BioSteady Caffeine Intake Detection Device

ECE 445 Senior Design Laboratory

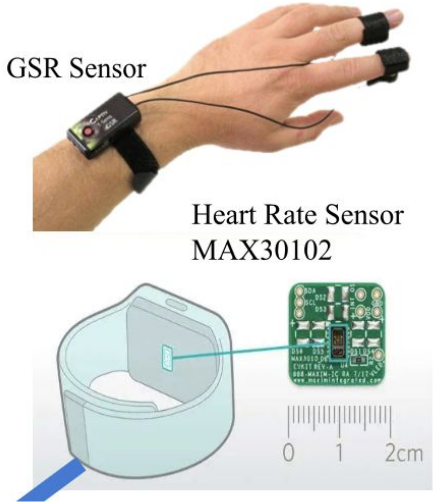
Team #46:
Alisha Chakraborty
Asmita Pramanik
Pranav Nagarajan

May 2, 2025

Problem Statement

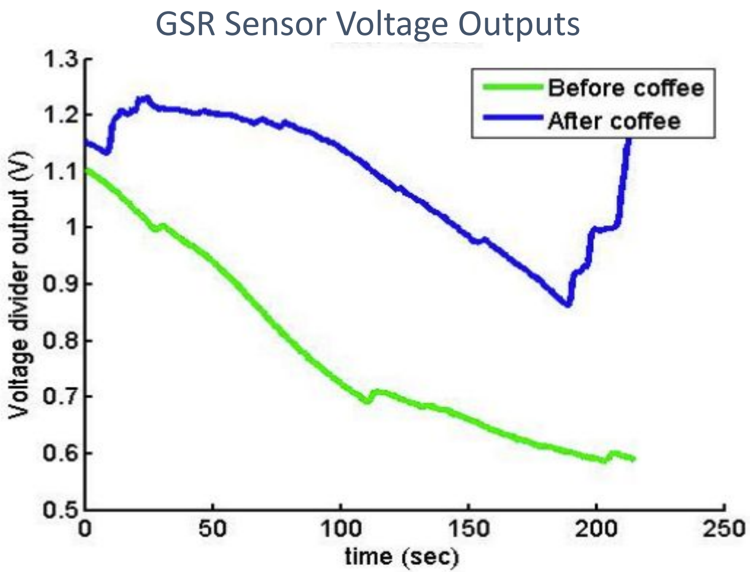
Motivation

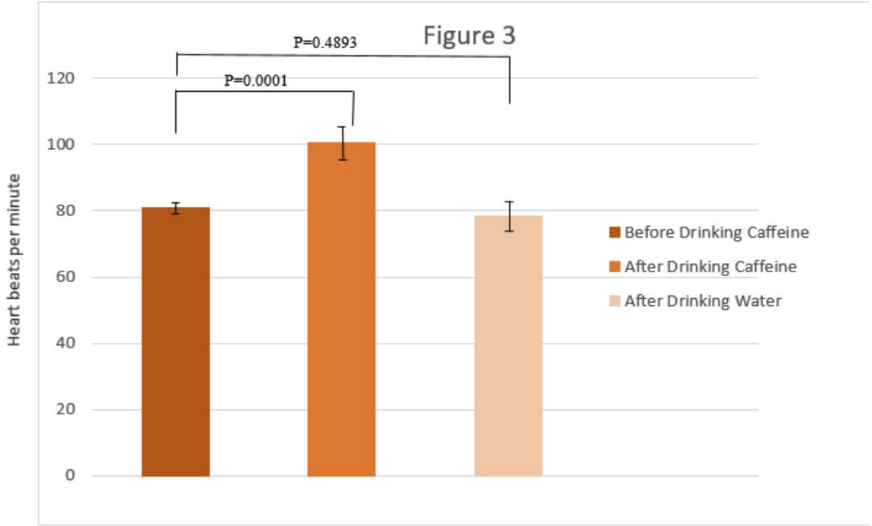
- Caffeine is widely used to maintain productivity in fast-paced environments.
- Its effects often go unnoticed, especially during high-stress periods.
- Both **stress and caffeine** increase heart rate and skin conductance.
- This overlap makes it hard to tell whether the body is reacting to stress or caffeine.
- Lack of real-time awareness leads to **uninformed decisions** that affect well-being



Problem Statement

Research





References : Villarejo, M. V., Zapirain, B. G., & Zorrilla, A. M. (2012). A stress sensor based on Galvanic Skin Response (GSR) controlled by ZigBee. *Sensors*, 12(5), 6075–6101.
Hovland, K. (n.d.). *The effects of caffeine on heart rate and blood pressure in college students*. Bethel University.
Guan, A., Hill, D., & Liang, L. (2022). *Can wearable sensors differentiate between caffeine and stress?* Journal of Emerging Investigators.
https://emerginginvestigators.org/articles/22-001/pdf

Our Solution : BioSteady

What It Is: A wearable device that monitors physiological signals to detect whether bodily changes are caused by **caffeine.**

How It Works:

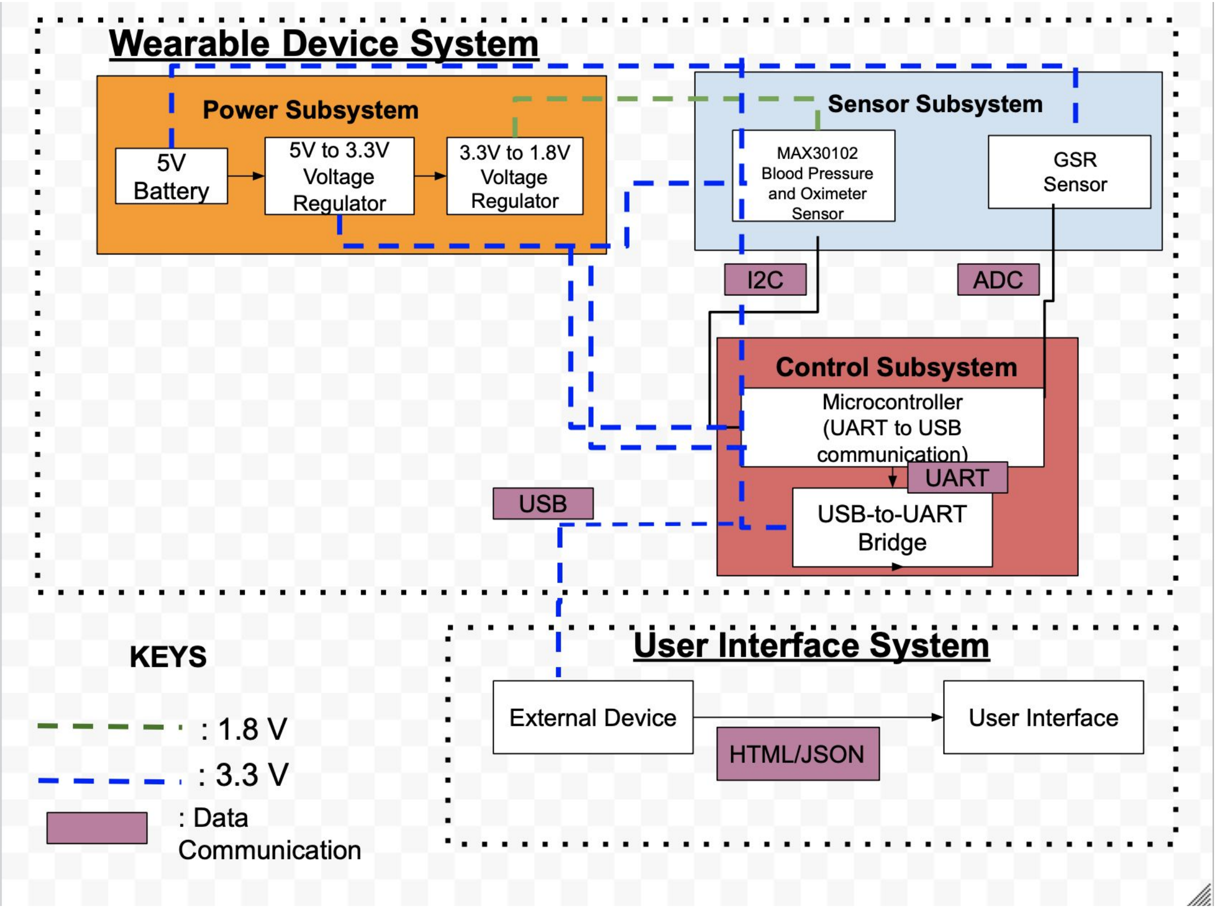
- Collects data via **MAX30102** (Heart Rate) and **GSR sensor**
- Processes signals using **STM32 microcontroller**
- Transmits data via **UART-to-USB** to a **web application**
- Classifies physiological states and provides user-friendly feedback

High Level Requirements

High Level Requirements

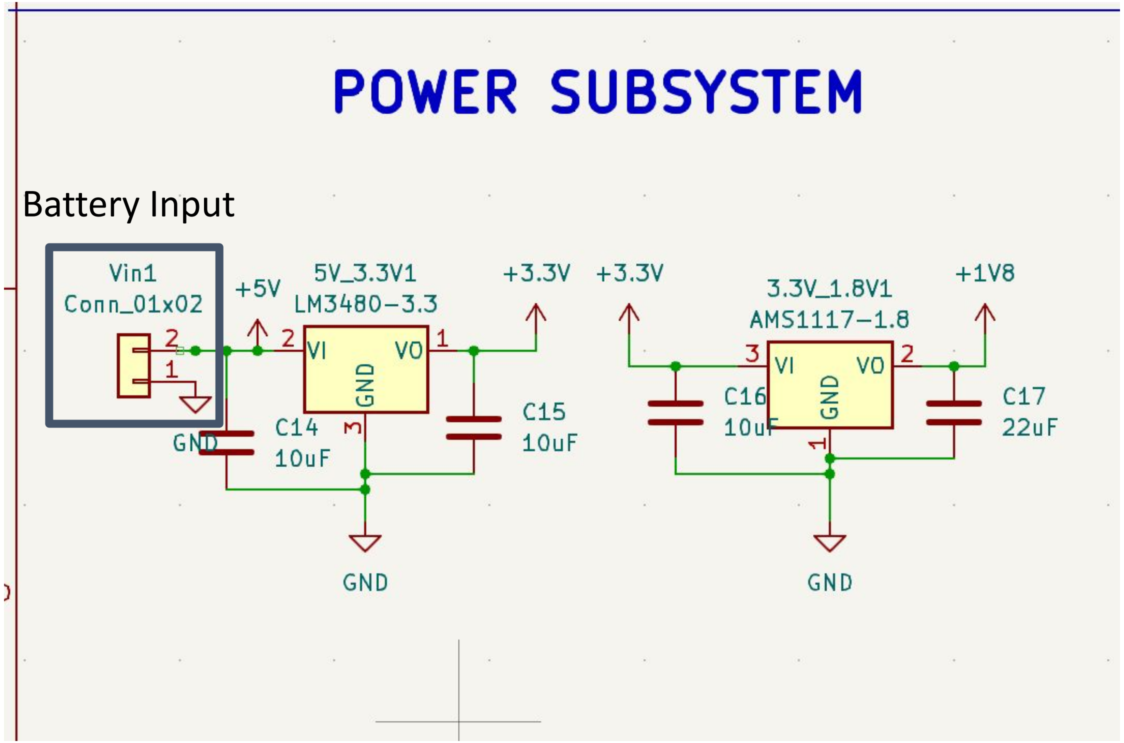
Data Collection & Processing	Data Transmission	User Interface
• The STM32L432KC microcontroller processes real-time physiological data from the MAX30102 heart rate sensor and Elecbee GSR sensor with minimal error. • It performs signal filtering to enhance data quality and improve the classification of stressed vs. caffeinated individuals. • Efficient processing is essential to ensure accurate real-time stress and caffeine detection.	• The microcontroller collects sensor data using appropriate protocols: I2C for MAX30102 and ADC for the GSR sensor. • Data must be transmitted with minimal latency to ensure real-time responsiveness. • Processed data is sent via UART and bridged over USB to an external device, such as a computer, for integration into a web application.	• The web application must display the user's physiological state analysis and distinguish between stress-induced and caffeine-induced responses. • It should provide actionable health recommendations based on the analysis. • The user interface (UI) must achieve at least 83% reliability through a data classification algorithm. • Information must be presented in a clear, user-friendly format for effective user engagement.

Block Diagram



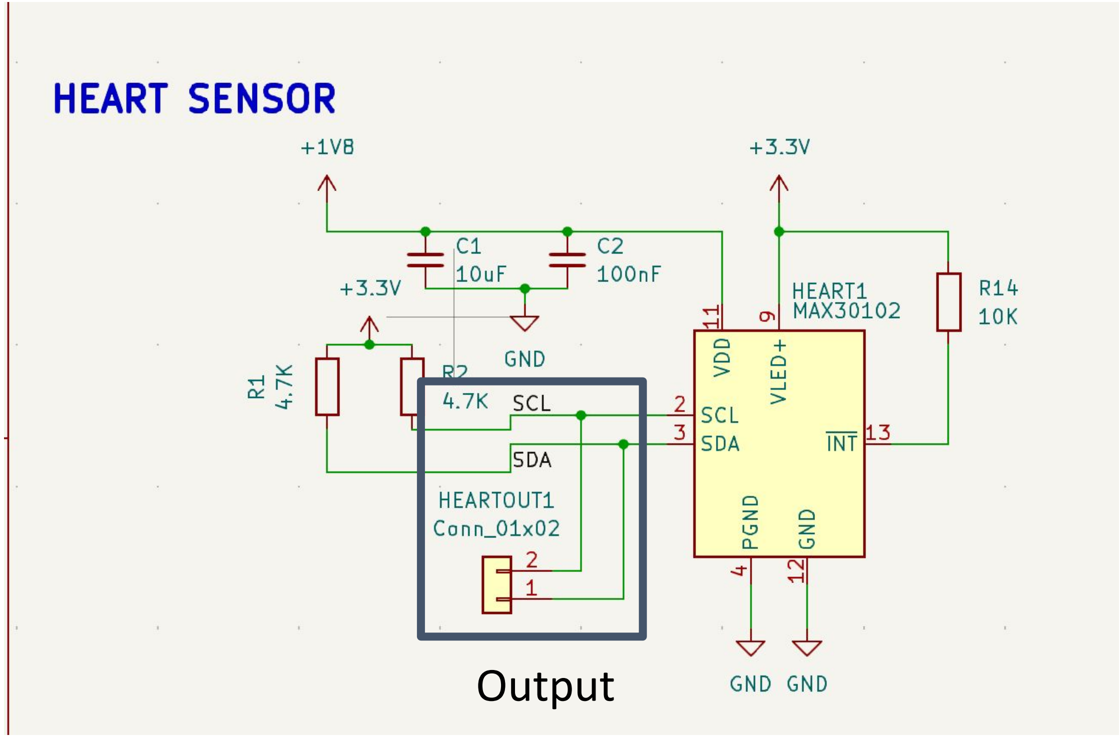
Technical Design

PCB Schematic and Layout : Power Subsystem



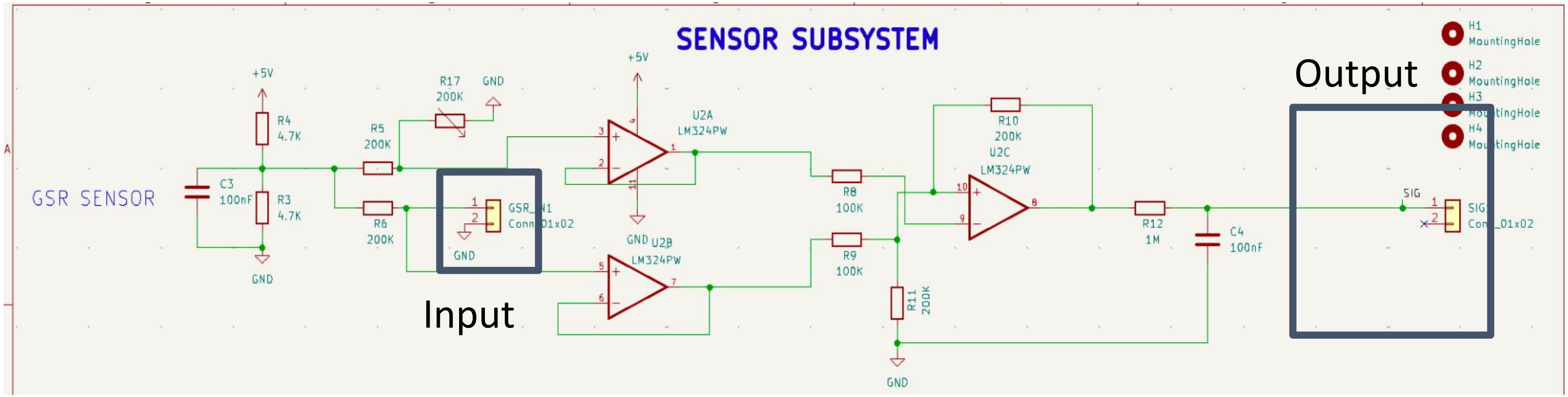
- **LM3480IM-3.3**:
  - Tiny 5V +- 5% to 3.3V, 100mA converter Quadi Low-Dropout Linear Voltage Regulator
- **AMS1117-1.8**
  - 3.3V +- 2% to 1.8V low dropout voltage regulator, 0.8 A

PCB Schematic and Layout : Sensor Subsystem



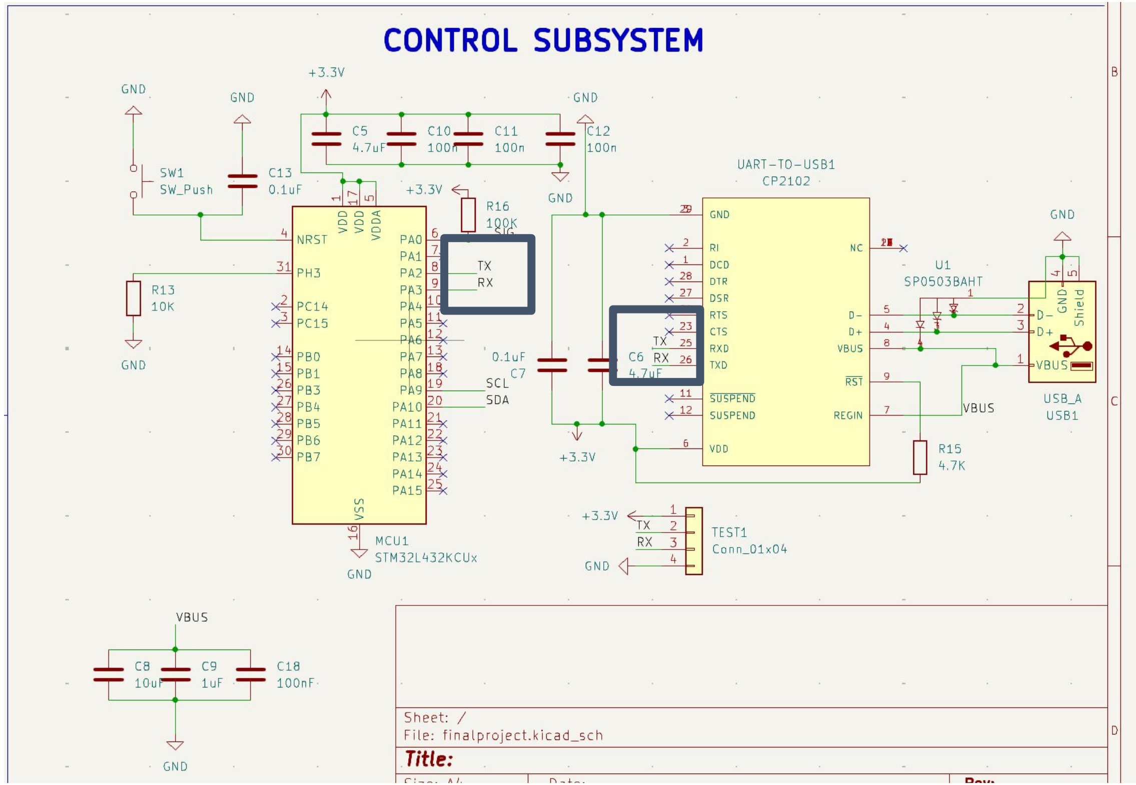
- **Oximeter and Heart Rate Sensor**
- MAX30102: High-Sensitivity Pulse Oximeter and Heart-Rate sensor
  - 1.8 V power supply and 3.3V LED power supply
  - Communication with STM32 MCU using standard I2C-compatible interface

PCB Schematic and Layout : Sensor Subsystem



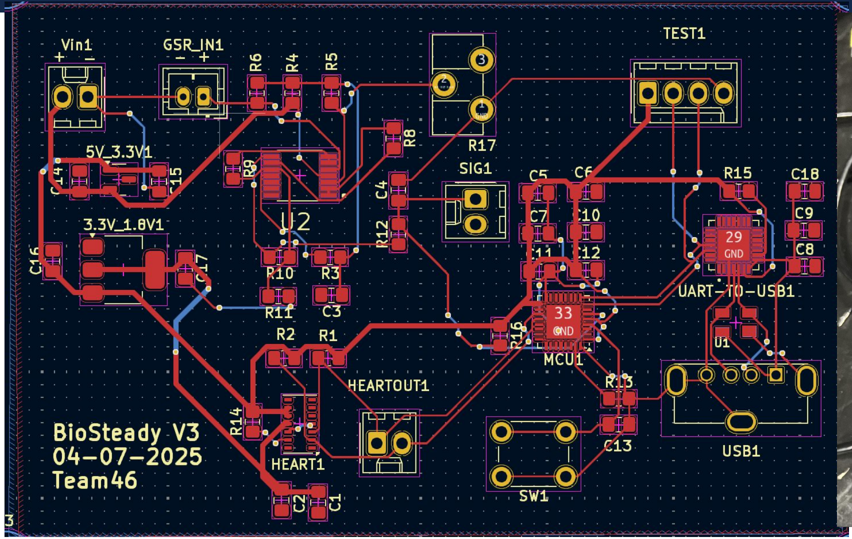
- **Galvanic Skin Response sensor:**
  - Operating voltage: 5V DC voltage
  - Detects skin resistance 10kOhm - 10MOhm using input electrodes
  - Output: Analog voltage signal of skin resistance, communication with STM32 HAL_ADC communication

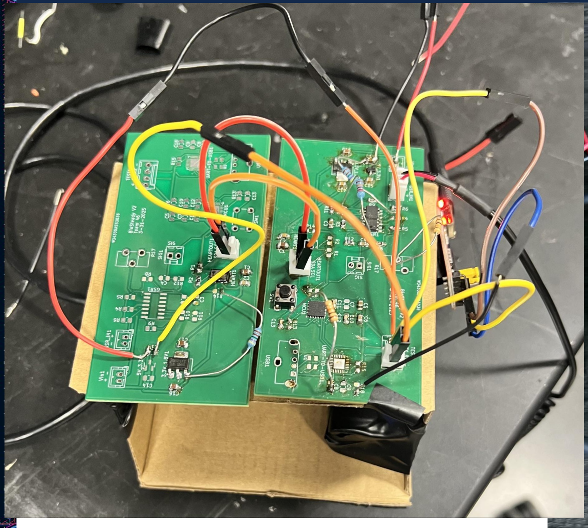
PCB Schematic and Layout : Control Subsystem

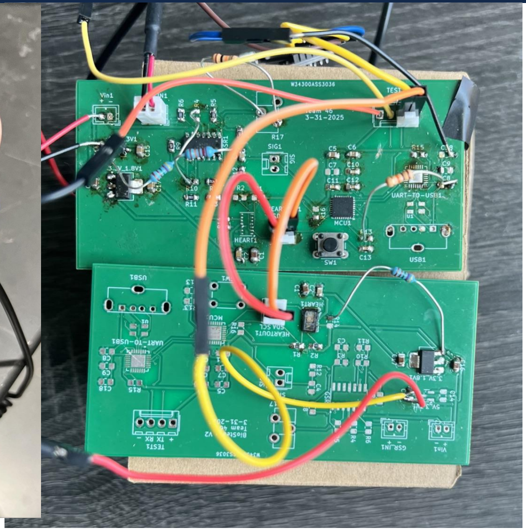


- **Microcontroller:** STM32L432KCU6 (ARM cortex-M4) to handle data collection and processing
- **Communication**: CP2102 Uart-to-USB bridge to enable serial data transfer to external device via USB-A connection

PCB Layout and Physical Board

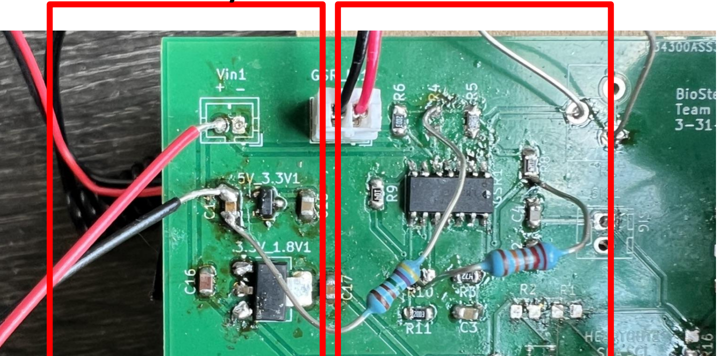




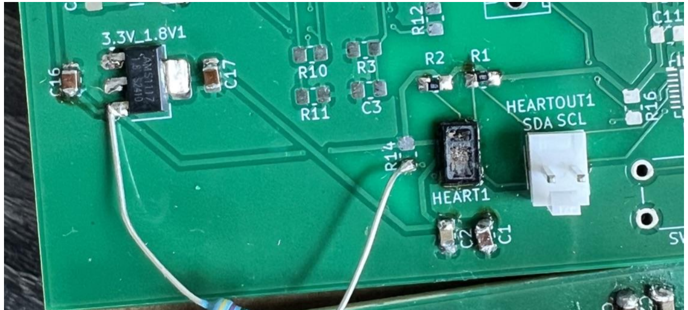


Power System

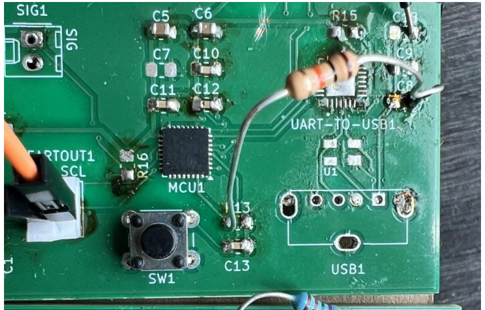
GSR sensor



MAX30102 Sensor



Control System



Microcontroller Unit Software

Unit for data acquisition and communication

Configuration of peripherals

- ADC for GSR Sensor (analog skin conductance)
- I2C for MAX30102 Heart Rate Sensor

Acquisition of Sensor Data

- Reads **IR light reflectance** from MAX30102
- Captures **analog voltage** from GSR and converts to nS

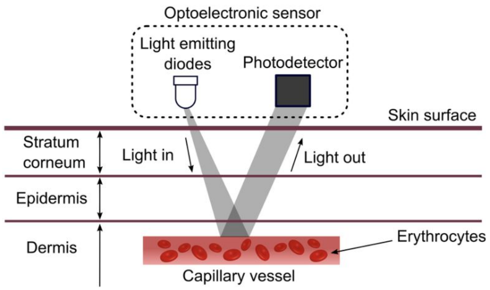
Formatting of Data

- Combines MAX30102 FIFO data bytes into 18-bit IR values

Transmission to Web Application

- Ensures data is sent for visualization every few seconds with HAL_Delay() throttling

```
while (1)
{

        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, 5000);
        sprintf((char*)buffer, "Value=%d\n",(int)HAL_ADC_GetValue(&hadc1));
        HAL_UART_Transmit(&huart2,buffer,strlen(buffer),HAL_MAX_DELAY);
        HAL_ADC_Stop(&hadc1);

        max30102_read(&max30102, 0x07, fifo_data, 3);

        // Combine to 18-bit value
        uint32_t ir_value = ((uint32_t)(fifo_data[0] & 0x03) << 16) |
                            ((uint32_t)fifo_data[1] << 8) |
                            fifo_data[2];

        // Transmit full data + each byte
        sprintf((char*)buffer, "IR= %lu\r\n",ir_value);

        HAL_UART_Transmit(&huart2, buffer, strlen((char*)buffer), HAL_MAX_DELAY);
        HAL_Delay(5000);
```



Sending MCU Output to Firebase

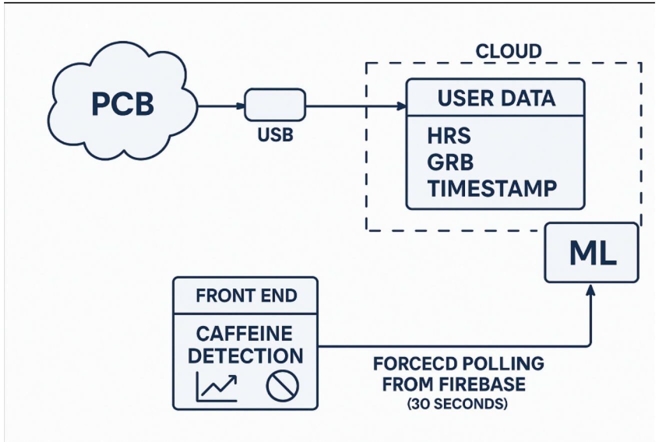
What the Script Does

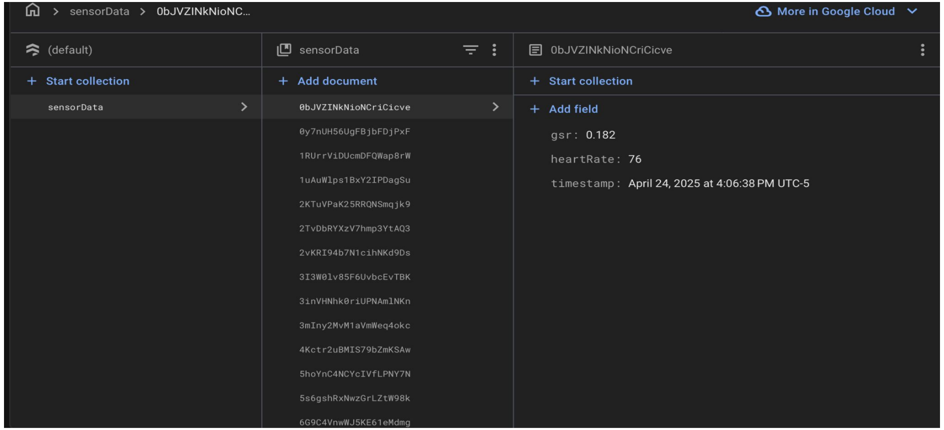
1. Connects to STM32 via Serial at 115200 baud
2. Parses and converts GSR
3. Parses IR values, buffers them with timestamps for peak detection
4. Estimates heart rate using **scipy.signal.find_peaks()** to detect heartbeats
5. Saves the structured data to Firebase for visualization

Typical Human Skin Resistance
-> 1KOhms - 1MOhms

```python
# 1) Parse GSR
if line.startswith("Value="):
    try:
        adc = int(line.split("=")[1])
        v_out = (adc / ADC_MAX) * V_IN
        if v_out:
            r_skin = R_FIXED * ((V_IN / v_out) - 1)
            pending_gsr = round((1 / r_skin) * 1e6, 3)
            print(f"→ Parsed GSR: {pending_gsr} µS")
    except ValueError:
        continue

# 2) Parse IR value from 3-byte line: IR=byte1,byte2,byte3
if line.startswith("IR="):
    try:
        ir_value = int(line.split("=")[1])
        ir_values.append(ir_value)
        timestamps.append(time.time())
        print(f"→ Parsed IR: {ir_value}")
    except ValueError:
        continue

# 3) Every 10 readings, calculate BPM and push to Firestore
if len(ir_values) >= 10:
    peaks, _ = find_peaks(ir_values, distance=1)
    if len(peaks) >= 2:
        peak_times = [timestamps[i] for i in peaks]
        intervals = np.diff(peak_times)
        bpm = int(60 / np.mean(intervals))
        print(f"Estimated BPM: {bpm}")

        if pending_gsr is not None:
            doc = {
                "gsr": pending_gsr,
                "heartRate": bpm,
                "timestamp": format_timestamp()
            }
            try:
                doc_ref.add(doc)
                print(f"Written to Firestore: {doc}")
            except Exception as e:
                print("Firestore write failed:", e)
```

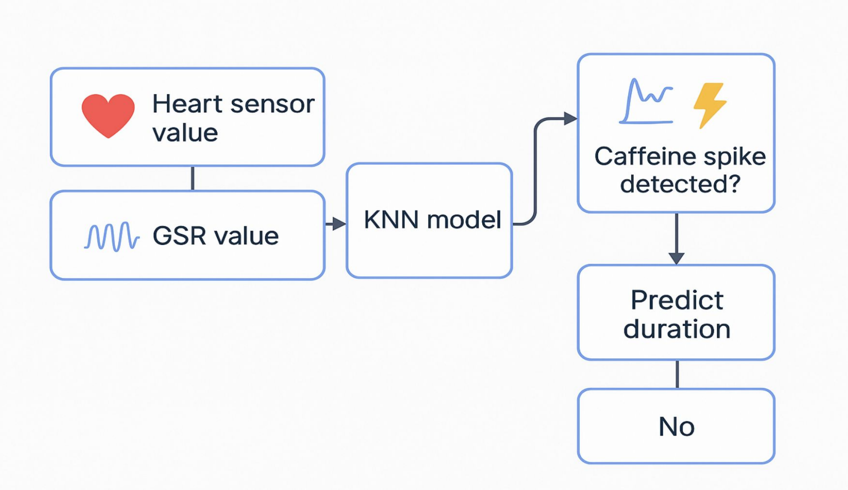
Backend Structure and Design Workflow

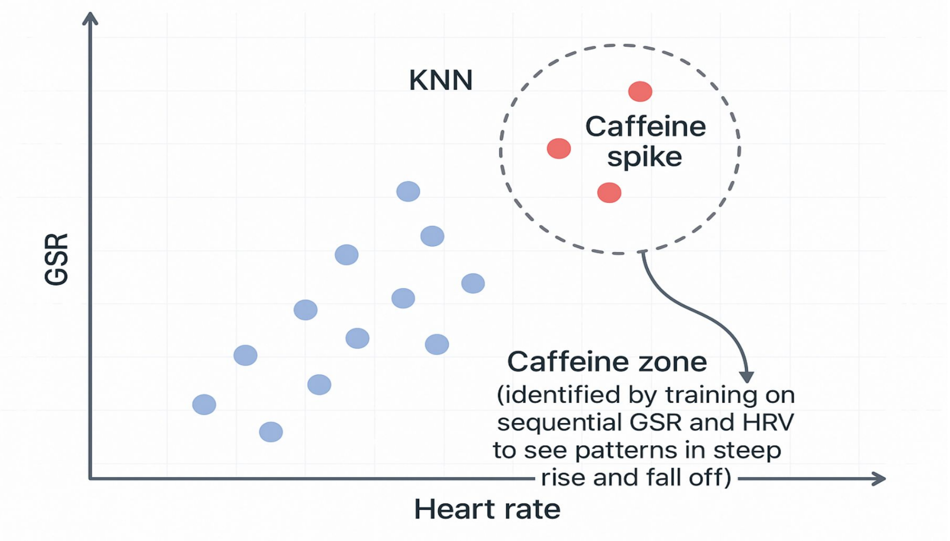


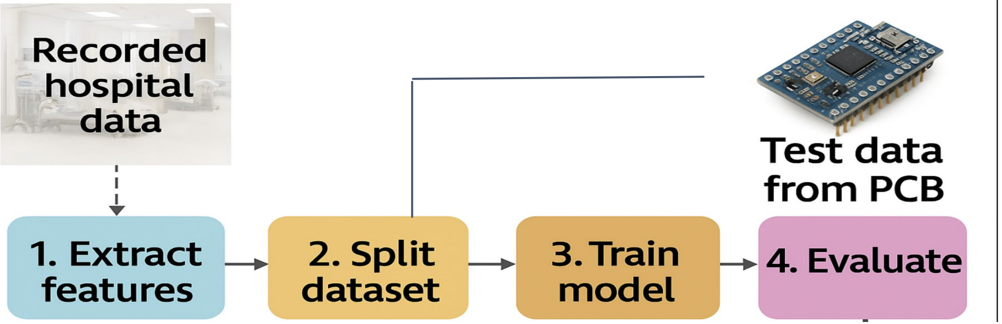



```
(venv) (base) pranav@MacBook-Air-93 caffeine-detector % python3 scripts/train_model.py

=== Accuracy ===
0.80

=== Classification Report ===
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        18
           1       0.00      0.00      0.00         2

    accuracy                           0.80        20
   macro avg       0.44      0.44      0.44        20
weighted avg       0.80      0.80      0.80        20

2025-04-13 23:15:45.427 Python[14205:6066104] +[IMKClient subclass]: chose IMKClient_Modern
2025-04-13 23:15:45.427 Python[14205:6066104] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

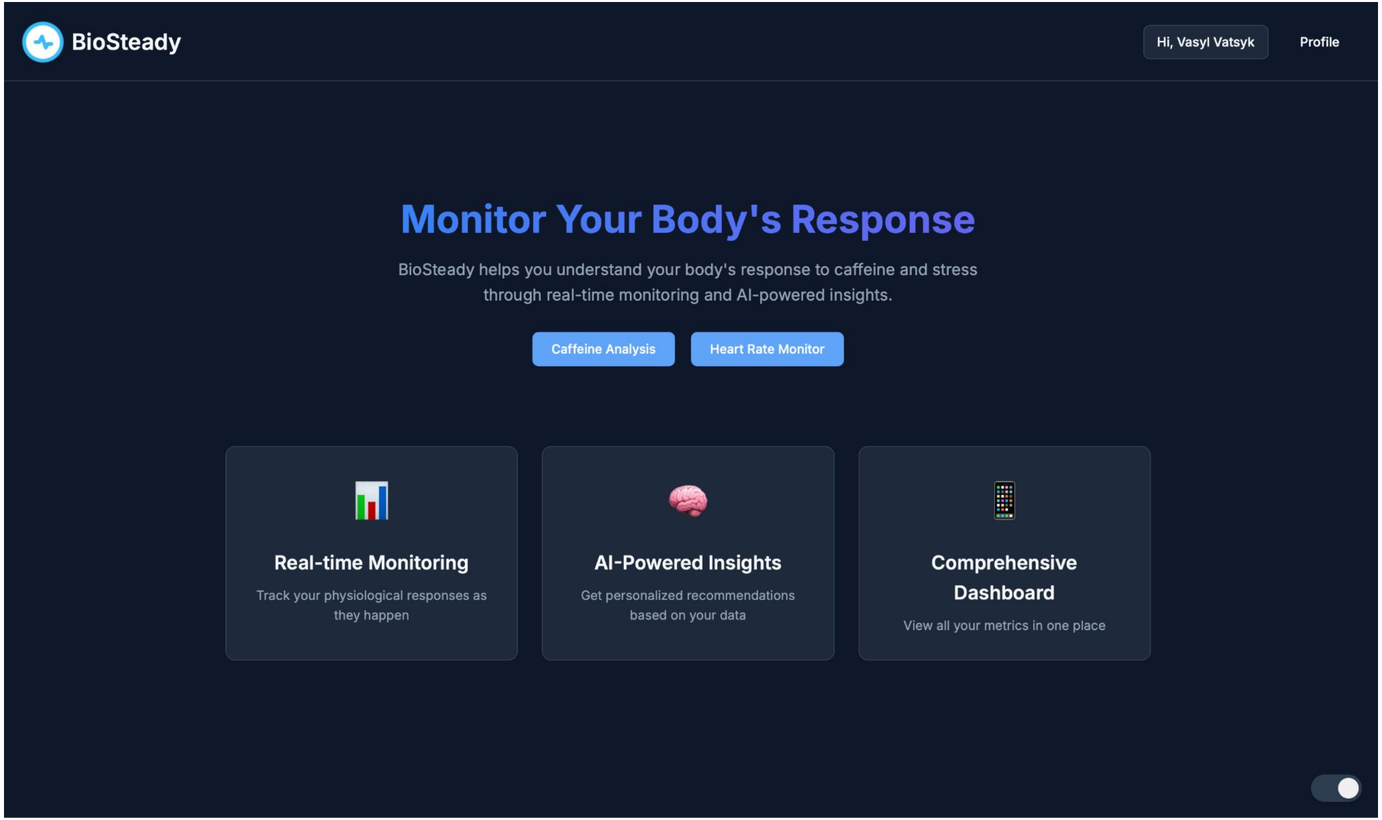
Model Explained



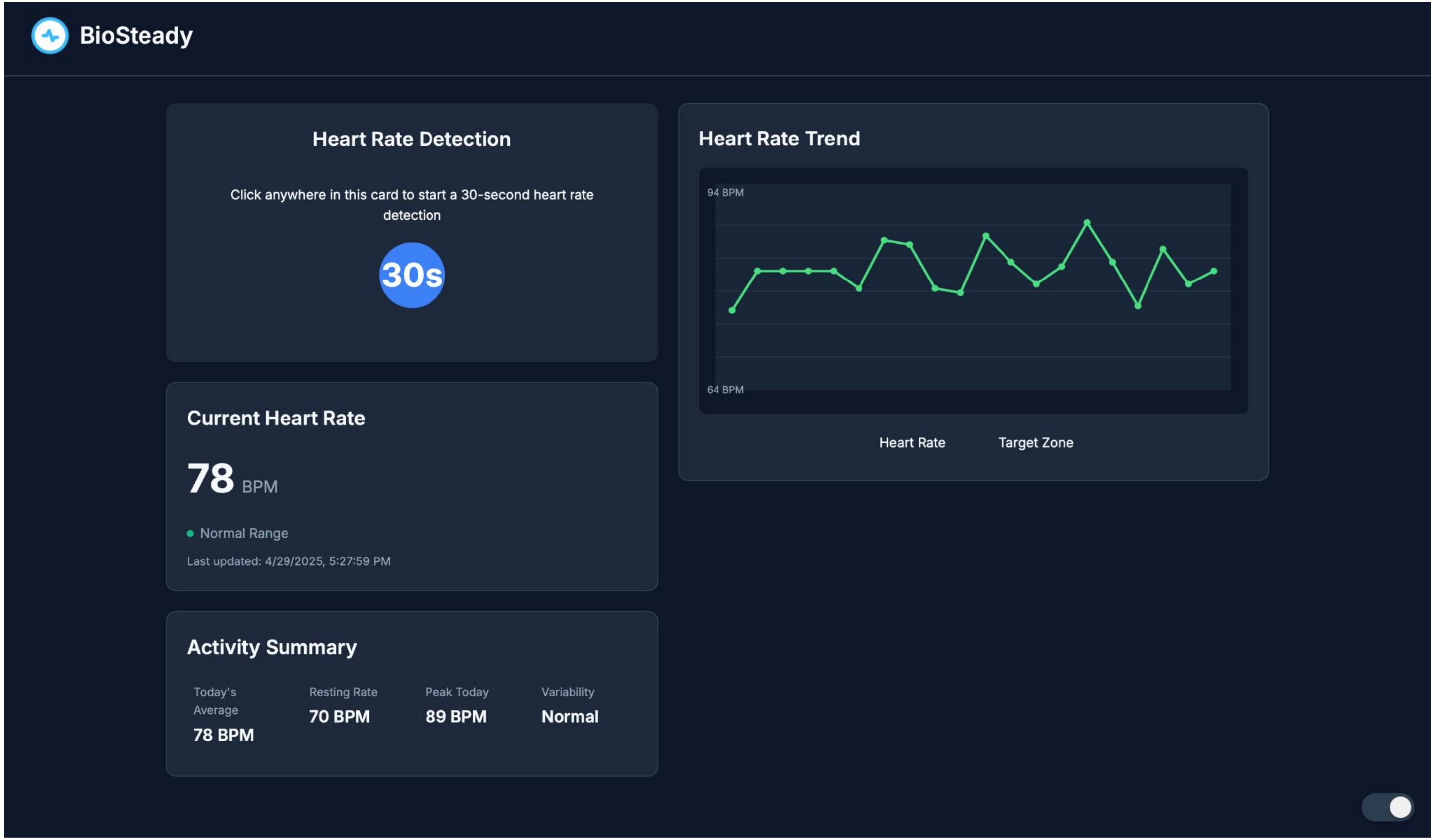




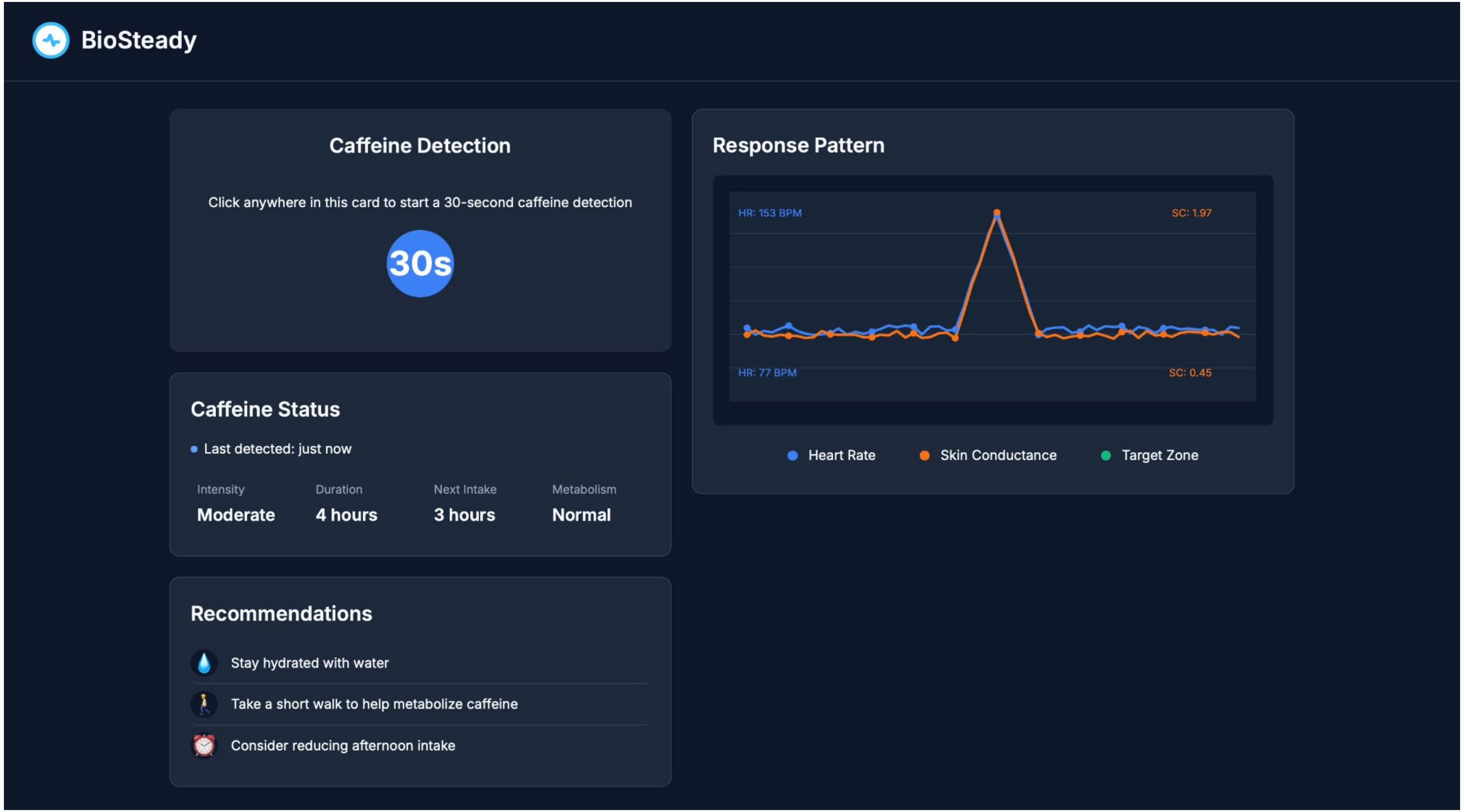
Front End - Landing Page

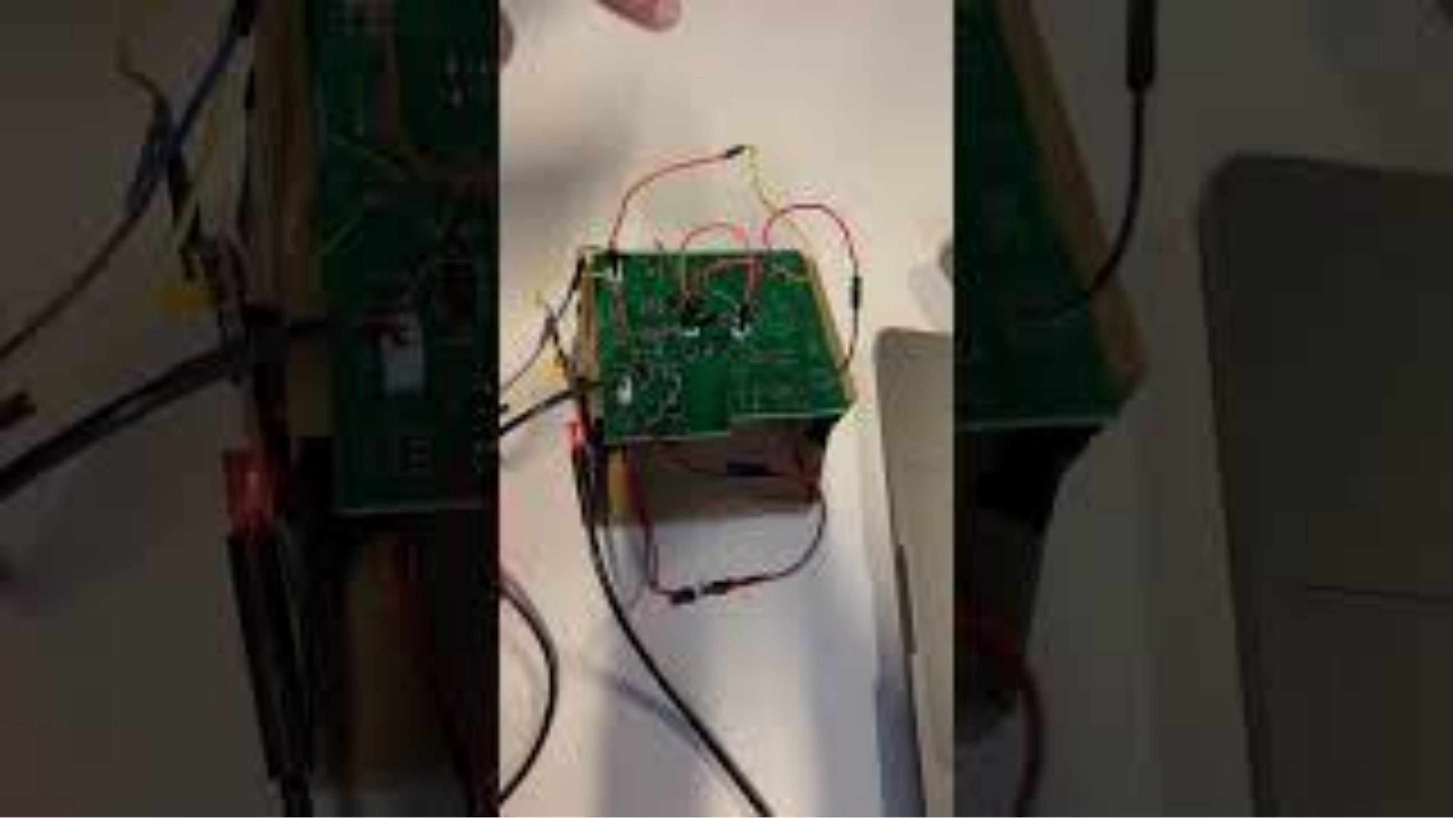


Front End - Live Heart Rate Sensor Page



Front End - Caffeine Detection Page





Requirements and Verification

Requirements and Verification

1. Biomedical Sensing

Requirement: HR sensor accuracy ±2 BPM; GSR output 0–3.3V
Verification: Compare HR sensor with ECG; simulate GSR changes with variable resistor
Success: 85–90% HR readings within range; voltage and response time meet spec

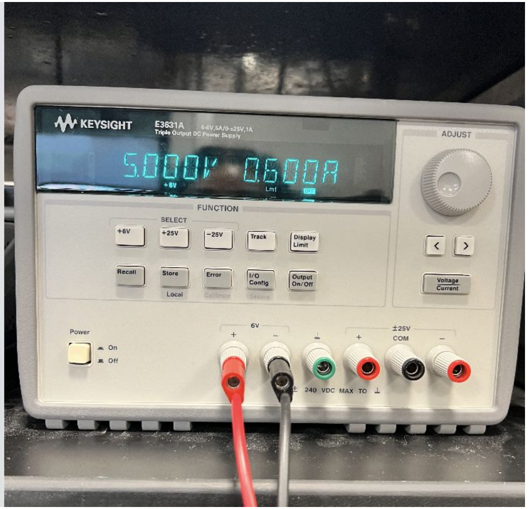
2. MCU & Power Management

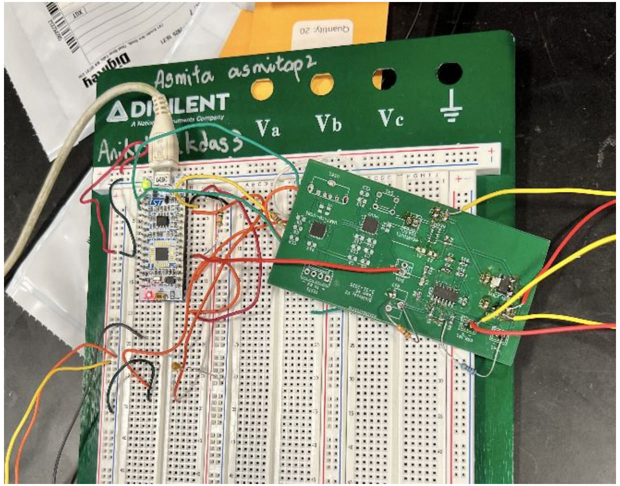
Requirement: Stable I2C & ADC data processing; 1.8V & 3.3V regulation within ±5%
Verification: Measure voltage stability and I2C error rate over 500+ transactions
Success: ≤2% I2C error rate; voltage tolerance within limits

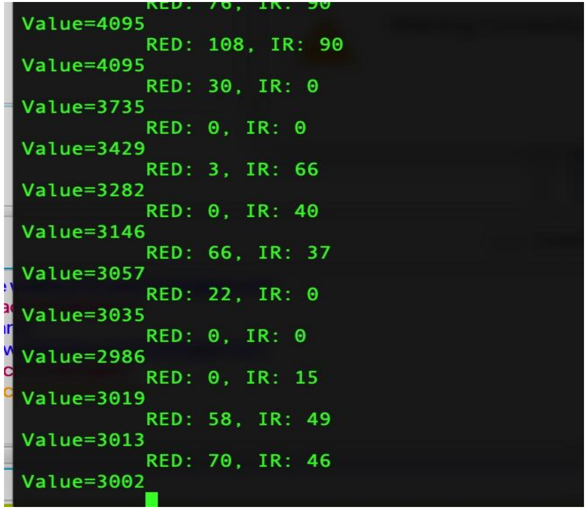
3. Web Application Integration

Requirement: Real-time data relay via UART-to-USB; <100ms UI display delay
Verification: Log transmission speed and UI render time
Success: 115200 bps transfer with <1% data loss; UI renders in <100m

Testing and Results








```c
void check_max30102_connection(void)
{
    uint8_t part_id = 0;
    char buffer[100];

    max30102_read(&max30102, 0xFF, &part_id, 1);  // 0xFF = PART_ID register

    if (part_id == 0x15)  // 0x15 is the expected ID for MAX30102
    {
        sprintf(buffer, "MAX30102 detected PART_ID=0x%X\r\n", part_id);
    }
    else
    {
        sprintf(buffer, "MAX30102 not detected Read PART_ID=0x%X\r\n", part_id);
    }

    HAL_UART_Transmit(&huart2, (uint8_t *)buffer, strlen(buffer), HAL_MAX_DELAY);
}
```

Testing sensor connection

Successes and Challenges

Successes & Challenges

Successes:

1. Full Integration of Sensors onto Hardware with data processing by microcontroller
2. Data transmission between device and front end design
3. Successful data uploads to cloud guaranteeing scalability

Challenges:

1. Broken LED Path within Heart Sensor
2. PCB Debugging
3. Writing MAX30102 driver code
4. Making the device easy to wear
5. I2C communication errors due to incorrect register configurations

Conclusion

BioSteady enables detection of caffeine responses using heart-rate and GSR sensors and a web-based interface.

Accurate data processing and a reliable UI promote healthier decision-making for users in high-stress environments.

The project prioritizes safety, transparency, and ease of use for everyday wellness monitoring

Scope of Improvements

Bluetooth Integration for wireless, on-the-go connectivity

Mobile App Development for more personalized insights

Enhanced ML Models for higher classification accuracy

Battery Optimization for extended wearability

Cloud Storage for long-term health tracking

Thank you!

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN