor can focus on teaching the curriculum rather than troubleshooting and maintaining outdated and old systems
- Because the CyVital system is developed in house, the cost per unit will be much cheaper than market solutions.

2. Requirements, Constraints, And Standards

2.1 Requirements & Constraints

Functional Requirements:

1. Sensor Data Acquisition
    a. The system can accurately and efficiently read data from various sensors, whether it is temperature, heart monitor, or blood oxygen saturation
2. Data Analysis
    a. The software, given any specific data set, can observe and deduce possible readings on output data
3. Data Visualization
    a. The system must properly display sanitized collected data, and users must be able to interact to break down each data type

```python
@dataclass
class EcgData:
    heart_rate: Optional[int] = None
    waveform_buffer: Deque[float] = field(default_factory=lambda: deque(maxlen=4096))
    raw_samples: List[float] = field(default_factory=list)
    sample_rate: int = 8192
```

Ex. Thread-safe data structures capable of storing a large number of values to be plotted

Educational Requirements:

1. Ease of implementation for students

    a. The CyVital software should be in a familiar language for biomedical students, to ease the learning curve of implementing functions and methods for completing labs
2. Modularity
    a. The project should have an element of modularity. The system should be designed in such a way that a student could collect data from any type of analog signal and write a program to analyze the data.
3. Educational Value
    a. The platform should coincide with the learning objectives of the course curriculum. Implementation of the project should help teach course concepts in an interactive and practical manner

Resource Requirements:

1. Sensor Compatibility
    a. The system must be compatible across different systems, OS, and potentially different dated sensors
2. Development Tools
    a. By utilizing Python, a universal language, consistent modules across the board will be trivial

Aesthetic Requirements:

1. User Interface Design
    a. The UI must be clean and modern, avoiding clutter and disorganization; users should be able to grasp large data without losing track of objective
2. Color Scheme

a. The UI needs to blend and approach the color wheel correctly; clashing colors will make data analysis difficult; data will be prioritized correctly with color schematics

Performance Constraints:

1. The system must process and display sensor data - any latency of no more than 2 seconds will be reported
2. The application must securely store data until the user terminates their account or until data is no longer applicable
3. Modular system: each sensor must be adjustable to match the DAQ as necessary

2.2 Engineering Standards

- IEEE 1073.4.1a-1999 - IEEE Standard for Medical Device Communications - Physical Layer Interface - Cable Connected
    - This IEEE standard describes a physical interface for the interconnection of computers and medical devices. The physical and electrical characteristics of the connector and signals necessary to exchange digital information between cable-connected medical devices and host computer systems are specified for this standard.
- IEEE 1228-1994 - IEEE Standard for Software Safety Plans
    - This IEEE standard specifies the acceptable requirements for the content of a software safety plan. This pertains to software safety plans for the development and maintenance of critical safety software. This standard requires that the plan be prepared in the context of system safety.

- IEEE 1016-1998 - IEEE Recommended Practice for Software Design Descriptions
    - This IEEE standard describes the part of the software design descriptions (SDDs). An SDD is a representation of the software system that is used as the main medium for communication. This is applicable to paper documents, databases, design description languages, or other means.
- IEEE 11073-10101: Nomenclature for vital signs data, which helps standardize the representation of medical data across devices.
    - This IEEE standard provides nomenclature for point of care(POC), personal health devices(PHD), and medical device communication(MDC) that supports the domain information model and service model components. This is specialized for vital signs such as ECG, respiration, neurology, and more.

3. Project Plan

3.1 Project Management/Tracking Procedures

For our project our team has chosen to do a Waterfall-Agile hybrid approach. This style combines the structured beginning phases of the waterfall method while having the flexibility of the agile development model.

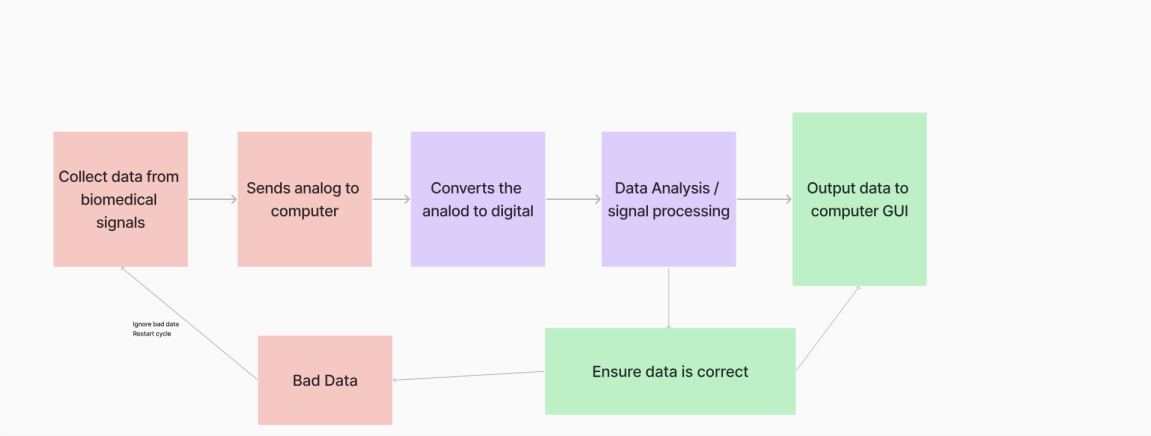
Initial Waterfall phase: The early stages of the Cy-Vital project(requirement gathering, design, and initial prototyping) need a linear approach for the groundwork of the project. This phase includes defining the technical requirements and establishing the needed hardware to work with physiological sensors. The early foundation is built with this phase and helps make the next phase much easier

Agile phase: After the initial system components are figured out, we can shift to an Agile development approach. This approach is a more iterative approach that is

continually evolving in order to refine features, respond to the feedback, and improve the project overall. This will help us with issues such as signal accuracy and interference which are adjustments that are ongoing.

The main progress-tracking tool we are going to use is GitHub and weekly team meetings. GitHub ensures effective collaboration and progress tracking so that we can see the project grow and evolve from a software perspective. This also helps with version control, documentation, and code management. All team members will be able to track changes, pull and push code, and review each other's work, ensuring quality code is being pushed. While weekly team meetings will ensure that the project overall is staying on track and we are not veering off or moving too slowly.

3.2 Task Decomposition



3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. Milestone: Frontend-Backend-Hardware Integration

Description: Achieve a seamless merge between the frontend/backend processing and hardware components.

Metrics & Evaluation Criteria:

- Complete system integration across all modules, validated through end-to-end testing.
- **Data Acquisition Quality (DAQ)**: Register DAQ signals with a 20% improvement in analytics processing speed on software produced by our team.
- **Progress Evaluation:** Performance metrics from the DAQ system to validate the 20% increase.
- **Refinement Opportunity:** Refine after each sprint iteration to ensure results across updates.

2. Milestone: Sensor and Lab Module Integration

Description: Complete 4 sensors and their code integration

Metrics & Evaluation Criteria:

- **Integration Rate:** Current target of 4 lab modules this semester. This is measured by sensor compatibility, accuracy, and responsiveness.
- **Signal Accuracy:** Ensure that each sensor measures biological signals with 95% accuracy, verification with calibration tests against baseline data.
- **Refinement Opportunity:** Address any calibration issues iteratively per module, applying feedback for each sprint as new modules are integrated.

3. Milestone: BME Lab Kits Creation and Software Release

Description: Develop and release Biomedical Engineering (BME) lab kits alongside software.

Metrics & Evaluation Criteria:

- **Client Satisfaction:** Achieve feedback indicating high satisfaction for five newly created lab modules, measured through surveys or client evaluations.
- **Kit Readiness:** Ensure that each lab kit meets functionality, durability, and ease-of-use criteria upon release.
- **Refinement Opportunity:** Reassess client feedback post-initial release to address usability or functionality feedback in future versions.

4. Milestone: Signal Processing Performance

Description: Increase the accuracy of our signal processing algorithms for reliable data accessibility.

Metrics & Evaluation Criteria:

- **Signal Processing Accuracy:** Achieve 95% accuracy on all biological signals, validated through repetitive testing.
- **Processing Speed:** Increase signal processing speed by 15% each sprint, measured by recording processing times under similar testing conditions.
- **Refinement Opportunity:** Adjust algorithm parameters and optimize code with each sprint to meet or exceed speed and accuracy targets.

5. Milestone: User Interface (UI) and User Experience (UX) Evaluation

Description: Evaluate and refine an easy-to-use UI.

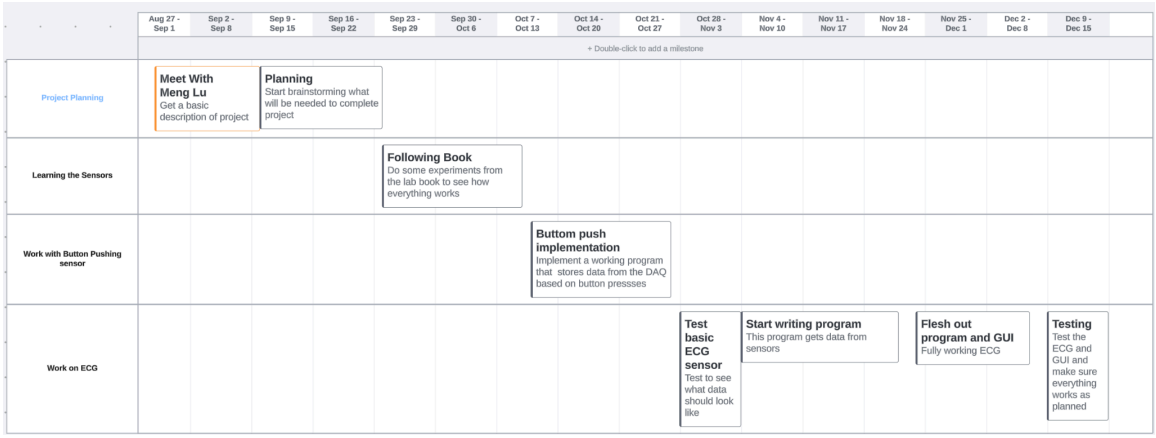
Metrics & Evaluation Criteria:

- **User Satisfaction Score:** Achieve an 85% satisfaction rate in user testing. Core areas of improvement are usability, clarity, and accessibility.

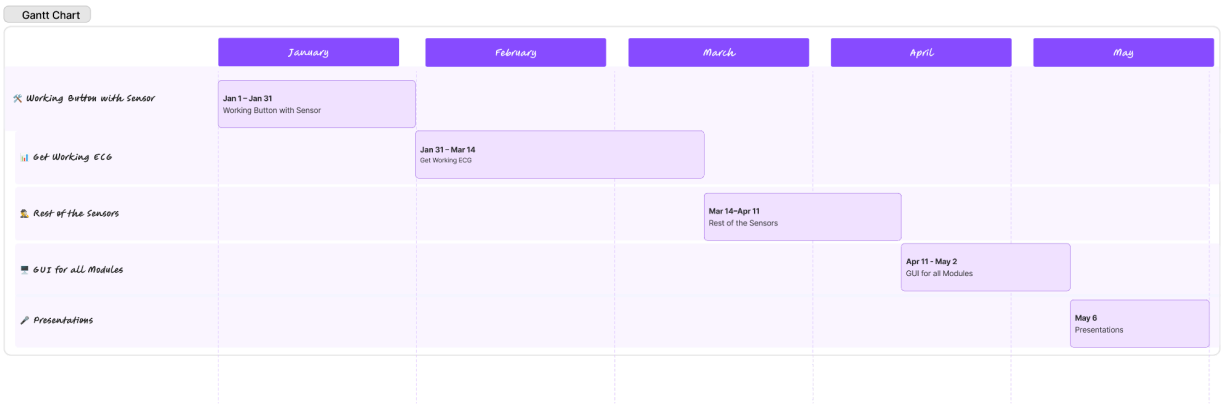
- **Task Completion Time:** Decrease average task completion time by 10%, measured by tracking the average user interaction time.
- **Educational Outcomes Met:** Ensure the platform achieves and aids in the course learning objectives defined in the course criteria
- **Refinement Opportunity:** Adjust UI elements based on user evaluation, aiming for improved usability and interaction flow.

3.4 Project Timeline/Schedule

Semester 1 Gantt chart:



Semester 2 Gantt chart:



Deliverables:

Working button with sensor: 1/31

Get working ECG: 3/14

Rest of the Sensors: 4/11

Gui for all Modules: 5/2

Presentations 5/6

3.5 Risks and Risk Management/Mitigation

Sensor Malfunction / Data Loss

Risk Description: Sensors may malfunction, leading to inaccurate or incomplete data capture.

Probability: 0.4, **Severity:** High

Mitigation Plan:

- Perform regular checks on sensors to ensure data accuracy.
- Explore the purchase of backup sensors or monitoring tools.

Software Bugs / Crashes

Risk Description: Software bugs could cause failures in the sensor interface or data processing.

Probability: 0.6, **Severity:** High

Mitigation Plan:

- Conduct unit and integration testing, especially for sensor-related code.
- Regularly update software and utilize version control through Git.
- Schedule code reviews to catch potential issues early.

GitLab Main Branch Nuke

Risk Description: The main branch on GitLab could be accidentally wiped

Probability: 0.05, **Severity:** High

Mitigation Plan:

- Require all main branch updates to be reviewed via pull requests.
- Use branch protection and continuous backups for recovery.

Meeting Performance Targets

Risk Description: The system may fail to meet specific data speed, accuracy, or reliability performance metrics.

```
def main():
    app_state = AppState()

    runner_params = hello_imgui.RunnerParams()
    runner_params.app_window_params.window_title = "ISU CyVitals Beta"
    runner_params.app_window_params.window_geometry.size = (800, 600)
```

Ex. Use imGui default handler to manage frame locking (FPS limiting to maximize smoothness between sensor and software)

Probability: 0.5, **Severity:** Moderate

Mitigation Plan:

- Run simulations on alternative tools or algorithms.
- Benchmark against different hardware or sensors to compare performance.

Tool/Technology Limitations

Risk Description: Tools like SQL Server or GitLab may have limitations impacting performance or reliability.

Probability: 0.3, **Severity:** Moderate

Mitigation Plan:

- Regularly assess new versions of tools for relevant feature updates.
- Consider alternative platforms or frameworks if limitations persist.

The only risk that came to pass was sensor malfunctions. We had issues with our original sensors that were outdated and would not work at all. To fix this issue, we got a newer version of the same type of sensor, and we were able to get a proper signal.

3.6 Personnel Effort Requirements

Task	Description	Team Members Involved	Estimated Hours per Member	Total Person-Hours
1. Project Planning	Define scope, goals, requirements	All	4	20
2. Hardware Design & Setup	Select and assemble sensors, test compatibility	3	10	30
3. Data Acquisition System Development	Develop and integrate data acquisition unit	3	12	36
4. GUI Initial Design	Create interface wireframes, basic functionality	2	8	16
5. GUI Development & Refinement	Build GUI features (start/stop, visualization)	2	20	40
6. Signal Processing & Noise Reduction	Implement filtering for ECG, EEG, etc.	2	15	30
7. Integration of Sensors & Data Sync	Synchronize multi-sensor data	All	10	50
8. System Testing & Debugging	Test individual components and overall integration	All	15	75
9. Documentation & User Guide	Create documentation and guides for users	2	10	20
10. Final Testing & Adjustments	Finalize testing, troubleshoot, and refine	All	12	60

11. Presentation & Demo Prep	Prepare for project presentation and demo	All	8	40

Explanation and Reference:

1. **Initial Planning:** All members will spend 20 hours defining the project scope, requirements, goals and creating an initial plan.
2. **Design and Development Phases:** This includes hardware and software tasks (2-7), such as signal processing, hardware, and GUI setup. Some processes will take longer than others because they are more difficult and require more attention.
3. **Testing and Documentation:** System testing requires all team members so that there are no errors or missteps in the system. Documentation is also key as it will provide the user documentation to look at if coming goes wrong or as an educational tool
4. **Presentation and Final Adjustment:** These final tasks ensure the project is polished and well presented for the client.

3.7 Other Resource Requirements

1. Collaboration & Communication Tools

- **GitHub**: Collaborative work
- **Discord**: Real-time communication for team meetings
- **Google Drive**: Usage for note sharing and collaboration

Importance: Enabled seamless coordination

2. Software & Development Tools

- **Python Ecosystem**: Libraries like NumPy (signal processing), Matplotlib (data visualization), and dwfpy (Analog Discovery 3 DAQ integration).
- **imGui Framework**: Graphical user interface.
- **WaveForms SDK**: Software development kit for Analog Discovery 3.

Importance: Provided the technical backbone for project.

3. Hardware & Testing Tools

- **Soldering Equipment**: For prototyping sensor circuits and multiplexer integration.

Importance: Ensured hardware reliability.

4. Expertise & Knowledge Resources

- **Faculty Mentorship**: Guidance from professors on biomedical signal standards.

Importance: Bridged knowledge gaps.

5. Physical Infrastructure

- **Lab Space**: Access to soldering stations, testing benches, and sensor calibration tools.
- **Volunteer Test Subjects**: For live ECG, EMG, and pulse oximetry data collection.

Importance: Enabled real-world prototyping.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	BME students are afforded the opportunity to use our system to apply their learning and develop biomedical sensors in practice, which better equips them for work in industry	Reduces cost of lab administration Allows for interactive approach to sensor development
Global, cultural, and social	This project is designed to improve the readiness and vocational skills of prospective biomedicals students before they begin work in industry	This project improves the readiness of students and increases their exposure to ethically sound engineering development
Economic	This project was designed to be as cost affordable as possible. Many systems currently on the market have costs that exceed over 10,000 dollars per kit, making them prohibitive to use.	Total cost of this project is under 600 dollars, while market competition can be over 10,000

4.1.2 Prior Work/Solutions

Existing products on the market include BIOPAC, OpenBCI, and BITalino. BIOPAC offers high-end, expensive physiological monitoring for research purposes. It supports multi-signal acquisition and a powerful analysis tool. OpenBCI is a notable open-source alternative with support for EEG, EMG, ECG, and more. The hardware is a bit more affordable than the BIOPAC. The BITalino platform is another low-cost solution that is for educational purposes. It provides basic GUI and sensor support.

Our project differentiates itself from the others by taking all the advantages of each and bringing them together. We are attempting to make an affordable, high-quality, open-source biosignal acquisition system. While others have one or two of these, we plan to implement all of these features.

[1] BIOPAC Systems Inc., "BIOPAC Student Lab," [Online]. Available: https://www.biopac.com/product/bsl-pro/.

[2] OpenBCI, "OpenBCI: Biosensing for Everyone," [Online]. Available: https://www.openbci.com/.

[3] H. da Silva et al., "BITalino: A novel hardware framework for physiological computing," *Proc. ACM Int. Joint Conf. Pervasive and Ubiquitous Computing Adjunct Publication*, 2014, pp. 479–482. DOI: 10.1145/2638728.2641693.

4.1.3 Technical Complexity

Provide evidence that your project is of sufficient technical complexity. Use the following metric or argue for one of your own. Justify your statements (e.g., list the components/subsystems and describe the applicable scientific, mathematical, or engineering principles)

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles –AND–
2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

4.2 Design Exploration

4.2.1 Design Decisions

The following key design decisions were made to balance technical feasibility, educational value, and cost-effectiveness:

1. **Python with Minimal External Libraries**
    - *Rationale*: Python was selected as the primary programming language due to its widespread adoption in biomedical engineering curricula. By minimizing external dependencies, students can focus on core signal processing concepts without struggling with complex library setups. This aligns with the requirement for modularity and accessibility.
    - *Trade-off*: While Python's slower execution speed compared to C++ or Java was noted, its simplicity and educational value outweighed performance concerns for this application.
2. **16:1 Multiplexer for Sensor Integration**
    - *Rationale*: A 16-channel multiplexer was chosen to accommodate future scalability (currently supporting 4 sensors). This allows students to add new sensors without redesigning the hardware.
    - *Trade-off*: Increased circuit complexity was accepted to prioritize modularity and long-term flexibility.
3. **Analog Discovery 3 (AD3) DAQ**
    - *Rationale*: The AD3 replaced earlier MCC and AD2 hardware due to its USB-C connectivity, higher sampling rates (up to 15 MHz), and compatibility with modern systems. Its built-in oscilloscope and signal generator functionalities simplified prototyping.
    - *Trade-off*: Transitioning hardware mid-project introduced delays but ensured compatibility with future lab environments.
4. **Open-Source GUI Framework (IMGUI)**
    - *Rationale*: After testing Tkinter and PyGUI, IMGUI was selected for its high frame rate and dockable windows, which improve real-time

data visualization. Open-source licensing allows customization by students.
    - *Trade-off*: IMGUI's steeper learning curve was mitigated through pre-built templates.

4.2.2 Ideation

For selecting the data acquisition unit, we considered multiple ideation techniques and reviewed various options through brainstorming and feature-mapping to balance connectivity, data transfer speed, and power consumption. **Options considered** included:

1. **USB Wired Connection**
    - A stable connection with low latency, widely accessible, and convenient for classroom settings
2. **Bluetooth**
    - Wireless, flexible, and easy to use; but can introduce latency and signal interference.
3. **Wi-Fi Module**
    - Offers long-range connectivity but might be affected by network congestion and can add latency
4. **Ethernet Wired Connection**
    - Stable with low latency, but limited because it is not widely accessible to all computers, which might hinder setup flexibility
5. **Radio Frequency (RF) Module**
    - This offers low power consumption and reduced interference, but lacks the bandwidth needed for the high-frequency data used in the physiological sensors

4.2.3 Decision-Making and Trade-Off

We used a weighted decision matrix to compare each option based on key criteria: **latency**, **data transfer rate**, **setup complexity**, and **reliability**. The table below outlines the scoring system used, with higher scores indicating greater suitability and all scores are out of 10

Option	Latency	Data Transfer Rate	Setup Complexity / Availability	Reliability	Total
USB Wired	9	8	8	9	34
Bluetooth	6	6	8	5	25
Wi-Fi Module	7	7	6	6	26
Ethernet	9	8	5	9	31
RF module	5	5	9	7	26

Decision Summary

- **Chosen Option**: **USB Wired Connection**
- **Reasoning**: USB scored highest due to its low latency, consistent data transfer rate, and high reliability in a lab setting. The direct connection also simplifies setup. Although Ethernet had similar benefits, the USB connection was chosen for its ease of use and availability, making it more suitable for classroom environments where flexibility is key.

4.3 Final Design

4.3.1 Overview

CyVital is a hands-on educational tool designed to help students learn about how the body's signals (like heart rate, brain activity, and muscle movements) are collected and analyzed. It has three main parts:

1. **Data Acquisition Unit**: This is the central hub. It collects signals from sensors attached to the body and sends them to the computer in real time.
2. **Physiological Sensors**: These are small devices that detect signals from the body. We sourced each of our sensors from SparkFun, including:
    - The SparkFun AD8232 ECG sensor
    - The SparkFun MyoWare Muscle sensor
    - The SparkFun MAX30101 photodetector for Blood Oximetry
        i. (Note: this sensor requires I²c configuration that must be setup using the dwfpy library in order to work)
    - iworx Reaction Time button
3. **GUI Software (Graphical User Interface)**: On the computer, this software shows the signals in real-time, so users can see their heart rate, brain waves, or muscle activity as it happens. The GUI should be designed to allow students to select a specific module and run their signal processing implementation.

Together, these parts help students interact with real physiological data, giving them an understanding of how medical and health-related data is gathered and visualized.

4.3.2 Detailed Design and Visual(s)

High-Level System Design

The CyVital system is organized into three interconnected subsystems:

1. **Data Acquisition Unit (DAQ)**:

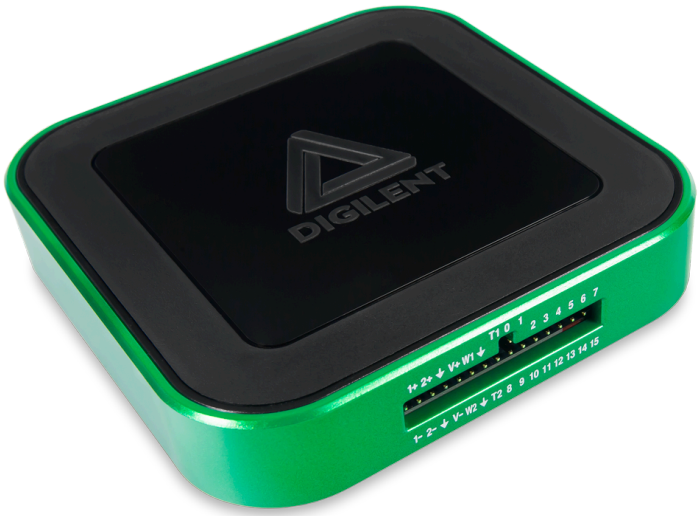


Figure 1. [1] Digilent, “Analog Discovery 3 – Getting Started,” Digilent Reference. [Online]. Available: https://digilent.com/reference/test-and-measurement/analog-discovery-3/start. [Accessed: Apr. 14, 2025].

   - This unit receives raw signals from each sensor, which are analog, and converts them to digital form using an Analog-to-Digital Converter (ADC).

- **Key Components**: ADC for signal conversion, microcontroller for data handling, wireless module for communication, USB-C connection, configurable power supply.

2. **Physiological Sensors**:
   - Each sensor captures specific physiological data:
     - **Reaction Time**: Records the reaction time after pressed.

     - **ECG Sensor**: Measures the electrical activity of the heart.

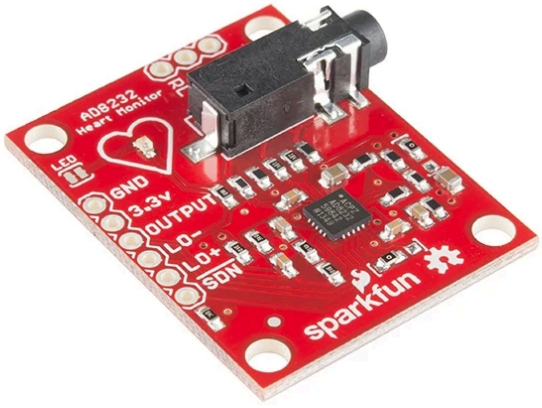


Figure [2] SparkFun Electronics, “SparkFun Single Lead Heart Rate Monitor – AD8232,” SparkFun. [Online]. Available: https://www.sparkfun.com/sparkfun-single-lead-heart-rate-monitor-ad8232.html. [Accessed: Apr. 14, 2025].

- **Photodetector Pulse Oximeter Sensor**: detects oxygen level by measuring the amount of light absorbed through skin.

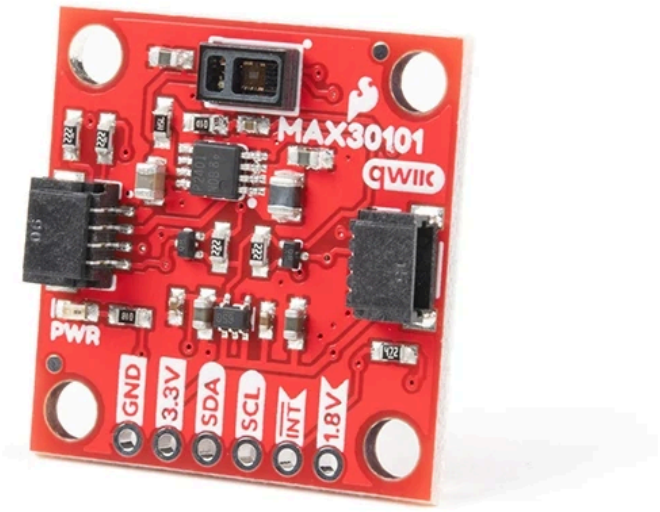


Figure [3] SparkFun Electronics, “SparkFun Photodetector Breakout – MAX30101 Qwiic,” SparkFun. [Online]. Available: https://www.sparkfun.com/sparkfun-photodetector-breakout-max30101-qwiic.html . [Accessed: Apr. 14, 2025].

- **EMG Sensor**: Measures muscle response.

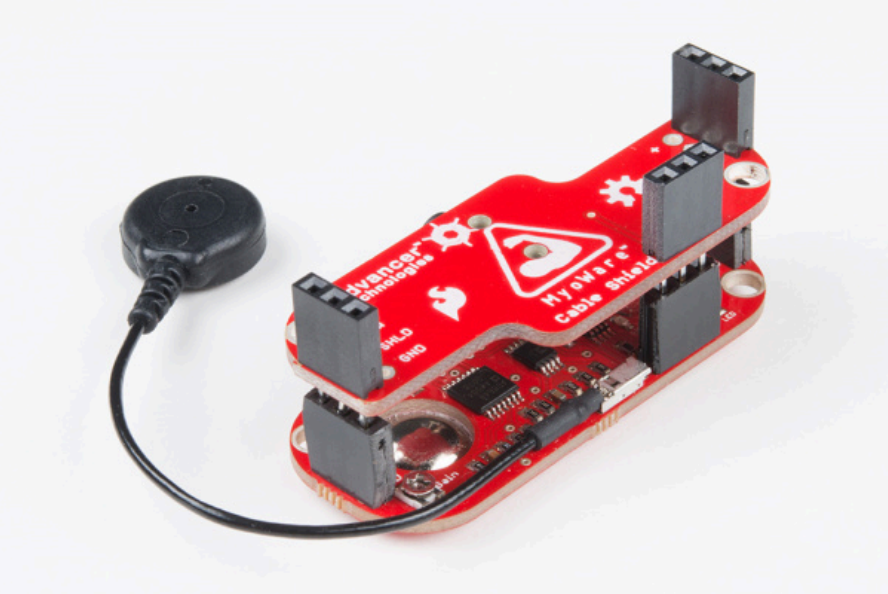


Figure [4] SparkFun Electronics, “MyoWare Muscle Sensor Kit – SparkFun Learn,” SparkFun Learn. [Online]. Available: https://learn.sparkfun.com/tutorials/myoware-muscle-sensor-kit/all. [Accessed: Apr. 14, 2025].

    - The sensors are connected to the DAQ, where their signals are processed and sent to the software.
3. **GUI Software**:
    - This software receives data from the DAQ and displays it on the computer screen (imGui - Python Bindings: https://pyimgui.readthedocs.io/en/latest/#).
    - It features adjustable settings for data analysis (e.g., viewing data over different time intervals).
    - Users can interact with the data, zoom in, and apply basic filters to understand different signal patterns.

Sub-System Details and Visuals

Each subsystem is modular, with clear wiring and connection interfaces:

- **Data Flow Diagram**: A block diagram shows the flow of data from sensors to the DAQ, which processes it and sends it to the GUI software for display.
- **Circuit Diagrams**: The DAQ circuit diagram includes the Analog Discovery 3, microcontroller, and sensor units.
- **GUI Wireframe**: A GUI mockup shows how data from each sensor appears on the computer screen, with labeled graphs and controls for settings.

The descriptions of each subsystem, wiring, and software protocols make the design easy for another engineer to replicate or modify as needed.

4.3.3 Functionality

In a classroom or lab setting, a student uses CyVital by connecting sensors to their body (for example, placing the ECG sensor on their chest to measure heart rate). Once connected:

1. **The System Starts**: The DAQ collects signals from each sensor and transmits them to the GUI software on a computer.
2. **Data Display**: The GUI software immediately shows real-time graphs for each sensor (heart rate, brain waves, etc.).
3. **User Interaction**: Students can adjust settings on the GUI, such as viewing data over different time frames or focusing on specific data (e.g., zooming in on a muscle movement spike). Students will be charged with implementing data analysis functions and signal processing methods for each sensor. The plotting is handled via matplotlib.

This real-time interaction helps students see and understand physiological signals and how these signals are processed in medical and research settings.

4.3.4 Areas of Challenge

Satisfaction to Satisfy requirements

1. **Easy to use GUI:** An easy to use interface was made for all the users to be able to use the program without having to follow a guide on how to use it.
2. **Straightforward lab modules:** Straightforward lab modules mean that the student users are able to use this program easily and are able to focus on learning the material rather than trying to get the lab done.
3. **Data acquisition is accurate:** Accurate data acquisition is important for the users because it will allow for the users to use the program at maximum potential.

Primary Challenge:

Our primary challenge was having different users be able to see data from different users. For example, we need the lab student lab results be shown for the TA users so they are able to grade / help. We also need the Course instructor user to be able to access the data if needed.

Solution:

We've developed a few immediate solutions to this problem. As with the lab portion of CPRE 2880, students are able to perform a live demo of their functions as well as having export data directly from the matplotlib -graph as part of their submission criteria

4.4 Technology Considerations

1. **USB-C DAQ:**

   Strength: Using USB-C allows for higher data transfer rates with low latency. USB-C is also a standard connector, which allows for future devices to connect to the unit

Weakness: creating USB-C sensors adds overall complexity to our project and requires implementation based strictly on USB-C connection standards

Trade-offs: We prioritize data transfer speeds and latency over development complexity in our design

2. **Python GUI:**

   Strength: Easy integration with standard data analysis libraries(numpy,Matplotlib, etc...) and visualizable real-time data plotting for BME students.

   Weakness: Python may be slower than C++ or Java for large datasets or high-frequency inputs. Performance limitations could be noticeable if processing very large data loads.

   Trade-offs: Prioritized simplicity, educational accessibility, and ease of modification. Python's accessibility for students outweighs performance concerns that might arise in an educational context.

3. **Digital Filtering Techniques for Signal Processing:**

   Strengths: Effective in removing noise from ECG, EMG, and Pulse-Oximeter signals, making the data clearer and easier for students to analyze.

   Weaknesses: Digital filters may introduce minor processing delays, and tuning them requires specific knowledge of signal processing.

   Trade-offs: Choose digital filtering over analog filtering for flexibility. Digital filters can be adjusted and improved within python scripts to fit with educational needs.

5 Testing

Our testing strategy is to test any and every change that we make. We aim to test all edge cases and make sure that they are accounted for. This is needed to ensure that our product conforms to our client's specifications while thoroughly targeting all areas with expected certainty.

5.1 Unit Testing

- **What is being tested:**
  - Individual sensors (e.g., EMG, ECG, O2).
  - Data processing algorithms.
  - GUI components (button responsiveness, data visualization accuracy).
- **How:**
  - Example data sets for sensors to validate signal acquisition and accuracy.
  - Automated tests for backend algorithms using test datasets.
  - GUI testing for responsiveness and functionality.
- **Tools:**
  - Python unittest and pytest.
  - Pre-released data analysis software for sensor testing

5.2 Interface Testing

- **Interfaces being tested:**
  - Sensor-to-hardware communication (DAQ and audio port).
  - Hardware-to-software integration for data transfer.
  - User interaction through the GUI.
- **How:**
  - Data sets for sensor input to test GUI responsiveness.
  - Ensure data integrity during transfer between modules.
- **Tools:**
  - Logic analyzers for hardware communication validation.
  - Python integration tests.
  - GUI testing frameworks

5.3 Integration Testing

- **Critical integration paths:**
  - Sensor hardware → DAQ → Backend processing → GUI visualization.
  - Backend algorithms → Data storage and retrieval.
- **How:**

    - Simulate real-world scenarios to ensure data flows correctly.
    - Validate time synchronization between hardware and software.
- **Tools:**
    - End-to-end testing scripts.
    - Performance monitoring tools

5.4 System Testing

- **System-level testing strategy:**
    - Combine unit, interface, and integration tests to validate overall functionality.
    - Test the system with live sensor data to ensure real-time operation.
- **Tools:**
    - Physical sensors are connected to the system for live testing.
    - Data visualization tools for analyzing results.
    - Debugging tools for identifying errors.

5.5 Regression Testing

- **Ensuring stability:**
    - Validate that new features (e.g., adding a new sensor) do not disrupt existing functionalities.
- **Critical features to monitor:**
    - Data acquisition from sensors.
    - Backend data analysis.
    - GUI responsiveness.
- **Tools:**
    - Version control (GitHub) for continuous integration testing.

5.6 Acceptance Testing

- **Demonstrating requirements are met:**
    - Conduct live demonstrations using physiological sensors with predefined scenarios.
    - Involve the client in reviewing data accuracy, GUI usability, and overall performance.
- **Involving the client:**
    - Provide real-time testing sessions with detailed reports.
    - Collect client feedback on usability and accuracy.

5.7 User Testing

How did you test whether your design addresses user needs? How did you involve your users? What were their reactions? What were your observations of users interacting with your design?

- **How we tested for user needs**
  - We had a couple of test subjects complete a test lab manual.
  - The manual makes the user go through all the features of Cy-Vital
  - We took notes on the process and made the user fill out an experience survey
- **User Reactions:**
  - Users typically describe our GUI as simple and intuitive when switching between sensors.
  - Ensure data integrity during transfer between modules.
- **Observations:**
  - Users often tried to change raw signal filters without understanding what it would do. This led us to take out the ability to change these settings as it makes the program more complicated.

5.8 Results

- **Accuracy & Reliability** — All sensors met or exceeded the accuracy target. No data-loss events were recorded during stress testing.

- **Performance** — Real-time processing kept total pipeline latency under a reasonable time requirement, ensuring responsive biofeedback.

- **User Satisfaction** — Test participants consistently described the interface as “simple” and “intuitive,” validating the design changes that removed direct filter manipulation.

Compliance Statement

All verified results demonstrate that Cy-Vital conforms to the client’s specifications for data integrity, real-time performance, and usability. The system is ready for final hand-off and deployment in the BME 3500 teaching labs.

6 Implementation

Currently, our implementation is focused on establishing the essential functionalities of the system. Below, we outline the progress made in implementing the key components: **physiological signal processing**, **data acquisition**, **signal processing**, and **circuit integration.**

1. Biological Signals

We have begun working on the following sensors:

- **Electrocardiogram (ECG):** Captures electrical activity of the heart.
- **Electromyogram (EMG):** Monitors electrical activity in muscles.
- **Pulse Oximeter:** Measures oxygen saturation levels.
- **Reaction Time:** This measures how fast user can react

Each of these sensors captures analog signals that require digital conversion for software processing. This step involves ensuring minimal noise and distortion during the signal capture phase, which is critical for accurate analysis.

Multiplexer Integration: Implemented a 16-1 multiplexer for switching between each of the 4 sensors and making the GUI able to switch easily between the sensors (whilst maintaining all backend requirements– handling context changes between sensors, ect.).

2. Data Acquisition

The data acquisition system bridges the gap between sensors and the software interface. Current progress includes:

- Configuring sampling rates to meet the requirements of real-time data analysis while preserving signal integrity.
- Testing sensor connections to ensure compatibility and robustness under various conditions.

This DAQ system ensures accurate, reliable, and synchronized data input for the signal processing module.

I2C connection: Established I2C sensor communication for Pulse Oximeter sensor. This is because the sensor works with this connection. Adjusted software and hardware to meet connection requirements.

3. Signal Processing

Our implementation of the signal processing module includes:

- **Filtering Techniques:** Preliminary application of low-pass and high-pass filters to reduce noise and artifact signals from raw data.
- **Signal Transformation:** Converting digitized signals into user-readable formats, such as waveforms for ECG and EEG data.

Example code snippet of data analysis

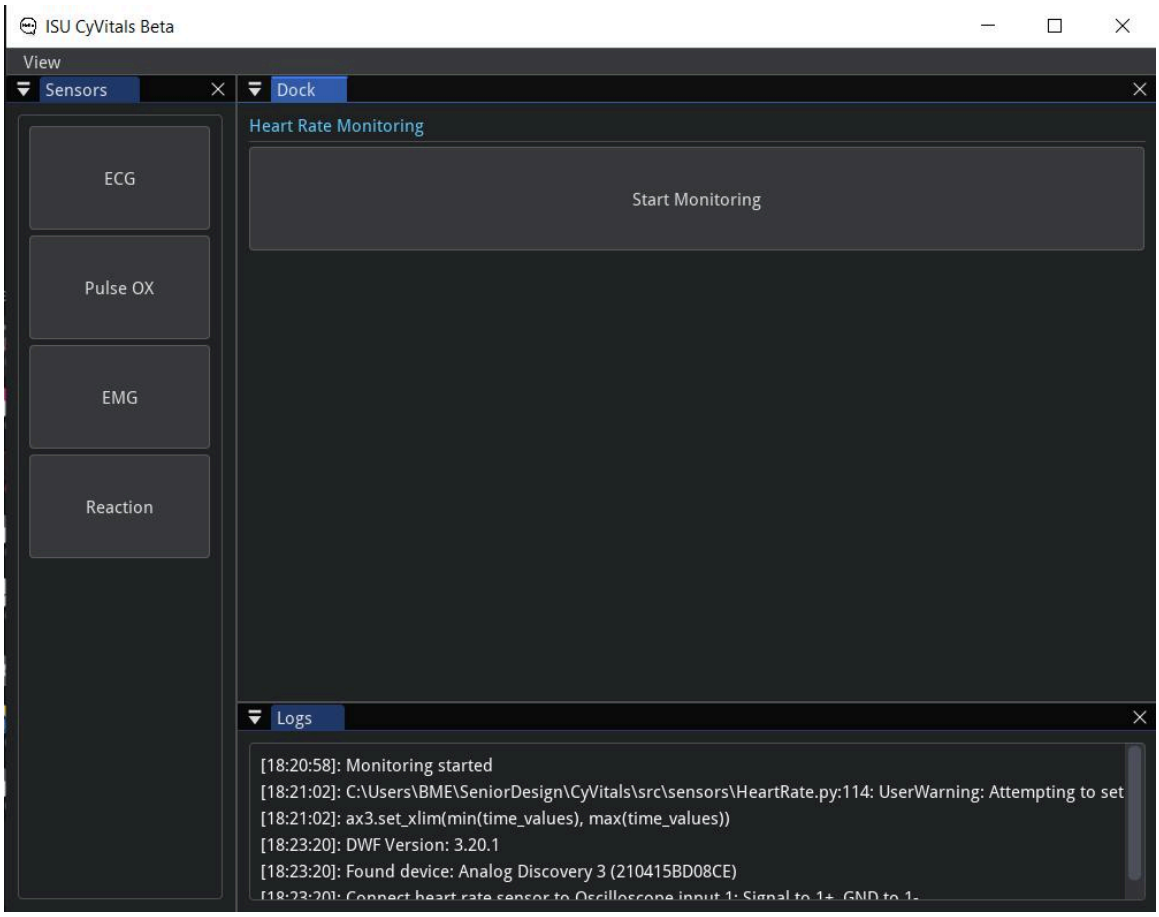
- **Real-Time Processing:** Prototype algorithms for displaying processed signals in real-time on the GUI.

```
status = scope.read_status(read_data=True)

# Get data for channel 1 only (index 0)
channel = scope.channels[0]  # Channel 1
new_samples = np.array(channel.get_data())

# Calculate metrics
dc = np.average(new_samples)
dcrms = np.sqrt(np.average(new_samples**2))
acrms = np.sqrt(np.average((new_samples - dc) ** 2))

print(f"CH{channel.index + 1}: DC:{dc:.3f}V DCRMS:{dcrms:.3f}V ACRMS:{acrms:.3f}V")

# Update waveform plot
x = np.arange(len(new_samples))
line1.set_data(x, new_samples)

# Detect peaks (heartbeats)
peaks, _ = find_peaks(new_samples, height=1.92, distance=200, prominence=0.01)

# Plot detected peaks
if len(peaks) > 0:
    peaks_plot.set_data(peaks, new_samples[peaks])
else:
    peaks_plot.set_data([], [])

# Calculate heart rate (BPM)
current_time = time.time() - start_time
time_values.append(current_time)

# Add newly detected peaks to peak_times
for peak_idx in peaks:
    peak_time = current_time - (len(new_samples) - peak_idx) / sample_rate
    peak_times.append(peak_time)
```

GUI implementation

These steps ensure that the raw biological signals captured by the sensors are accurately cleaned, processed, and presented for analysis or educational purposes.

Integration and Next Steps

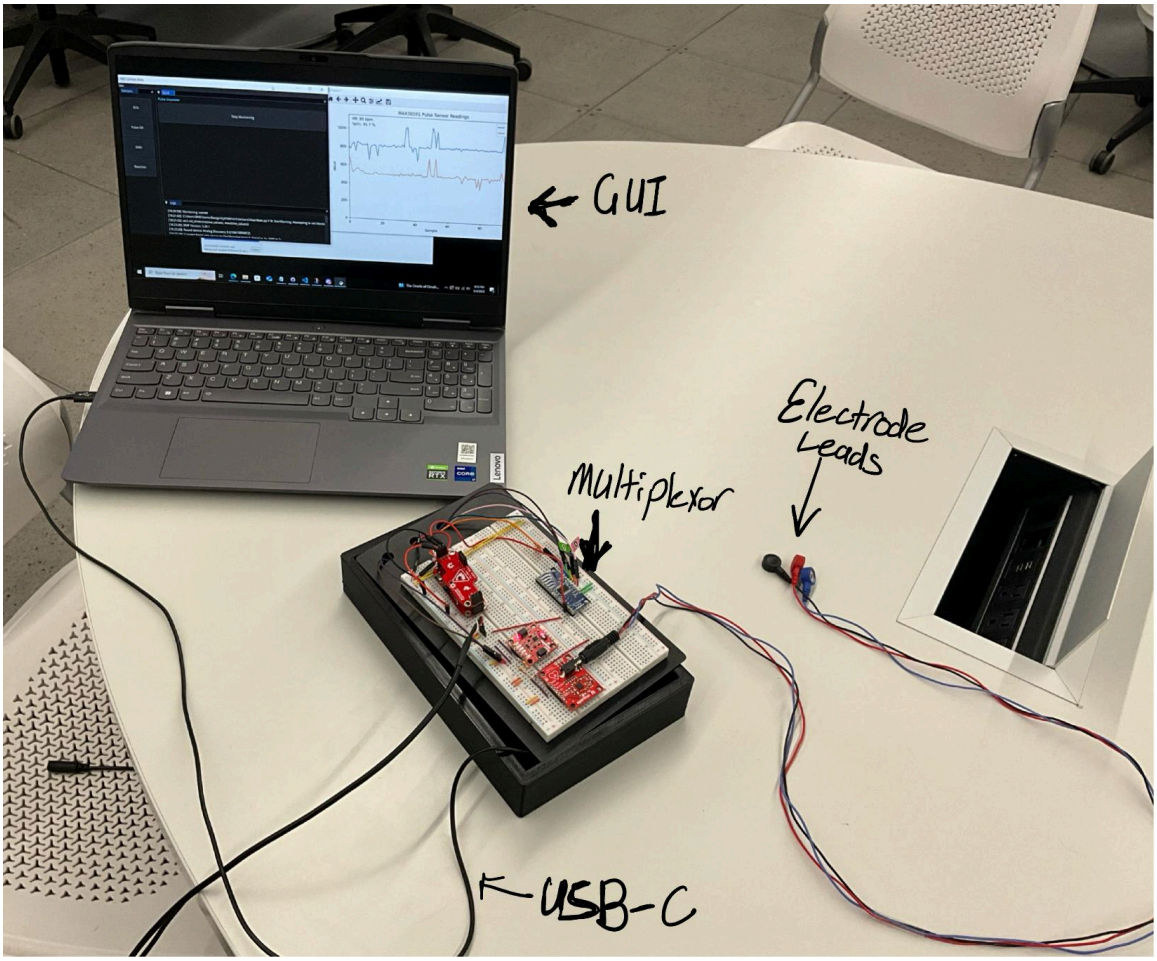


Our design uses a 16-1 multiplexer to control which sensor to read from. Each sensor is powered using a 3.3V supply of power coming from the AD3 DAQ.

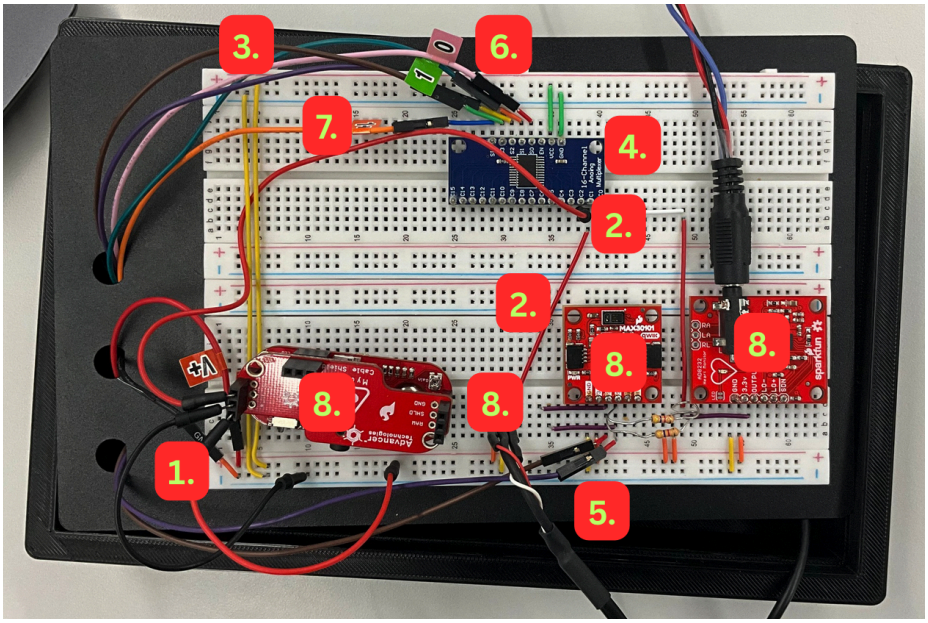
6.1 Design Analysis

What works well - The 16:1 multiplexer integration has been one of the best features we have. It seamlessly works to switch between each sensor and works well the the custom GUI we have made. All our sensors work well to collect and transfer data to the DAQ and into the computer.

What does not work well - The signal processing is not the best for our project right now. Our data is noisy at times, and the data does not look the cleanest. There is a decent amount of variability and occasional distortion. This is because any movement with the sensor can cause noise that is difficult to process.



High level system implementation



Sensor hardware implementation

Wiring Chart

Node	Wire description
1	V++ and GND
2	16-to-1 Mux inputs
3	V++ and GND extension
4	16-to-1 Inputs
5	SDA and SCL input lines for pulse oximeter
6	Select lines for 16-to-1 mux
7	Output of Mux to input of Analog Discovery 3
8	Our 4 sensor modules ECG, EMG, Pulse Oximeter, and Reaction Time

7 Ethics and Professional Responsibility

In this section, we outline our approach to professional responsibility, focusing on **engineering ethics and professional conduct** as defined by the **IEEE Code of Ethics** and **IEEE 1016-1998**. Our project prioritizes maintainability, clarity, and ethical considerations in software design and implementation, ensuring a robust system for data analysis and presentation. These practices aim to uphold the principles of transparency, sustainability, and user well-being.

Over the course of this project, our understanding of ethical responsibility has changed in many meaningful ways. One of the big things is team dynamics and ethical culture. Through the collaborative time we have together, we have seen the value in fostering a good working environment where ethical concerns are openly discussed. Especially since we are working with medical technology, this data is highly sensitive to each person.

7.1 Areas of Professional Responsibility/Codes of Ethics

Below is a table addressing the seven areas of professional responsibility, incorporating primary goals stated from **IEEE 1016-1998** to further development for software documentation and maintainability:

Area of Responsibility	Definition (in our own words)	Relevant IEEE Code of Ethics/1016-1998 Item	How Our Team Adheres
Work Competence	Deliver high-quality, maintainable solutions.	"Maintain and improve technical competence" (IEEE Code of Ethics, Item 6); Emphasize clarity and maintainability in software design (IEEE 1016-1998).	Implemented structured design documentation to guide future modifications.
Financial Responsibility	Provide cost-effective solutions without sacrificing quality.	"Avoid real or perceived conflicts of interest" (Item 4).	Selected modular components and reusable code to reduce overall costs.
Communication Honesty	Report progress and limitations clearly and transparently.	"Be honest and realistic in stating claims" (Item 3).	Used IEEE 1016-1998 practices for documenting software

			functionality and limitations.
Health, Safety, Well-Being	Ensure the system prioritizes user safety and data security.	“Hold paramount the safety, health, and welfare of the public” (Item 1).	Designed features to safeguard sensitive user data and ensure accurate signal processing.
Property Ownership	Respect intellectual property and user data ownership.	“Avoid injuring others, their property, or reputation” (Item 8).	Properly attributed external libraries and ensured user data remains locally controlled.
Sustainability	Minimize environmental impact in design and use.	“Strive to comply with sustainable development practices” (Item 10).	Used modular design to reduce hardware waste and documentation practices for efficient updates.
Social Responsibility	Enhance educational tools for broader societal benefit.	“Enhance the quality of life of humankind” (Item 1).	Developed software aligned with educational goals, ensuring accessibility and usability.

Performance Evaluation

Area of Strength:
Our team excels in **Work Competence** by strictly following the practices outlined in **IEEE 1016-1998** for acceptable software documentation. This includes defined modules, interpretable diagrams, and required comments in code to enhance maintainability and encourage continuous development. Thus, the project can continue to develop and shape into any required standard.

Area for Improvement:
We need to improve in **Sustainability**, due to the usage of software and hardware within the project. While a modular design helps reduce waste, we plan to enhance this by

supplementing required components with eco-friendly alternatives and encouraging the usage of energy-efficient programming practices.

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Ensure bioinstrumentation designs enhance diagnostic accuracy (e.g., more reliable EKG or pulse oximetry readings).	Avoid imposing practices or technologies that could harm specific cultural or social groups	Ensure design is able to be used by anyone that is taking the lab	Develop bioinstrumentation solutions that will work on a diverse population
Global, Cultural, and Social Considerations	Address health disparities by designing devices that meet the needs of anyone that might be taking the lab	Avoid imposing practices or technologies that could harm specific cultural or social groups	Ensure the design respect cultural practices and individual preferences	Develop bioinstrumentation solutions that distribute benefits to all our users including lab students, lab TA, and the professor.
Environmental	Design devices that support sustainability, such as reusable or recyclable components	Minimize the environmental footprint by reducing hazardous waste or energy consumption	Offer eco-friendly options to users without compromising functionality or forcing choices.	Ensure the environmental benefits of your design do not come at the expense of specific regions or habitats
Economic	Design devices that will simulate how bio instrumentation is done in the real world	Avoid disruption to existing healthcare systems or economies	Provide cost-effective design options to accommodate a range of budgets, ensuring users can make informed financial decisions.	Avoid undue financial burdens on students by keeping costs reasonable and fair.

Area for Improvement - Environmental Beneficence: We don't prioritize sustainability by selecting components that are energy efficient and low-cost. This reduces electronic waste and encourages sustainable design practices. By doing this, we could help contribute to responsible environmental solutions. To improve this area, we can look into selecting components that are energy efficient or have multiple sensors in one.

Strength - Economic Respect for Autonomy: Our design is currently very cost-effective. Each of the sensors can be interchanged with other sensors if necessary. If something is too expensive then we can switch it out for a cheaper sensor. Our code is also open source, so it is free to use.

Our team values the following virtues:

1. **Honesty:** Transparent reporting of limitations and results.
2. **Responsibility:** Adherence to ethical standards and user expectations.
3. **Respect:** Valuing all contributions and prioritizing user autonomy.

How we support these virtues as a team:

- Documented and communicated design limitations, accurately reported testing data, and provided realistic performance expectations to stakeholders.

- Followed IEEE 1016-1998 guidelines, prioritized user safety with local data storage, and incorporated feedback to align the design with user needs.
- Encouraged inclusive discussions, valued all contributions, and ensured user autonomy by giving full control over their data.

Individual Reflection:

Sajan Patel:

- **Virtue Demonstrated: Honesty**
    - **Why Important:** Honesty builds trust among team members and our client ensuring credibility and reliability in our project's outcomes.
    - **Example:** I transparently explained which open-source libraries I used for my code.
- **Virtue to Demonstrate: Trust**
    - **Why Important:** Trust is important because it helps strengthen the bond between teammates, helping with overall work quality.
    - **Future Action:** I will try to start trusting more so that we can function as a well-oiled machine and become better friends in the future.

Daniel Karpov:

- **Virtue Demonstrated:** Courageous
    - **Why Important:** Courage is important because it means that risks are not afraid to be taken. It is also a virtue that can allow for taking blame for mistakes and acknowledging when you're wrong.
    - **Example:** Asking clarifying questions even if it means showing that you might not understand a topic as much as other people.
- **Virtue to Demonstrate:** Clear and thorough documentation

    - **Why Important:** This is important because it allows for other members of the team to understand what was being worked on and what was changed. It gives more structure to the project and allows people working on it to be more organized.
    - **Future Action:** In the future it is important to document everything new that is added to the project and work hard in order to document even the smallest changes.

Chuck Mallek:

- **Virtue Demonstrated: Diligence**

    - **Why Important:** Diligence is important because it highlights spending time to complete menial tasks that can be otherwise procrastinated and forgotten about. This reinforces my helpfulness and timeliness as a group member.
    - **Example:** I always developed our lightning talks and design documents before the due date instead of waiting till the last minute to work on them.
- **Virtue to Demonstrate: Patience**
    - **Why Important:** This is important because it allows others in my group to work at their own pace, which would improve overall work quality.
    - **Future Action:** I will be more patient with my group members so that they can turn in their best work without being rushed.

Tysung Beresford:

- **Virtue Demonstrated: Hard Working**
    - **Why Important:** This is important because it pushes the other team members to increase their level to match.
    - **Example:** Wanted to stay longer to work on a part of the project and everyone else ended up staying.
- **Virtue to Demonstrate: Cooperative**
    - **Why Important:** Cooperativeness is important because it allows for collaboration between group members
    - **Future Action:** I will communicate with the team when I plan on working on the projects

Jay Patel

- **Virtue Demonstrated: Trust**
    - **Why Important:** Trust is important because it allows me to trust other members with their own work and not worry about holding them to their word.
    - **Example:** I trust my other team members to do their part to split the work equally between everyone.
- **Virtue to Demonstrate: Leadership**
    - **Why Important:** Leadership is important because it allows group members to execute their roles in a plan.
    - **Future Action:** I will be more of a leader in the future and help lead others by dividing work up evenly and laying out a plan for future goals.

8 Conclusions

8.1 Summary of Progress

The CyVital project made significant progress toward creating an affordable and user-friendly educational platform for biomedical engineering students. We have developed a modular system– integrating a data acquisition unit, physiological sensors and a Python-based graphical user interface for real-time data visualization and analysis. Key accomplishments include successful ECG sensor testing, initial signal processing and iterative testing of system components. Constraints such as hardware limitations, time and resource restrictions have posed challenges, particularly with the transition between older hardware into the newer hardware (Analog Discovery 3). While the project remains as a successful accomplishment, it still retains a strong potential for improvement and numerous features.

8.2 Value Provided

CyVital delivers significant value by addressing the core challenges of affordability, accessibility, and educational effectiveness in biomedical instrumentation education.

User Needs Addressed

1. **Biomedical Engineering Students**
    - *Hands-On Learning*: CyVital's modular sensors (ECG, EMG, Pulse Oximeter) and Python-based GUI enable students to interact with real-world biomedical signals.
    - *Reduced Complexity*: The GUI's intuitive design minimizes need prior knowledge compared to other software
2. **Lab Teaching Assistants (TAs)**
    - *Streamlined Instruction*: All modules are pre-built under our "simple" policy
    - *Scalability*: The multiplexer supports up to 16 sensors
3. **Course Professors**
    - *Curriculum Alignment*: CyVital's open-source codebase allows students to continue working
    - *Cost Savings*: Significantly cheaper

Problem Resolution

CyVital directly tackles the limitations of outdated, expensive tools:

- **Affordability**: Use default brand to save money
- **Modernization**: The Python GUI replaces legacy applications
- **Signal Integrity**: Digital filtering to reduce issues
- **Scalability**: The 16:1 multiplexer and modular design allow seamless integration of new sensors

Broader Context

1. **Public Health & Safety**
    - CyVital equips students with practical skills to develop medical devices
2. **Economic Impact**
    - Low-cost access to biomedical instrumentation allows all backgrounds to have hands-on experience
3. **Sustainability**
    - Modular components extend the system's lifespan
4. **Educational Innovation**
    - CyVital's open-source framework encourages collaboration.

8.3 Next Steps

1. Advanced Signal Processing

- **Machine Learning for Noise Reduction**: Implement data models to smooth waveforms at a more consistent rate
- **Automated Diagnostics**: Develop algorithms immediately find issues

Importance: Enhances educational value

2. Expanded Sensor Diversity

- **Additional Sensors**: Integrate EEG and other similar sensors
- **Wireless Connectivity**: Replace USB with Bluetooth/Wi-Fi modules to remove issues of wiring

Importance: Diversifies lab modules fosters stronger levels of knowledge

3. Enhanced Features

- **Cloud-Based Collaboration**: Create a platform for students to upload, share, and compare datasets
- **Virtual Labs**: Develop remote access for students who are unable to attend in person

Importance: Promotes collaborative learning

4. Hardware Optimization

- **Custom PCB Design**: Consolidate all components into a single-do all box
- **Low-Power Modes**: Implement energy-efficient components

Importance: Reduces carbon footprint and energy costs

5. Development

- **Open-Source Ecosystem**: Allow students to access code and change sensor data
- **Curriculum Integration**: Collaborate with universities

Importance: Supports community driven support

9 References

1. Analog Devices. *Single-Lead, Heart Rate Monitor Front End Data Sheet AD8232*. Norwood, MA:Analog Devices,2013.[Online].Available:https://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf
2. Measurement Computing. *USB-1608G Series Multifunction DAQ Devices User's Guide*. Norton, MA: Measurement Computing, 2012. [Online]. Available: https://www.mccdaq.com/pdfs/manuals/USB-1608G-Series.pdf
3. IEEE Standards Association, "IEEE Standard 1012-2016: System and Software Verification and Validation," *IEEE Xplore*, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/1012-2016
4. Restackio, "Ieee Software Coding Standards," 2024. [Online]. Available: https://www.restack.io/ieee-software-coding-standards
5. Matplotlib Development Team, *API Reference: Version 3.9.3*. [Online]. Available: https://matplotlib.org/stable/api/index.html. Accessed: Dec. 7, 2024
6. PIA Group, *BioSPPy: Biomedical Signal Processing in Python*. [Online]. Available: https://github.com/PIA-Group/BioSPPy. Accessed: Dec. 7, 2024

10 Appendices

Appendix 1 – Operation Manual

Overview

CyVital GUI is an application built using the imgui_bundle framework. It provides an interface for visualizing sensor data, logging user interaction, and offering a responsive UI with dockable windows.

Main Features

1.Navigation Sidebar

- Located on the **left panel**
- Provides buttons to switch between views:
    - **ECG**
    - **Blood O2**
    - **EMG**

Clicking any button updates the main content area with the corresponding sensor interface.

Views Description

2. ECG View

- Displays:
    - A **Counter slider** to set a value between **0 - 100**
    - A **Reset Counter** button to set the counter back to 0
    - During runtime, ECG function will plot peaks and implement student designed algorithms to calculate heart rate.

3. Blood O2 View

- Displays:
    - Display level indicator plot for blood O2 levels.

4. EMG View

- Displays:
    - **Counter value** (as set in ECG view)
    - **Input text** (as entered in Blood O2 view)
- Every time the EMG view is opened, the current input text is **logged**

5. Logs Panel

- Located at the **bottom** of the screen
- Logs messages with **timestamps**
- Automatically displays the **input text** used in the EMG view
- Keeps the **latest 1000 entries** (older ones are discarded to manage memory)

6. Window Settings

- **Title:** CyVital GUI
- **Window Size:** 800 x 600
- **Menu Bar:** Minimal interface

How to Run

1. Ensure dependencies are installed:
   imgui_bundle, typing, datetime
2. Run the script using Python:
    a. Ex: python Main.py

Appendix 2 – alternative/initial version of design

- During the development of the CyVital project, our team had multiple shifts in scope, design, hardware, and development of our project. Initially, we developed the project based on MCC hardware, which was later discovered to be incompatible with multiple objectives we had established, such as concurrent data analysis and sampling. After discussion, our team decided to move platforms to the Analog Digital 2 acquisition device, which improved compatibility with our sensors and provided much better sampling rate controls. A third hardware change was implemented to move from the AD2 to the Analog Discovery 3 data acquisition unit. This final change was implemented to add native USB C support and allow for future development of the platform using up to date hardware. For software development, we tested and prototyped multiple libraries for the GUI. Initially we tested multiple python libraries such as PyGUI, and TKinter. After testing, the team decided on the IMGUI python library, which offered the best control of frame rates and was easiest to implement based on our configuration.

Appendix 3 – Other considerations

- N/A

Appendix 4 – Code

- lil-skies/CyVitals: CyVitals is dedicated to providing modular hardware in the form of sensors and its corresponding software counterpart to read, analyze and display data seen within physiological sensors.

Appendix 5 – Team Contract

Team Members

- Daniel Karpov - Software Engineer
- Jay Patel - Software Engineer
- Ty Beresford - Software Engineer
- Sajan Patel - Software Engineer
- Chuck Mallek - Computer Engineer

Required Skill Sets for Your Project

- Data Acquisition and Signal Processing
    - Retrieve and process data from physiological sensors like heart rate, EEG, and EMG.
- Embedded Systems Programming
    - Develop software to control hardware and process real-time sensor data.
- Signal Interpretation and Analysis
    - Analyze physiological signals to teach students how to extract meaningful insights.
- Hardware-Software Integration

    - Ensure seamless functioning of all sensors within a unified system.
- User-Centric Design for Education
    - Create an intuitive platform for students with varying technical expertise.
- Biomedical Engineering Fundamentals
    - Relate sensor operations to real-world healthcare applications.
- Team Collaboration
    - Coordinate development and share progress among team members.
- Testing and Validation
    - Ensure the system's safety, accuracy, and usability for educational purposes.
- Machine Learning
    - Expand future capabilities for real-time signal pattern detection and analysis

Skill Sets covered by the Team

- Daniel Karpov
    - Data Acquisition and Signal Processing
    - Signal Interpretation and Analysis
    - User-Centric Design for Education
    - Testing and Validation
    - Team Collaboration

- Jay Patel
    - Data Acquisition and Signal Processing
    - Team Collaboration
    - User-Centric Design for Education
    - Signal Interpretation and Analysis

- Ty Beresford
    - Embedded Systems Programming
    - Testing and Validation
    - Hardware-Software Integration

- Sajan Patel
    - Embedded Systems Programming
    - Testing and Validation
    - Hardware-Software Integration

- Chuck Mallek
    - Signal Interpretation and Analysis
    - User-Centric Design for Education
    - Hardware-Software Integration

Project Management Style Adopted by the team

The team used an agile approach where we would set weekly goals and then work towards them before we moved onto the next goals. This allows us to have an iterative approach towards the project making it more manageable.

Individual Project Management Roles

- Daniel Karpov - Data Processing
- Jay Patel - Data Processing
- Ty Beresford - Full stack Software
- Sajan Patel - Full Stack Software
- Chuck Mallek - Physical and Electrical Design

Team Contract

Team Members:

1. Ty Beresford
2. Sajan Patel
3. Jay Patel
4. Daniel Karpov
5. Chuck Mallek

Team Procedures

1) Day, time, and location (face-to-face or virtual) for regular team meetings:
    a) Three days a week during regular class hours in person; additional virtual meetings via Discord on an as-needed basis.
2) Preferred method of communication updates, reminders, issues, and scheduling:
    a) Primary: Discord group chat and calls.
    b) Secondary: Google Drive for documentation, phone numbers for backup communication.
3) Decision-making policy:
    a) Consensus vote, with majority vote as a fallback.
4) Procedures for record keeping:
    a) Each team member is responsible for recording their contributions. Records to be shared via Google Drive.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

    a. Attendance and active participation are mandatory.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. Complete assigned tasks promptly and communicate any delays proactively.
3. Expected level of communication with other team members:
    a. Maintain open communication and respond promptly to messages.
4. Expected level of commitment to team decisions and tasks:
    a. Fully commit to team decisions and tasks.

Leadership

1. Leadership roles for each team member:
    a. Ty Beresford: GUI Development
    b. Sajan Patel: Full Stack Lead
    c. Jay Patel: Backend and Data Analysis
    d. Daniel Karpov: Backend and Data Analysis
    e. Chuck Mallek: Physical Systems Integration
2. Strategies for supporting and guiding the work of all team members:
    a. Regular meetings for updates and problem-solving; proactive communication to assist where needed.
3. Strategies for recognizing the contributions of all team members:
    a. Acknowledge contributions during meetings and ensure equal credit in project milestones.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team:
    a. Daniel Karpov - Data Processing
    b. Jay Patel - Data Processing
    c. Ty Beresford - Full Stack Software
    d. Sajan Patel - Full Stack Software
    e. Chuck Mallek - Physical and Electrical Design
2. Strategies for encouraging and supporting contributions and ideas from all team members:
    a. Open discussion during meetings; equal opportunity to propose ideas.
3. Procedures for identifying and resolving collaboration or inclusion issues:
    a. Discuss concerns in team meetings, escalate unresolved issues to the professor if necessary.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. Develop a high-quality application with a functional GUI for recording, analyzing, and storing sensor data.
2. Strategies for planning and assigning individual and team work:
    a. Weekly meetings to set expectations and assign tasks; roles adjusted based on project needs.
3. Strategies for keeping on task:
    a. Regular check-ins during meetings; track progress via Discord and Google Drive.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    - First offense: Issue an email/text with a description of the infraction and corrective action needed.
2. What will your team do if the infractions continue?
    - Second offense: Notify the professor via email/text.
    - Third offense: Schedule an in-person meeting with the offender and the professor to resolve the issue. If unresolved, consider removal from the team.

a) I participated in formulating the standards, roles, and procedures as stated in this contract.
b) I understand that I am obligated to abide by these terms and conditions.
c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract

1) Daniel Karpov - September 18th 2024
2) Ty Beresford- September 18th 2024
3) Sajan Patel- September 18th 2024
4) Chuck Mallek- September 18th 2024
5) Jay Patel - September 18th 2024