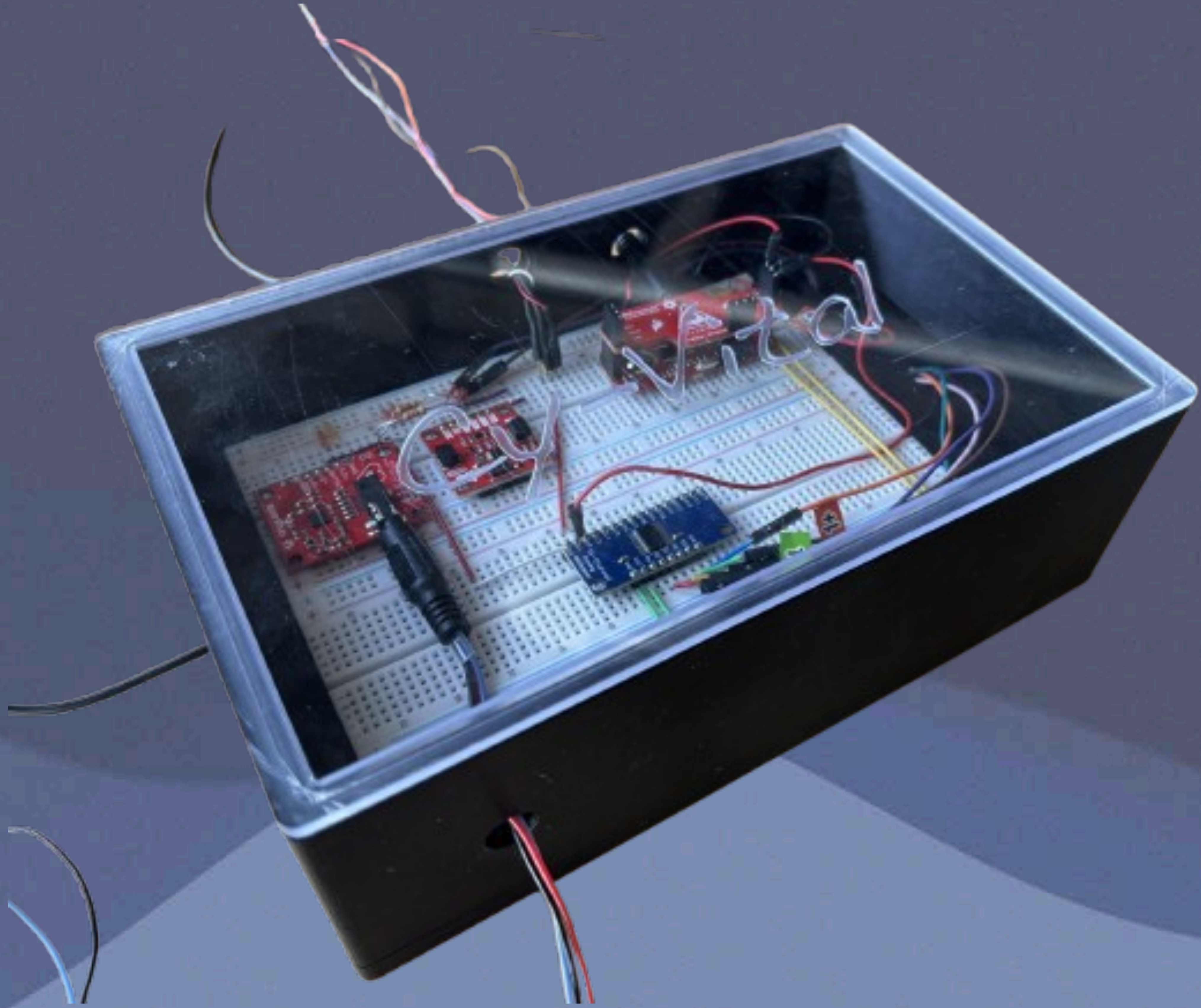
CYVITAL

SENIOR DESIGN GROUP 22



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

Meet the Team

Jay Patel

Software Engineering

Data Analysis

Tysung Beresford

Software Engineering

GUI Development

Sajan Patel

Software Engineering

Fullstack

Daniel Karpov

Software Engineering

Data Analysis

Chuck Mallek

Computer Engineering

Hardware Design

Problem Statement

CyVital

- Next spring, biomedical engineering students enrolled in BME 3500 will need a hands on, interactive lab apply course concepts in practice

- For this lab, a hardware and software kit needs to be developed to teach students how to program, test, and collect data from various physiological signals.

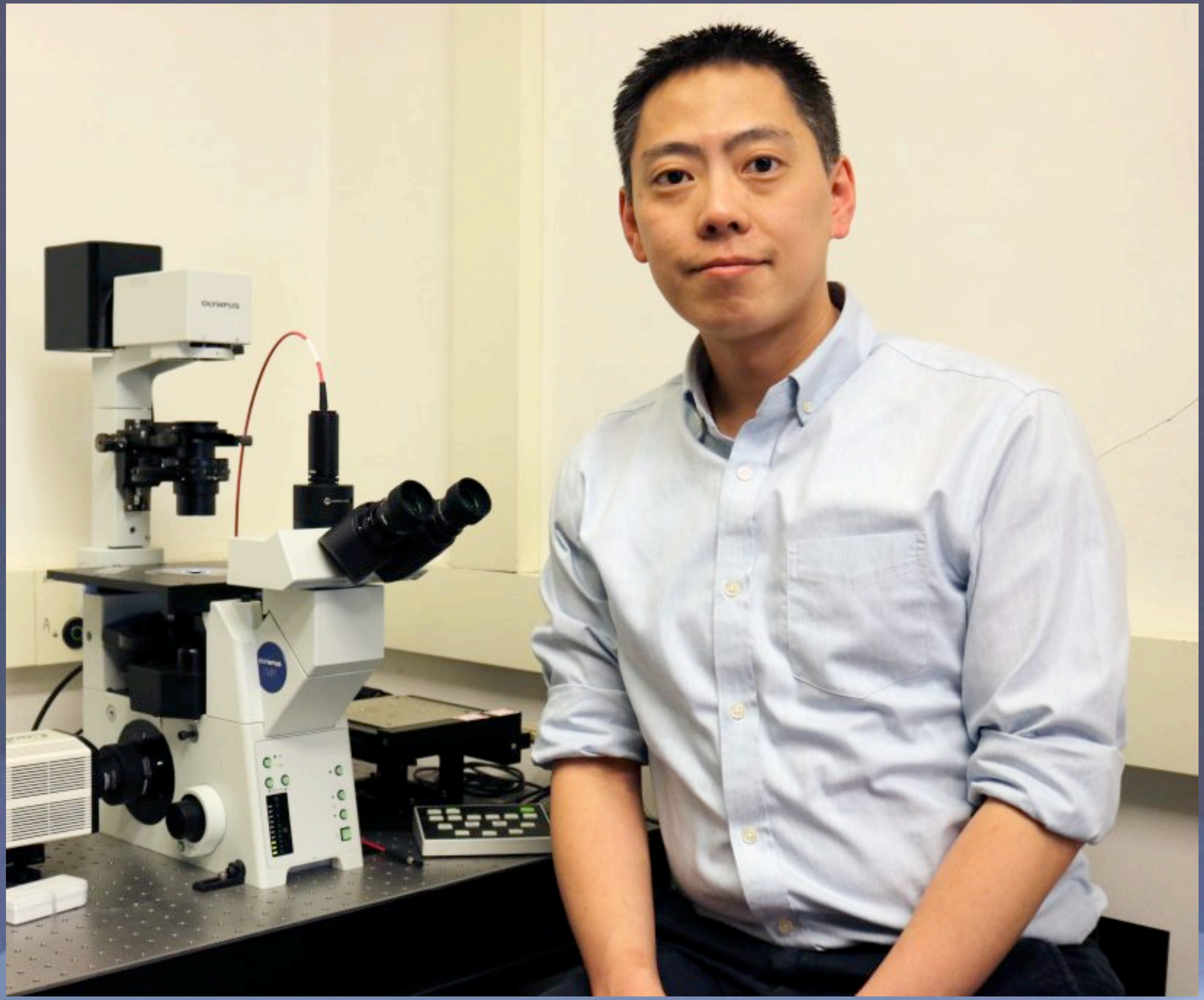
Prereq: *PHYS 2320, EE 4420,* ***and*** *BME 2700*
Introduction to medical devices, sensors, lab equipment, wearable devices, and imaging equipment that are commonly used to measure and analyze biological systems. Fundamental principles of biology, electronics, physics, and engineering that are required to design and build these bioinstruments. Basic concepts of biomedical signal analysis for medical diagnosis and health monitoring. Measurements and analysis of bioelectrical, biomechanical, and biophotonic signals. Simple design of instruments to address a specific biomedical problem. (Typically Offered: Spring)

CyVital

Overview

We worked with our client and faculty advisor, Dr. Meng Lu, for the creation of the BME 3500 lab to enable hands on intuitive learning for the aspiring BME students.

- CyVital is a solution designed to guide students through setting up and implementing their own data analysis programs for various sensors.



Ph.D., Electrical Engineering, University of Illinois (2008)

M.S., Electrical Engineering, University of Illinois (2006)

B.S., University of Science and Technology of China (2002)

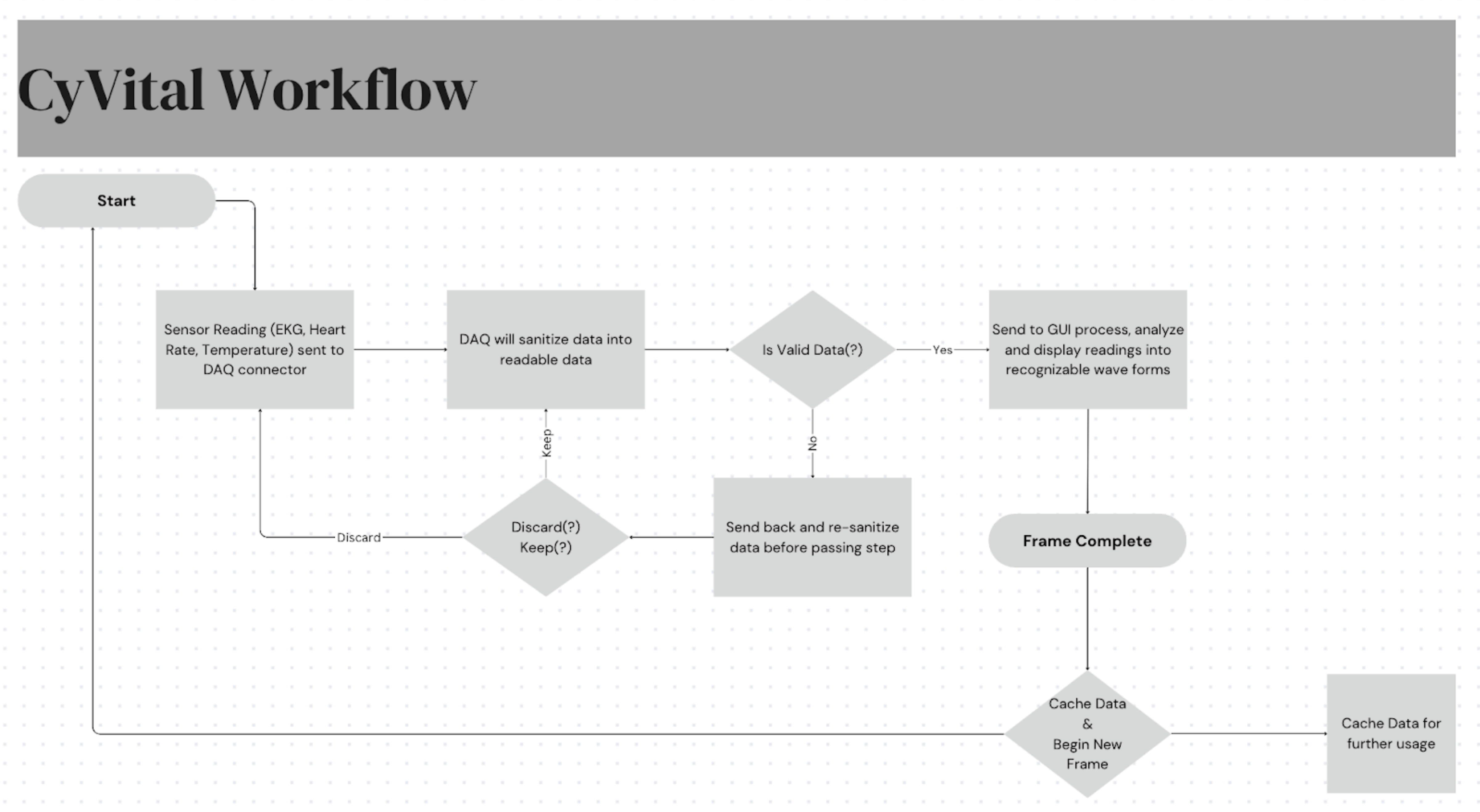
Core Area(s): Electromagnetic, microwave, and nondestructive evaluation; microelectronics and photonics

Department's Strategic Area(s): Materials, devices & circuits; bioengineering

Design

Requirements

- Each sensor analysis method should accurately record and read physiological signals
- The lab must adhere to learning objectives for the class
- The analysis methods should be written in a familiar language for students
- The software should include a GUI to allow students to view data in real time



CyVital is built for Students

- Intuitive software interface to allow students to easily program sensors and view data
- The labs are designed to reinforce and teach core concepts from the course
- Hardware will be standardized to allow students to easily connect and work with a variety of sensors.

CyVital is built for TAs

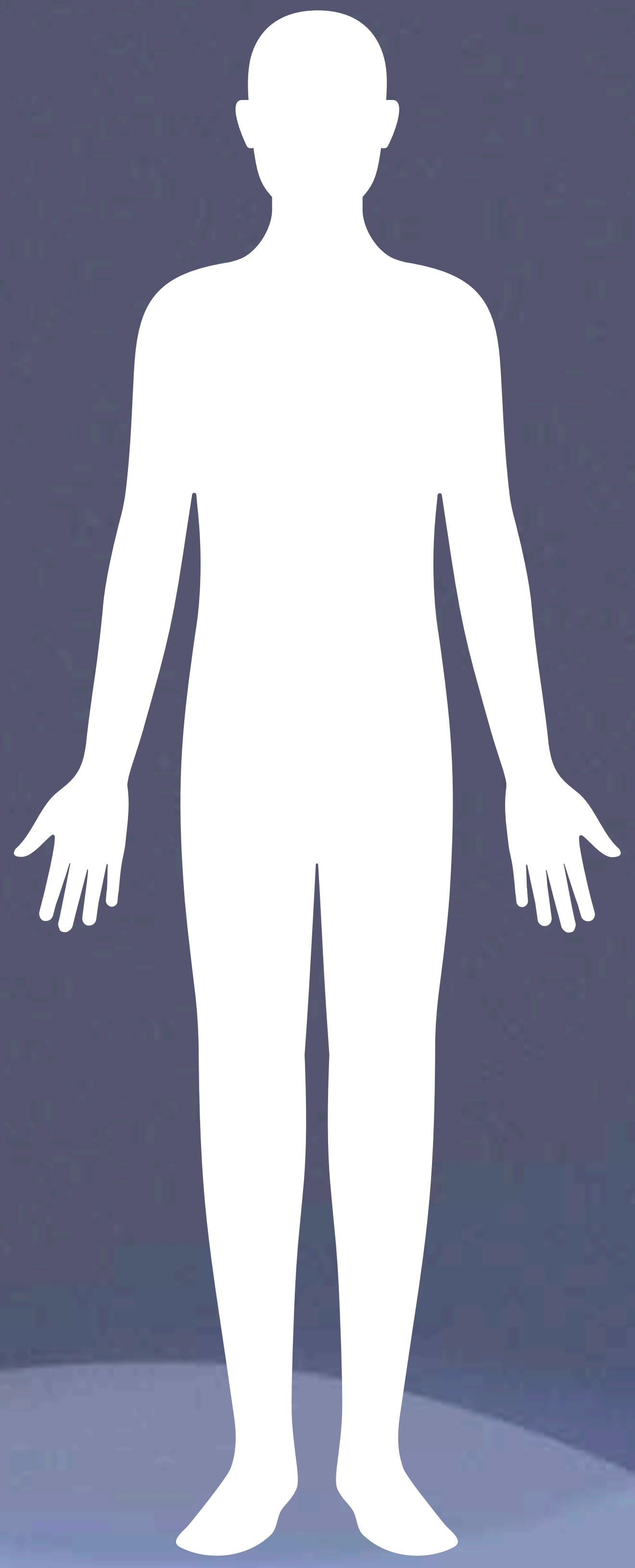
- Clear, easy-to-teach-labs
- A finished and documented answer implementation will be provided
- Include step-by-step instructions and explanations, enabling quick troubleshooting and efficient student support.

CyVital is built for Professors

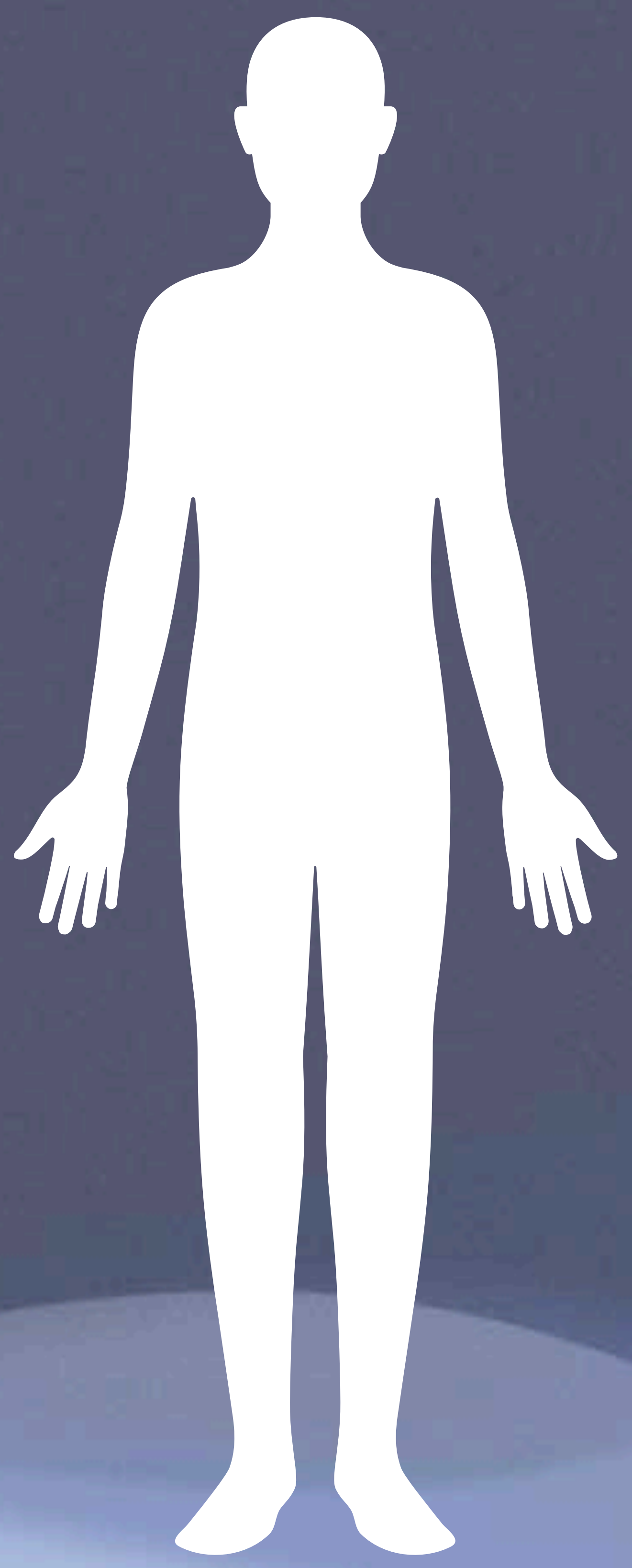
- Labs will be easily gradable with clear rubrics for successful hardware and software implementation.
- Lab modules will align with course curriculum for providing students with hands-on application of theoretical concepts
- Labs can be done in teams which fosters student collaboration within sections

CyVital

How it works

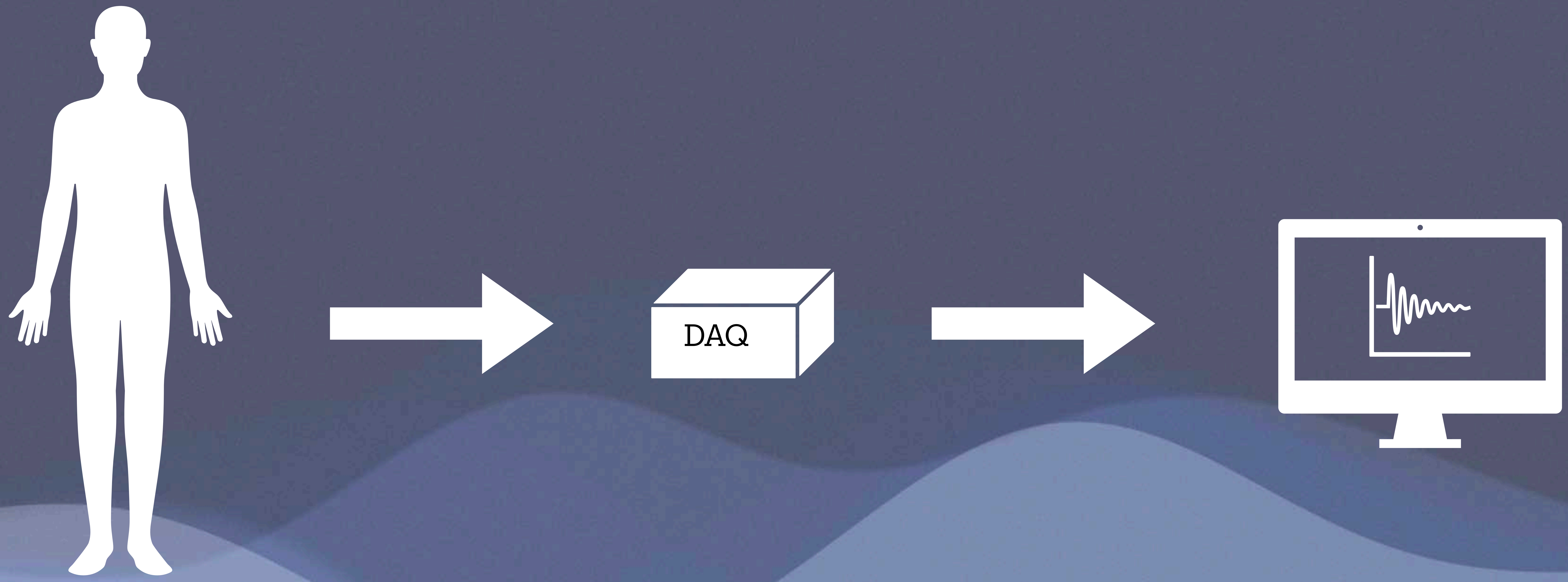


Physiological Signals



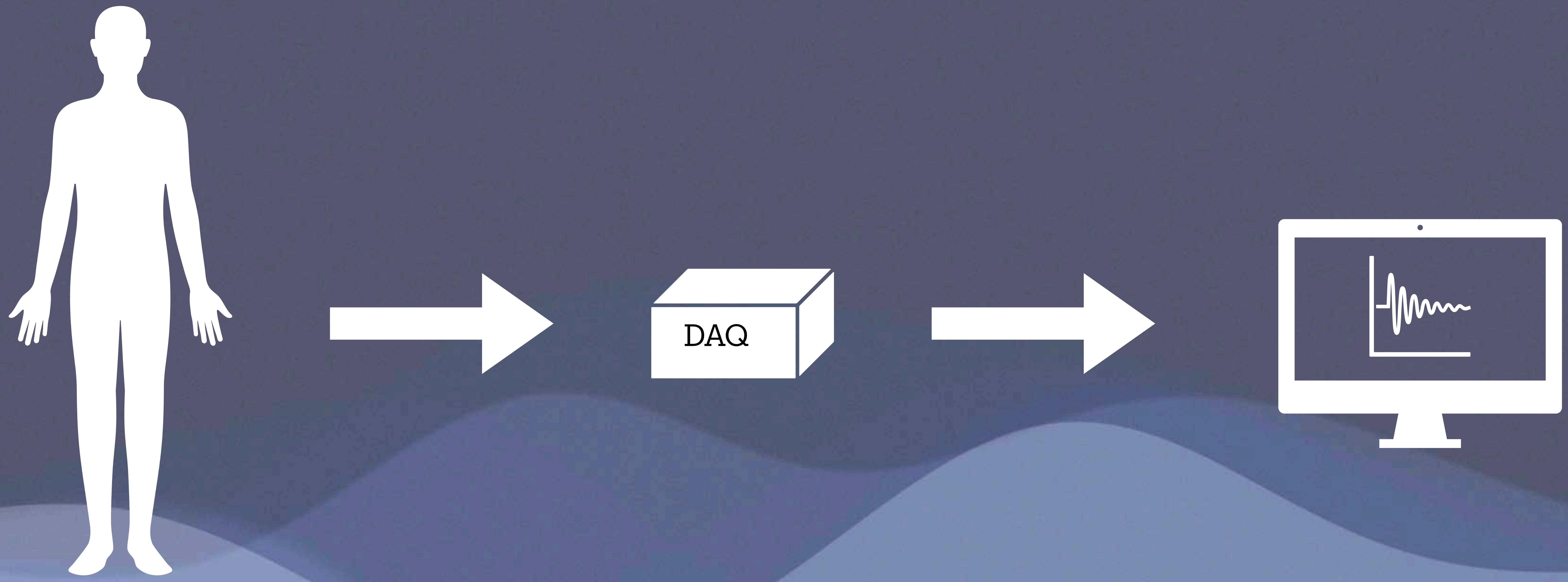
- The human body naturally produces physiological signals that can be detected using analog and digital sensors
  - Thumb Switch (Reaction Time)
  - Electrocardiogram (Heart)
  - Pulse Oximeter (Blood)
  - Electromyography (Muscles)

- These analog signal needs digitalization


DAQ

Data Acquisition

- Converts analog signal into digital format
- Interface between sensors and software
- Ensures accurate data sampling


DAQ

Signal Processing

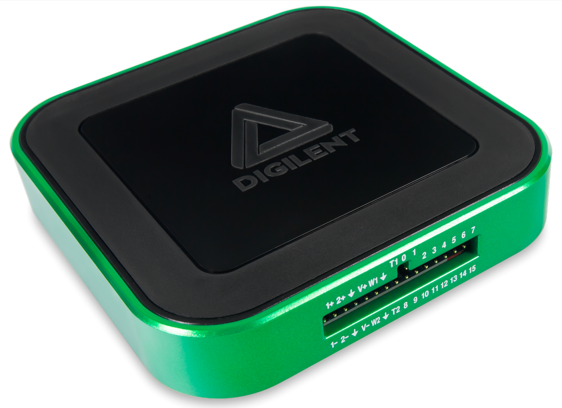
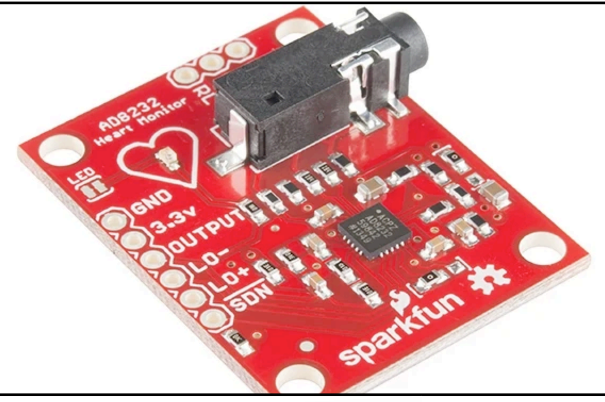
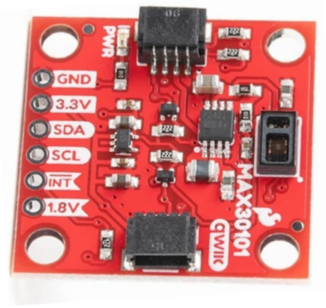
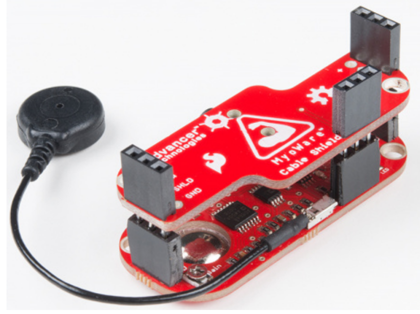
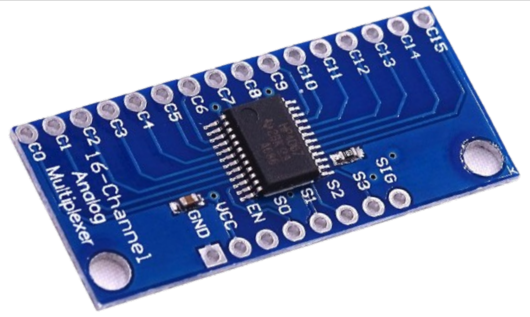
- Filters and cleans raw data
- Converts digital signal into readable formats
- Implements student created processing methods
- Process signals in real time
- Displays and graphs sensor values in GUI

CyVital Implementation

CyVital
Cost Breakdown

Final Breakdown:

- DAQ dominates 74% of the total price
- The 4 sensor modules comes in at roughly 26% of the $510.48
- Our software is free

Component	Model	Image	Price
Data Acquisition Unit	Digilent Analog Discovery 3		$379.00
ECG sensor	SparkFun AD8232 Heart Rate Monitor		$25.00
Pulse Oximeter	SparkFun MAX30101		$35.00
EMG Sensor	SparkFun MyoWare V1		$44.5
16 to 1 Analog Multiplexor	CD74HC4067		$5.99
Reaction time/ miscellaneous	Thumb switch / electrodes, leads, etc...		$20.00
Total:	CyVital Lab Kit		≈ $510.48

Market Research

Cost Comparison

Our current prototype for *4 sensor modules costs* around $500 for hardware

- Existing kits on the market are upwards $10,000+
- Competitors up-charge software over 20 times hardware cost
- Saving Iowa State a lot of money and gives us hands on experience

Description	Item #	Unit	Qty	Your Price	Extended Price	Remove
PHYS TEACHING SYSTEM BASE	470231-634	EA	1	$6,824.00	$6,824.00	×
PSYCHOLOGY SET	470231-638	KT	1	$801.59	$801.59	×
GENERAL BIO SET	470231-642	KT	1	$1,603.09	$1,603.09	×
BP TRANSDUCER	470231-658	EA	1	$335.50	$335.50	×

Item Subtotal =	$9,564.18
Estimated Shipping =	$30.82
Hazard =	$0.00
Special Services / Handling / Fuel Surcharge =	$0.00
Estimated Tax =	$575.71
Estimated Order Total =	$10,170.71

Need tax exempt status?Upgrade to a permanent account to submit your tax exempt IDdocumentation.

Shopping Cart Tools

Shopping Cart Management

- Save as a Shopping List
- Share Your Shopping Cart

Build your cart from

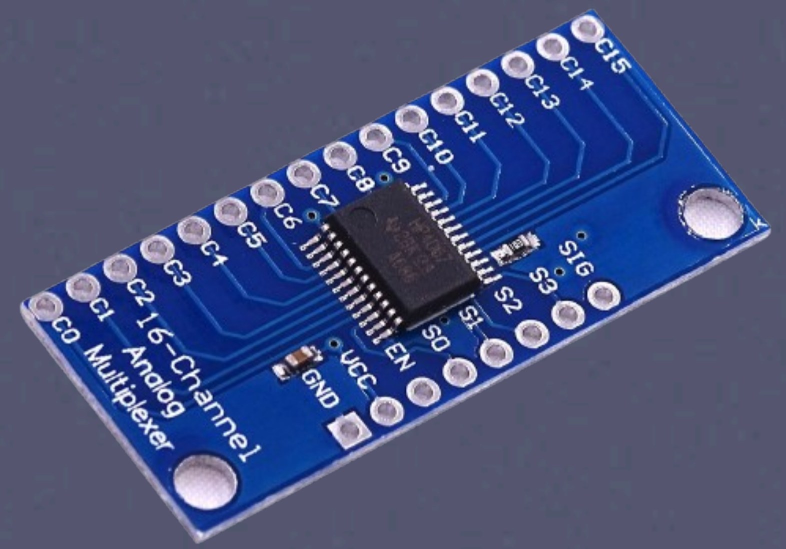
Competitor [iworx] LabKit

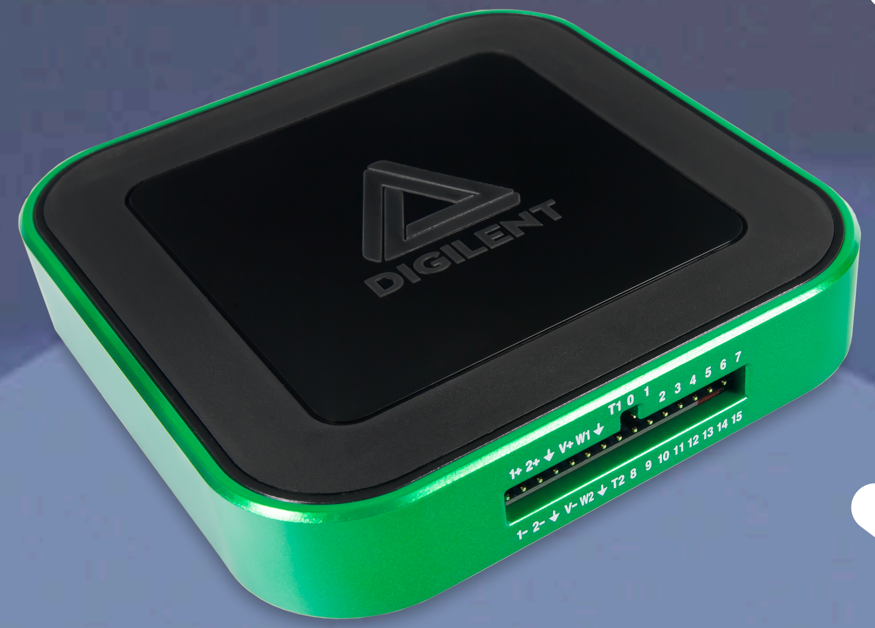
$10,170.71 vs $510.48

Hardware
Implementation

4 Step Process

- Sensor module picks up signal (ECG, EMG, Pulse Oximeter, Reaction time)
- Multiplexor selects which analog signal to output (besides Pulse Oximeter)
- DAQ that converts this signal to digital data (Analog Discovery 3)
- Digital data is transferred to our python Frontend / Backend (USB-C)





Hardware

Issues

Challenge: Sensor Accuracy (heart rate, muscle sensors)

- Sensors require specific location (grounded to a bone)
- Adequate pressure is needed to maintain reliable contact
- Small movements caused signal loss or noise

Solution:

- Used straps and mounts to give pressure and stabilize sensors
- Adjusted gain on our sensors to increase signal amplitude

Hardware
Definite Changes

Last Semester

Our prototype only enveloped ECG signal with almost no modularity

- USB-3.0 connection
- No Housing for our Lab Kit
- Outdated DAQ device

This Semester

Our prototype is more modular with 4 sensors-modules total

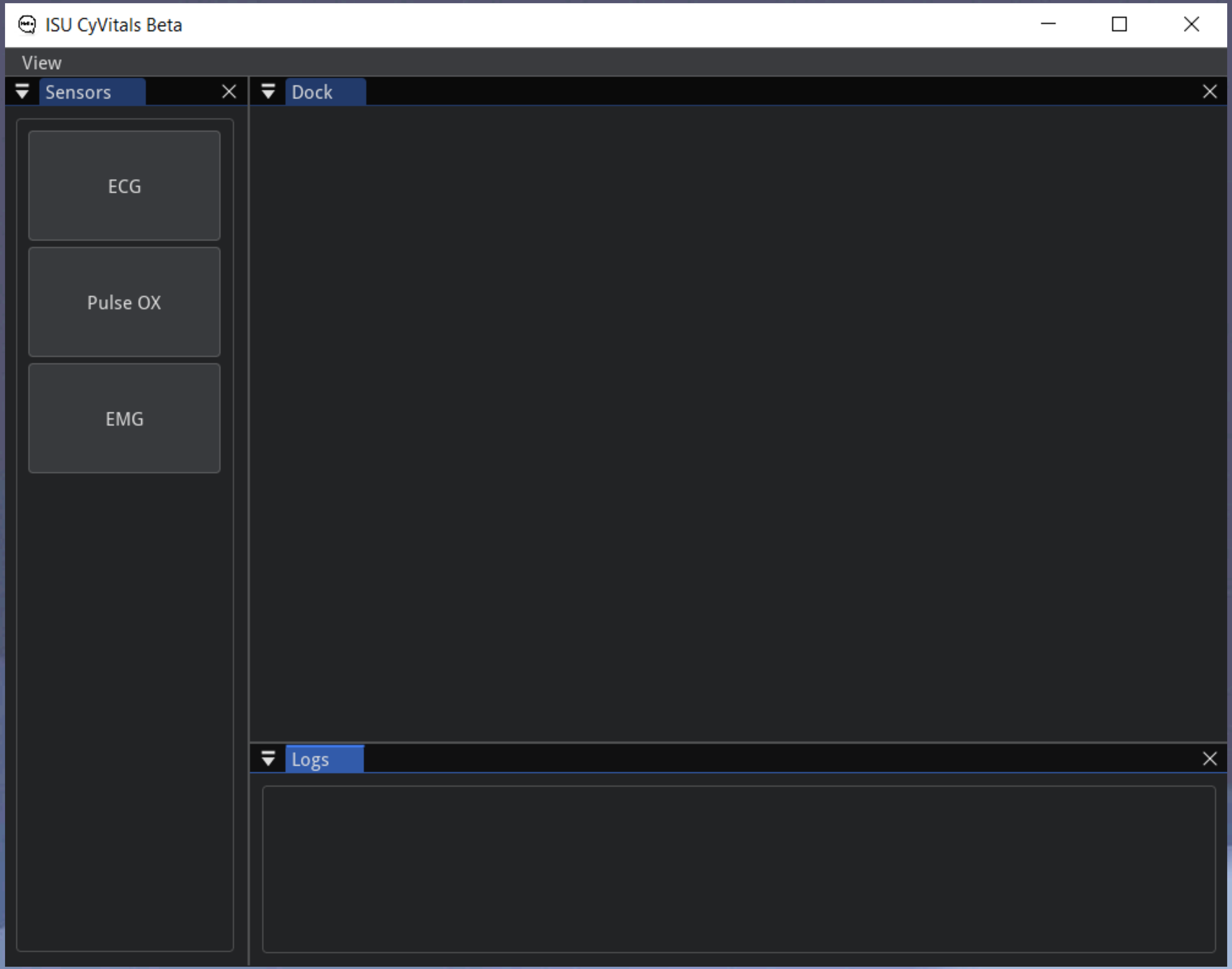
- USB-C for universal connection
- Housing for our Lab Kits
- 16-to-1 Mux for signal selection
- Up-to-date DAQ technology

Software Implementation

Graphics

imguiHandler.py

- Library: imgui-bundle [https://github.com/pthom/imgui_bundle]
- Inspiration: Visual Studio IDE
- Features:
    - Frame-locking (based on monitor refresh rate)
    - Continuous data acquisition
    - Fully integrated logging system
    - Thread-safe structures



Backend Script(s)

Src > Sensors

- Each module is implemented with a functioning backend script written in python
    - The DAQ is connects and records data via the [dwfpy] python library
    - Each sensor data analysis method is created for both TA implementation and guide
        - Skeleton code for students to implement
    - Data is plotted in real-time via [MatPlotLib]

```python
# —— Filter Functions ————————————————————————————————————
def butter_bandpass(lowcut, highcut, fs, order=2):
    nyq = 0.5 * fs
    return butter(order, [lowcut / nyq, highcut / nyq], btype='band')

def apply_bandpass_filter(data, lowcut=0.5, highcut=40.0, fs=10000, order=2):
    b, a = butter_bandpass(lowcut, highcut, fs, order)
    return filtfilt(b, a, data)

# —— Peak Detection ——————————————————————————————————————
def detect_ecg_peaks(signal, sample_rate, min_prominence=0.1, min_height=0.05):
    min_distance = int(sample_rate * 0.4)  # ~150 BPM max
    peaks, _ = find_peaks(signal, distance=min_distance, prominence=min_prominence, height=min_height)
    return peaks

def calculate_bpm(peak_times, current_time, window_duration=10):
    peak_times = [t for t in peak_times if t > current_time - window_duration]
    if len(peak_times) > 1:
        rr_intervals = np.diff(peak_times)
        avg_rr = np.mean(rr_intervals)
        bpm = 60 / avg_rr if avg_rr > 0 else 0
        return bpm, peak_times
    return 0, peak_times

# —— Plot Setup ——————————————————————————————————————————
fig, (ax1, ax3) = plt.subplots(2, 1, figsize=(10, 8))
fig.suptitle('Heart Rate Monitor')
```

Software Challenges

- I^2C
  - Kept getting a device not found error
  - Pulse-Oximeter has a 7-bit base address
  - 8 Bit address needed for added read and write bits
  - Had to shift the bit to make it 8 bits
- Device Conflict
  - Device already in use
  - Had to manage shared devices
  - Had to make sure all connections are ended

Lab Manual

Instruction

- Featuring
  - Installation, set-up and runtime instructions
  - Pre Lab
  - Implementation Guide
  - Post Lab

ECG Lab: Heart Rate Monitoring with DWF Python

Overview

This laboratory exercise guides you through developing a real-time heart rate monitoring system using the Digital Waveform Framework (DWF) Python library and a data acquisition device. By the end of this lab, you will:

1. Understand the principles of ECG signal acquisition
2. Process raw ECG signals to extract heart rate information
3. Implement peak detection algorithms for identifying cardiac events
4. Calculate heart rate (BPM) from ECG data
5. Create real-time visualizations of physiological data

Prerequisites

- Python 3.7+
- NumPy
- SciPy
- Matplotlib
- DWFpy library (`pip install dwfpy`)
- Data acquisition hardware (Digilent Analog Discovery or similar)
- ECG sensors/electrodes

Hardware Setup

1. **Connect the ECG sensor to your data acquisition device:**
   - Signal wire to oscilloscope input 1+ (positive)
   - Ground wire to input 1- (negative)
   - If using three-lead ECG, follow the sensor's documentation for proper electrode placement
2. **Verify connections:**
   - Ensure the data acquisition device is connected to your computer via USB
   - Check that the ECG sensor is properly connected to the subject (if applicable)

Code Structure

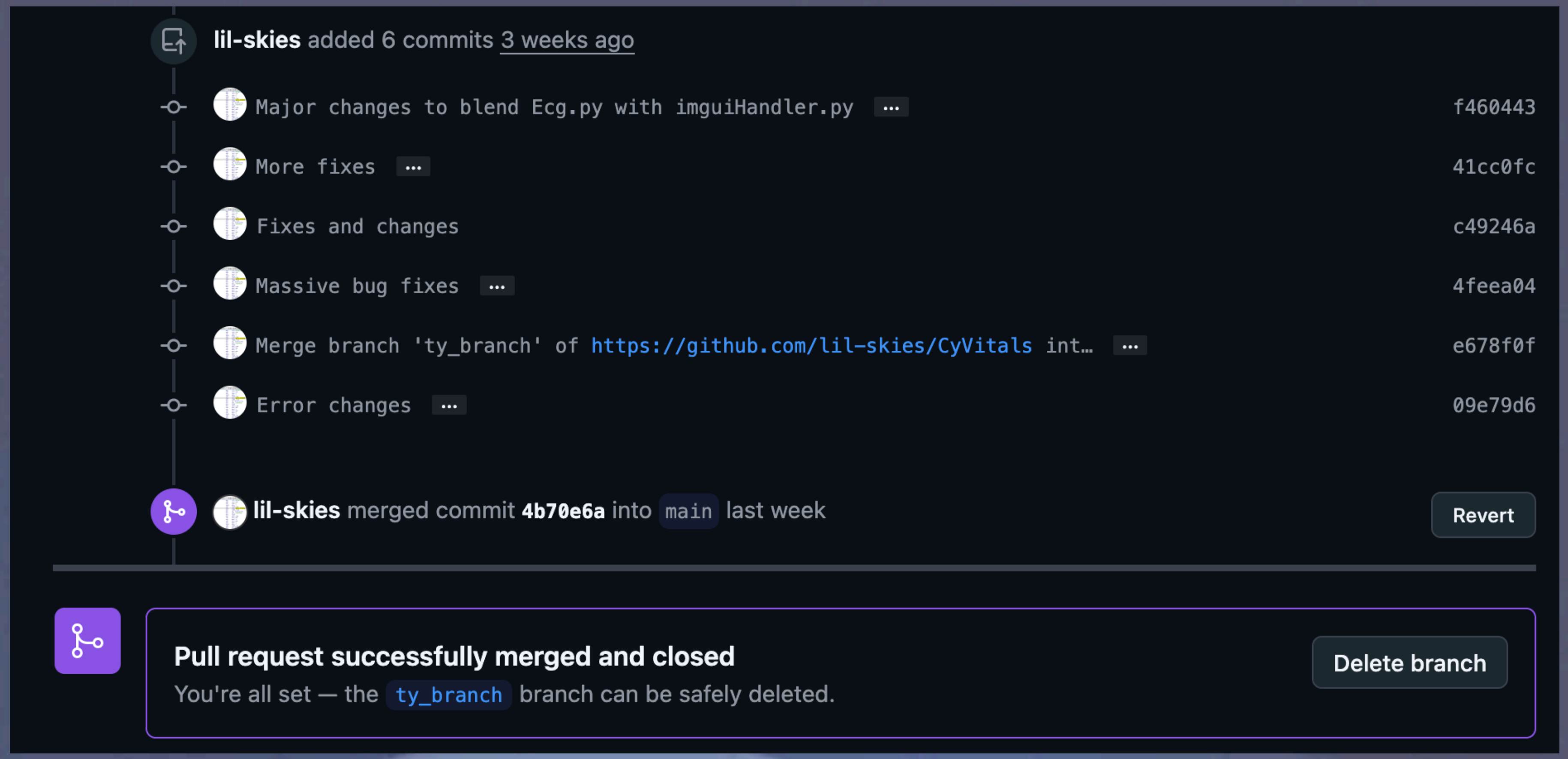
The provided skeleton code includes:

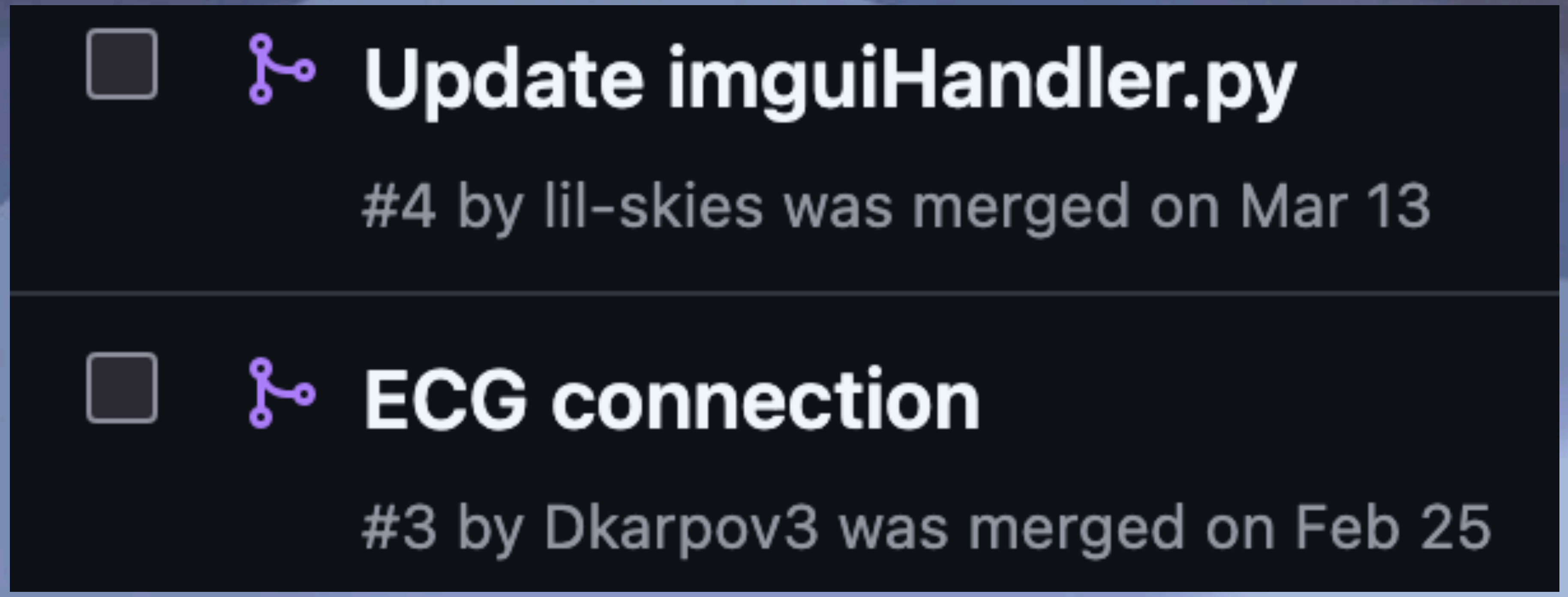
Testing

- Formatted exception handler with extensive logging
- Integration Testing
- Stress testing
- Live testing with comparative readings from apple watch

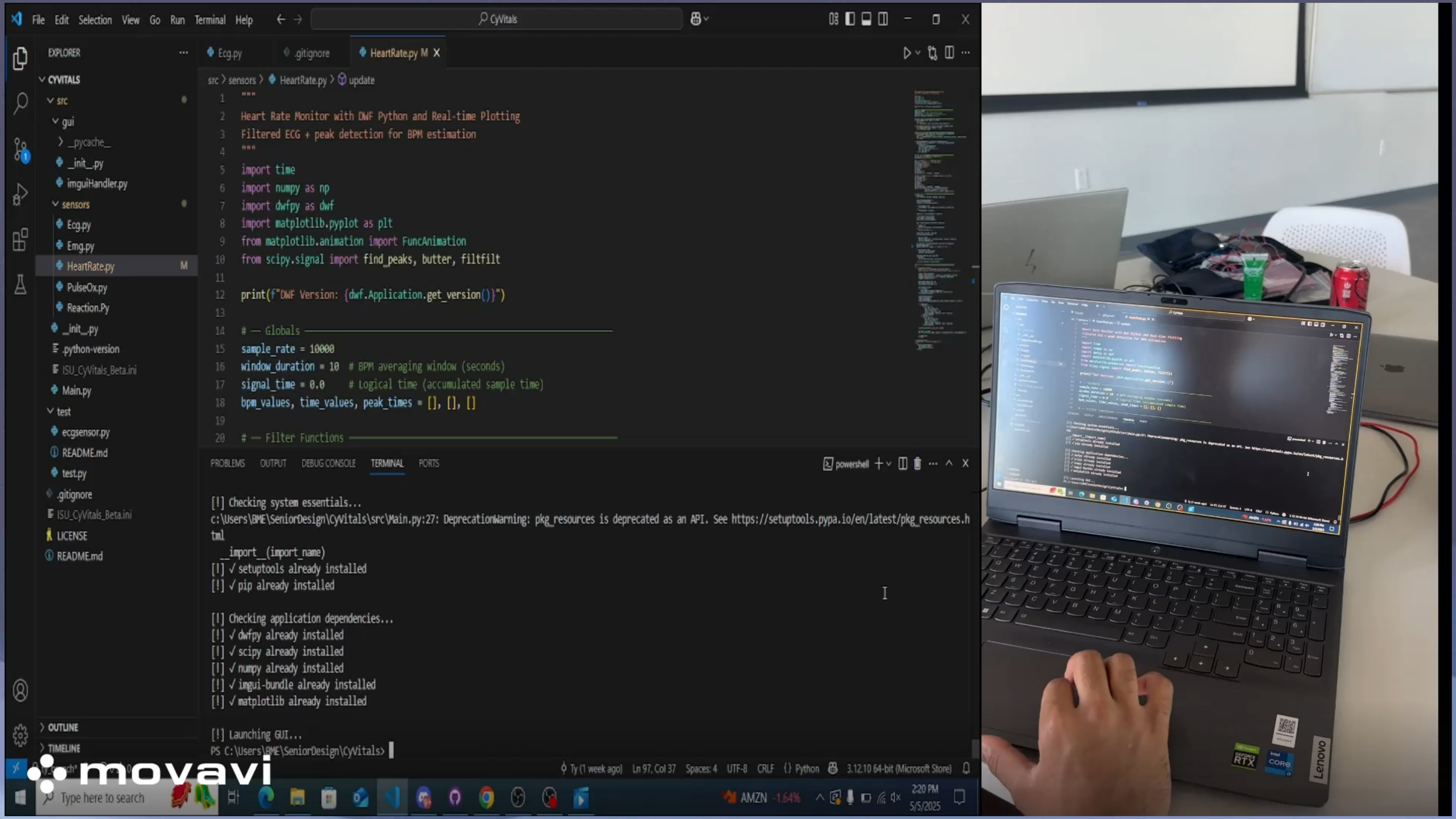
```python
def add_to_logs(message: str):
    with log_lock:
        timestamp = datetime.now().strftime('%H:%M:%S')
        log_messages.append(f"[{timestamp}]: {message}")
        if len(log_messages) > 1000:
            log_messages.pop(0)

def custom_log_gui():
    imgui.begin_child("Logs", imgui.ImVec2(0, 0), imgui.ChildFlags_.borders)
    imgui.push_style_color(imgui.Col_.text, imgui.get_color_u32((1.0, 1.0, 1.0, 1.0)))
    with log_lock:
        for msg in log_messages:
            imgui.text_unformatted(msg)
    imgui.pop_style_color()
    imgui.end_child()
```





Demo



Heart Rate Monitor with DWF Python and Real-time Plotting
Filtered ECG + peak detection for BPM estimation
import time
import numpy as np
import dwfpy as dwf
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.signal import find_peaks, butter, filtfilt
print(f"DWF Version: {dwf.Application.get_version()}")
# — Globals
sample_rate = 10000
window_duration = 10 # BPM averaging window (seconds)
signal_time = 0.0 # Logical time (accumulated sample time)
bpm_values, time_values, peak_times = [], [], []
— Filter Functions
[!] Checking system essentials...
[!] √ setuptools already installed
[!] √ pip already installed
[!] Checking application dependencies...
[!] √ dwfpy already installed
[!] √ scipy already installed
[!] √ numpy already installed
[!] √ imgui-bundle already installed
[!] √ matplotlib already installed
[!] Launching GUI...
movavi

Conclusion

Learning

- Interpreting waveforms
  - Filtering / smoothing, collecting analog signals
- Data structures
  - Continuous data acquisition
  - Python GUI development
- I2C

Conclusion

Future Ideas

- Because CyVital is built modularly, many new sensors both digital and analog could be added
- Additional cost savings could be implemented by testing alternative data acquisition devices
- Given sufficient data, ML algorithms could be developed to analyze recorded values from sensors.

Thank you for a great year!

Questions?