Sleep Apnea Monitoring System

Carson Copeland
ELET 3425 Embedded Systems
University of Houston
Houston, United States
cpcopela@cougarnet.uh.edu

Samuel Metzler
ELET 3425 Embedded Systems
University of Houston
Houston, United States
stmetzle@cougarnet.uh.edu

Hunter Rychlik
ELET 3425 Embedded Systems
University of Houston
Houston, United States
hjrychli@cougarnet.uh.edu

Enlai Yii
ELET 3425 Embedded Systems
University of Houston
Houston, United States
enleeyee@gmail.com

Abstract—**This paper presents the design and implementation of a real-time Sleep Apnea Monitoring System using embedded systems. The device reads pulse and SpO2 data via SPI from a MAX30102 sensor interfaced through an Arduino Nano, processes the data using a TM4C123GXL microcontroller, and visualizes results on an OLED while transmitting alerts over Bluetooth. The system detects heartbeats, calculates BPM, and identifies sleep apnea episodes if there is a heartbeat loss and SpO2 remains below 90% for more than 10 seconds.**

Keywords—sleep apnea, TM4C123GXL, MAX30102, SPI communication, embedded systems, OLED, Bluetooth, real-time health monitoring

I. Introduction

Sleep apnea is a prevalent and serious sleep disorder characterized by repetitive interruptions in breathing during sleep. According to recent studies, approximately 30 million individuals in the United States alone are affected by sleep apnea, with nearly 80% remaining undiagnosed [1]. If left untreated, sleep apnea can lead to severe long-term health complications including cardiovascular disease, hypertension, stroke, and chronic fatigue.

While significant advancements have been made in the field of sleep medicine, the diagnostic process for sleep apnea remains expensive, time-consuming, and often inaccessible to patients. Traditional diagnostic methods, such as polysomnography, require overnight clinical monitoring, posing logistical and financial challenges for many individuals.

Recent technological progress in embedded systems and healthcare monitoring has created opportunities for the development of affordable, user-friendly, and portable solutions for sleep apnea detection. However, the sleep health sector has lagged behind other areas of medicine in adopting modern, accessible technologies. Existing commercial devices are often either cost-prohibitive or provide limited functionality.

This project introduces a low-cost, portable sleep apnea monitoring system designed to detect and monitor apnea episodes in real time. The device continuously tracks breathing patterns and heart rate during sleep, leveraging infrared (IR) reflectance technology for heartbeat detection. The system can detect pauses in breathing, monitor oxygen saturation, and trigger alerts when severe apnea events are detected. Future expansions may include offline data storage and post-processing of recorded events when network connectivity becomes available.

By providing an accessible, at-home solution, the system aims to benefit individuals who suspect they may have sleep apnea but lack access to specialized healthcare facilities or cannot afford traditional sleep studies. The device enhances early detection capabilities, empowering patients to take proactive control of their sleep health.

II. Project idea/methodology justification

The selection of components for the Sleep Apnea Monitoring System was carefully considered based on a combination of hardware capabilities, ease of integration, power consumption, and real-time performance requirements.

A. TM4C123GXL Microcontroller (Tiva C LaunchPad)

The TM4C123GXL was chosen as the main controller platform due to several reasons. First, the LaunchPad was a requirement for the course lab environment, ensuring compatibility with provided development tools and educational objectives. Beyond course requirements, the TM4C123GXL features a Cortex-M4F core operating at 80 MHz, with a hardware floating-point unit (FPU) that significantly accelerates mathematical computations required in heart rate signals processing. Additionally, its low power consumption and deterministic interrupt handling via the Nested Vectored Interrupt Controller (NVIC) are advantageous for real-time physiological monitoring where time precision is critical.

The Tiva C LaunchPad does require low-level c programming expertise from structures, functions, header files, and procedural programming paradigm. It lacks built-in wireless communication requiring an external Bluetooth module (i.e. HC-05) with the initialization and processing code. With the idea of self-initialization code, the board has a more complex peripheral configuration compared to other brands as Arduino or Raspberry Pi.

Identify applicable funding agency here. If none, delete this text box.

B. *MAX30103 Pulse Oximeter Sensor*

The MAX30102 was selected as the biosensing element for monitoring IR reflectance related to pulse and SpO2 levels. The sensor provides an all-in-one solution with integrated red/IR LEDs, photodetector, and analog/digital processing. Moreover, it benefits from open-source Arduino driver libraries, simplifying early testing and ensuring ease of interfacing via I2C. The high sensitive for heart rate and SpO2 detection proved advantageous with its compact, low-power design and extensive public documentation and community support.

The MAX30102 requires careful placement and pressure control for accurate readings. With this restriction, noise and motion artifacts will affect measurements, making the sensor not medical-graded for any true diagnosis. The sensor required careful calibration for SPO2 calculation due to the complexity of not being professional-graded. And, from a programming point, the MAX30103 has no community support or drivers for the Tiva C LaunchPad, more complex programming required.

C. *Monochrome 1.3" OLED Graphic Display*

For visual feedback between the system and the user, a 1.3" OLED using the I2C bus was selected. OLED displays are highly readable under various lighting conditions from high contrast and wide viewing angles, plus consumes less power compared to LCDs when displaying fewer pixels. The display is limited in size, restricting complex graphics with slower update rates compared to parallel-interface displays. The OLED requires display refresh logic management (clearing, updating regions).

D. *HC-05 Bluetooth Module*

The HC-05 module was chosen to provide wireless data communication to a mobile phone, tablet, or computer. It uses simple UART serial communication, enabling easy integration with the TM4C123GXL. The module is highly affordable and widely available with simple AT-command interface for configuration.

The drawbacks include an unstable paring with Window and Mac devices. Which was a downside during the live demonstration. The module is limited to Bluetooth Classic (not Bluetooth Low Energy), restricting iOS device compatibility. The lack of modern security features makes the module more vulnerable to hacking and injections compared to Bluetooth Low Energy modules.

E. *Arduino Nano*

The Arduino Nano was deployed as the SPI master device responsible for interfacing with the MAX30102 and forwarding the sensor data to the Tiva board. Using Arduino isolates the high-level management of the sensor from the critical real-time requirements of the TM4C123GXL. The Arduino simplified development via Arduino driver libraries and community support. The board offloads sensor-specific initialization and processing form the Tiva board with a flexible SPI transaction control.

By introducing a second microcontroller, there is more complexity to the whole system from power management, SPI communication, and separate codebases. There was a synchronization issue due to careful SPI protocol handling and the Arduino's clock speed (16 MHz) limits maximum SPI transfer rates. The data transfer was limited to 1 binary digit, instead of the original 4 digit binary code.

F. *Summary*

Overall, the selection of components was motivated by balancing real-time performance, ease of programming, power efficiency, and system modulatory. Each hardware module complements specific functional requirements, and together, they enable a scalable and robust embedded monitoring system.

III. PROJECT IDEA AND METHODOLOGY

The proposed system consists of a two-microcontroller architecture designed to facilitate reliable real-time detection and monitoring of sleep apnea events. The system architecture separates high-level sensor management from critical signal processing, ensuring that real-time constraints are met without overwhelming a single processing unit.

A. *Arduino Nano and MAX30102 Interface*

An Arduino Nano microcontroller interfaces directly with the MAX30102 pulse oximeter sensor using the I2C protocol. The MAX30102 sensor captures infrared (IR) reflectance signals from the user's fingertip, which correlate with blood oxygen saturation and heartbeat variability.

To enhance portability and flexibility, the Arduino Nano acts as an SPI master device, continuously collecting IR data samples and transmitting them in real time to the TM4C123GXL microcontroller via the Serial Peripheral Interface (SPI) bus. This modular separation allows the Arduino to manage sensor initialization, calibration routines, and preprocessing steps without burdening the main data processing unit.

B. *Arduino Nano and MAX30102 Interface*

The TM4C123GXL microcontroller serves as the SPI slave and the primary data processing unit. Upon receiving IR data from the Arduino Nano, the Tiva board performs the following tasks:

- Beat Detection: A slope-based peak detection algorithm is applied to identify the rising and falling edges characteristic of a heartbeat. Detection is triggered upon the transition from a rising to falling IR signal exceeding a predefined threshold.
- BPM Calculation: The interval between detected beats is recorded, and a rolling average of recent intervals is used to compute the beats per minute (BPM), constrained to physiologically realistic limits.
- SpO2 Monitoring: Although SpO_2 calibration is fixed in this prototype, future expansions will incorporate dynamic SpO_2 calculations using IR and red light absorption ratios.
- Apnea Detection: Apnea events are flagged if heartbeats cease for more than 10 seconds or if simulated SpO_2 values drop below a predefined threshold for a sustained period.

- Output Functions: Real-time results are displayed on an OLED screen over I2C, and critical alerts are transmitted over UART to an HC-05 Bluetooth module.

C. *Communication Methodology: SPI Polling*

Communication between the Arduino and the TM4C123GXL is implemented using SPI polling. In this approach, the Tiva board actively monitors the Receive FIFO Not Empty (RNE) flag within the SSI2 status register to detect incoming data.

Polling was selected over an interrupt-driven interface for several reasons:

- Simplicity: Polling avoids the overhead associated with interrupt handling, simplifying software implementation in a resource-constrained real-time environment.
- Data Integrity: Since data arrives at a known fixed sampling rate (~100 Hz), polling ensures timely data handling without risking missed samples due to interrupt latency.
- System Robustness: Reduces complexity in synchronization logic compared to interrupt-driven buffering mechanisms.

D. *Design Constraints and Considerations*

a) Timing Accuracy and Data Rate Requirements

Accurate heartbeat and apnea event detection necessitated strict timing guarantees. The system sampling rate was established at 100 Hz, as configured in the main application (#define SAMPLE_RATE_HZ 100). Thus, a new IR sample was transmitted from the Arduino Nano to the TM4C123GXL every 10 ms.

The SPI communication was configured at 250 kbps. With each frame consisting of a single byte, the transfer time per sample was approximately 32 us, resulting in an overall SPI communication load of less than 0.32% of available MCU processing time.

This low data overhead validated the decision of employ polling-based SPI communication rather than interrupt-driven handling, ensuring predictable system behavior without unnecessary complexity.

TABLE I. TIMING AND SPI COMMUNICATION CHARACTERISTICS

Parameter	*Value*
Sampling Rate	100 Hz
SPI Transfer Time per Byte	32 us
SPI Data Rate	100 bytes
CPI Load due to SPI	< 0.32%

b) Memory Usage Constraints

Memory usage on the TM4C123GXL was minimized through static allocation strategies. Key memory allocations included:

- irBuffer[100]: 100 bytes for IR sample smoothing.
- BeatDetector structure: approximately 16 bytes for beat interval tracking
- UART transmission buffers for Bluetooth JSON output.

Given the TM4C123GXL's 32 KB SRAM, the system's estimated RAM usage remained under 2%, leaving sufficient headroom for additional functionality or future expansions.

c) Power Efficieny Considerations

The combined system, including the TM4C123GXL microcontroller, Arduino Nano, OLED display, and HC-05 Bluetooth module, was estimated to consume approximately 60-80 mA during active operation. This was derived based on datasheet specifications:

TM4C123GXL: ~18.8 mA at 80 MHz

Arduino Nano: ~15 mA (idle)

MAX30102 sensor: ~0.6 mA

HC-05 Bluetooth: ~30-40mA (active transmission)

Such power levels make the device suitable for battery-powered applications. For example, a 1000 mAh battery could support continuous operation for approximately 12-16 hours.

TABLE II. TYPICAL SYSTEM POWER CONSUMPTION ESTIMATES

Component	*Typical Current Draw*
TM4C123GXL	100 Hz
Arduino Nano	32 us
MAX30102 Sensor	100 bytes
HC-05 Bluetooth	< 0.32%

d) Communication Bandwidth Constraints

The UART Bluetooth communication utilized a 9600 bps baud rate, equivalent to a theoretical maximum of 9960 bytes/sec. Given that a typical JSON transmission packet was approximately 24 bytes and sent once per second, UART bandwidth utilization remained well under 5% of available capacity, ensuring reliable wireless alert delivery without congestion.

TABLE III. COMMUNICATION LINK UTILLIZATION

Communication Link	*Data Rate*	*Utilization*
SPI	100 Hz	< 1%
UART Bluetooth	32 us	< 5%

e) OLED Display Refresh Timing

The OLED display updated text strings reflecting IR values, BPM, and SpO2 levels. Textual updates over I2C at a clock speed of 400 kHz resulted in negligible system load, as updates occurred approximately once per second upon new heartbeat detections.

f) Summary

The cumulative design choices, summarized in Table IV, ensured that timing deadlines were consistently met, memory consumption remained low, power usage was acceptable for portable operation, and communication links operated well within bandwidth limits.

TABLE IV. SUMMARY OF SYSTEM DESIGN CONSTRAINTS AND MITIGATIONS

Constraint	***Handling Strategy***
Real-Time Timing	100 Hz sampling, SPI polling
Memory Utillization	Static allocation (< 2% SRAM)
Power Consumpution	Low current (~60-80 mA total)
SPI Communication	Low bandwidth, minimal CPU load
UART Communcation	Low utilization (< 5% of link)
OLED Display Updates	Sparse updates over I2C

IV. PROJECT DESIGN, DEVELOPMENT, IMPLEMENTATION, ANALYSIS OF FINDINGS

A. *Hardware Configuration*

The Sleep Apnea Monitoring System integrates several hardware components optimized for low power and real-time performance:

- MAX30102 Sensor: Measures infrared (IR) reflectance values indicative of blood oxygen saturation (SpO2) and heart pulse rate. Connected via I2C to the Arduino Nano.
- OLED Display (128x64 pixels): Provides real-time visual feedback of IR levels, BPM, and SpO2. Interfaced over I2C with the TM4C123GXL microcontroller.
- Bluetooth HC-05 Module: Enables wireless transmission of heart rate and apnea alerts via UART communication to external device.
- Tiva SSI2 Module (TM4C123GXL): Configured in SPI slave mode to receive continuous IR data streams from the Arduino Nano.
- GPIO and UART Debugging: Dedicated UART outputs and GPIO status LEDs were used for debugging communication states and heartbeat detection in real time.

The overall system architecture and circuit diagram is shown in Fig 1 and Fig 2.

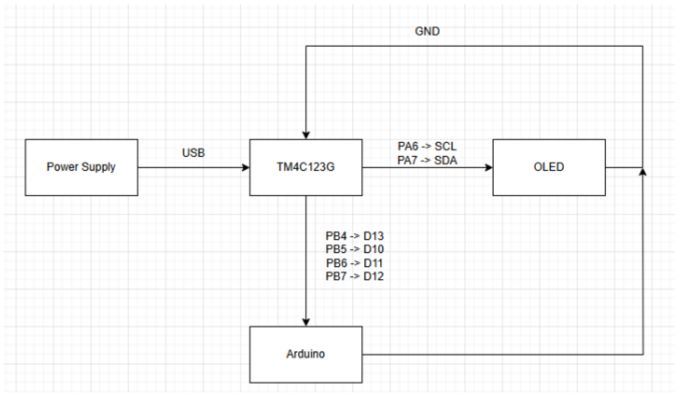


Fig. 1. System Hardware Architecture Diagram

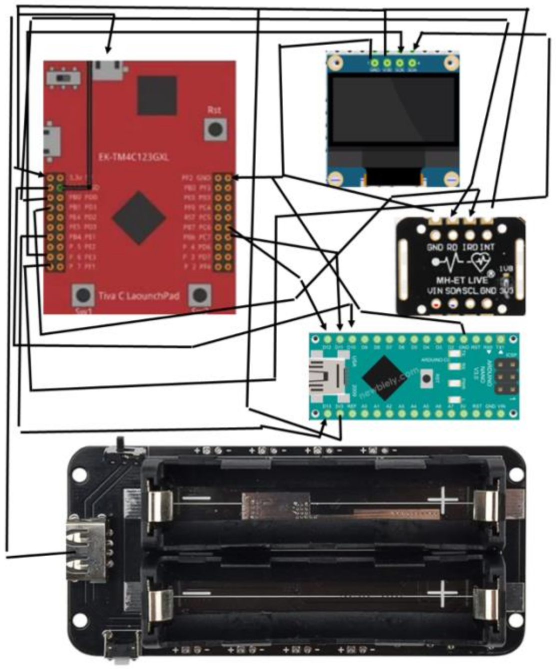


Fig. 2. Circuit Diagram

B. *Software Components*

The software stack was structured into modular components.

- SPI Driver: Configured SSI2 peripheral in slave mode; data was retrieved by polling the Receive FIFO Not Empty (RNE) flag. Each incoming SPI frame consisted of a state byte, one compressed IR data byte, and a checksum.
- Beat Detection Algorithm: Employed a simple slope-based peak detection algorithm. Rising-to-falling slope transitions, exceeding a minimum threshold, were interpreted as heartbeats.
- BPM Calculation Module: Computed the beats per minute (BPM) by averaging the intervals between the last hour detected beats. Outlier filtering was applied by constraining the BPM within predefined physiological limits (40-180 BPM).

- Apnea Detection Logic: Triggered an apnea event if either a) no heartbeat was detected for more than 10 seconds, or b) SpO2 dropped below 90% for longer than 10 seconds.
- Bluetooth Output Formatter: Serialized monitoring data and event notifications into lightweight JSON format strings, transmitted via UART to the HC-05 module.

A detailed flow of the software execution is shown in Fig. 2 and Fig. 3.

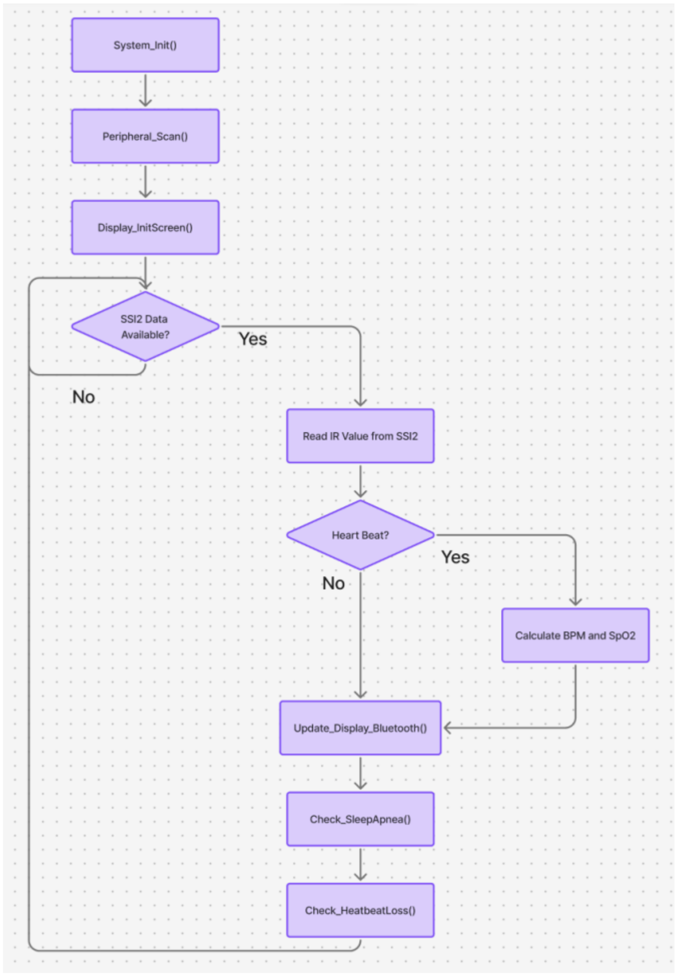


Fig. 3. Software Flowchart

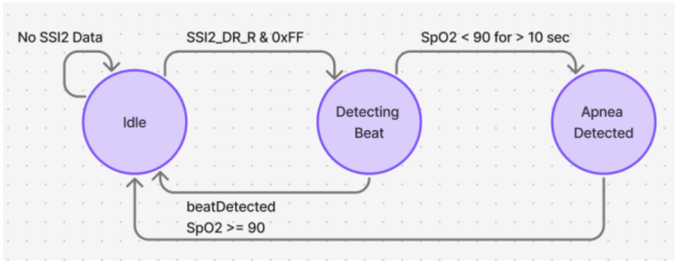


Fig. 4. Finite State Machine

C. *Implementations Challenges*

Several technical challenges were encountered during the design and integration phases of the Sleep Apnea Monitoring System.

a) SPI Communication Synchronization

Initially, the system was designed to transmit four fields over SPI communication: [START][IR VALUE][BPM][STOP]. However, during early testing, it was observed that only the first byte [START] was successfully received, while the subsequent fields were missing or corrupted. An interruption was added to flush any corrupted data, but that did not prove successful. Further investigation revealed that the Arduino Nano and TM4C123GXL microcontroller were not fully synchronized in terms of SPI clock and frame handling. Additionally, data alignment issues caused only the leading digit of the IR value to be transmitted.

An Arduino Serial output and Tiva UART output with the SPI alignment issue is shown in 5 and 6.

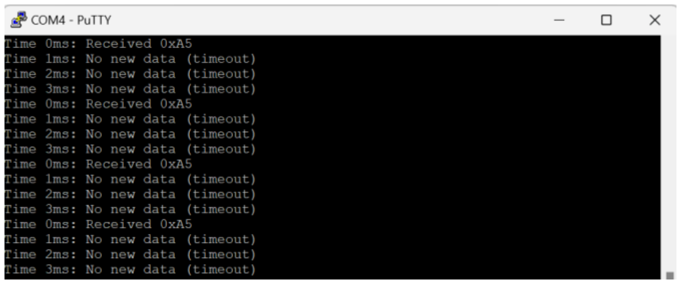

COM4 - PuTTY
```
Time 0ms: Received 0xA5
Time 1ms: No new data (timeout)
Time 2ms: No new data (timeout)
Time 3ms: No new data (timeout)
Time 0ms: Received 0xA5
Time 1ms: No new data (timeout)
Time 2ms: No new data (timeout)
Time 3ms: No new data (timeout)
Time 0ms: Received 0xA5
Time 1ms: No new data (timeout)
Time 2ms: No new data (timeout)
Time 3ms: No new data (timeout)
Time 0ms: Received 0xA5
Time 1ms: No new data (timeout)
Time 2ms: No new data (timeout)
Time 3ms: No new data (timeout)
```

Fig. 5. SPI Alignment Output (Tiva)

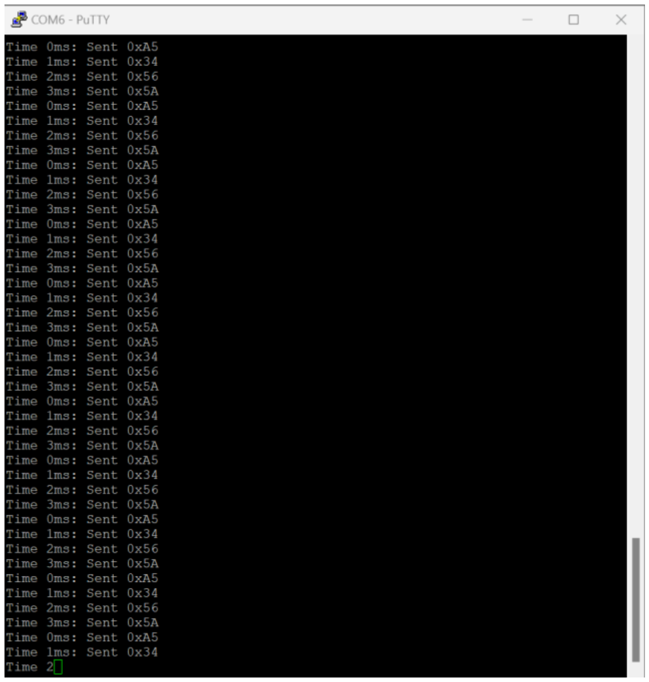

COM6 - PuTTY
```
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2ms: Sent 0x56
Time 3ms: Sent 0x5A
Time 0ms: Sent 0xA5
Time 1ms: Sent 0x34
Time 2
```

Fig. 6. SPI Alignment Output (Arduino)

To address this, the IR value was compressed, between 0 to 255, on the Arduino into a single byte before transmission, and later decompressed by the Tiva board. This workaround stabilized SPI communication at the expense of slight IR value resolution loss.

b) MAX30102 Direct Integration Failure

An attempt was made to eliminate the Arduino Nano by directly connecting the MAX30102 sensor to the TM4C123GXL using a custom driver library developed for TivaWare. However, despite successful I2C initialization, the sensor failed to detect a user's finger reliably. The MAX30102's infrared LED dimmed correctly upon proximity detection, but meaningful IR data was never acquired, with UART outputs consistently reporting either default or zero values. Due to time constraints, this approach was abandoned, and the Arduino Nano was incorporated to manage sensor interfacing.

The Tiva UART output with the custom MAX30103 library is shown in Fig 7.

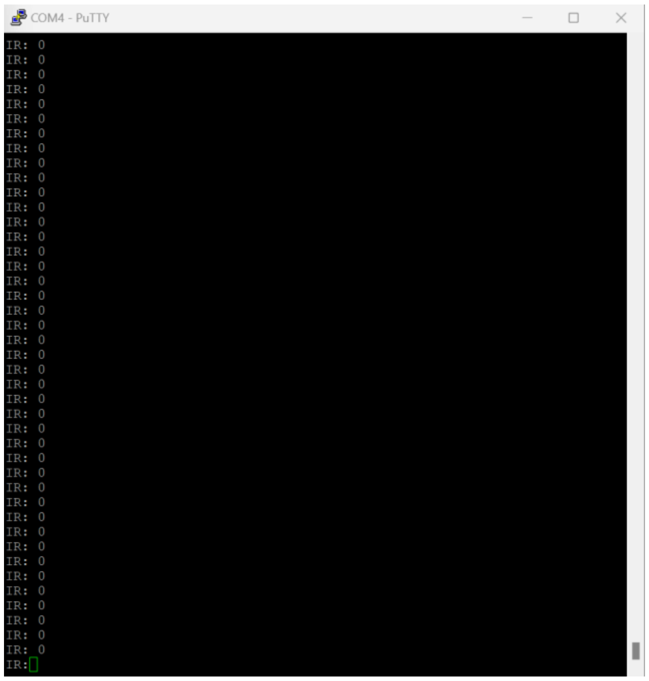

```
COM4 - PuTTY
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR: 0
IR:
```

Fig. 7. Tiva Output with custom MAX30102 driver.

c) HC-05 Malfunction

During the formation of the final product, the HC-05 integrated with the final prototype until the Bluetooth module was placed in the Arduino Nano. The initial prototype, as seen in Fig 9, sent the correct JSON data to a Windows machine. To address the malfunction, multiple attempts to troubleshoot and apply different parts were done which resulted unsuccessful. In the end, the HC-05 module was removed from the final prototype. However, the code still remains in the codebase for future implementations with HC-06 or HC-07. A successful output can be seen in Fig 8 and a draft of the prototype with the HC-05 is shown in Fig 9.

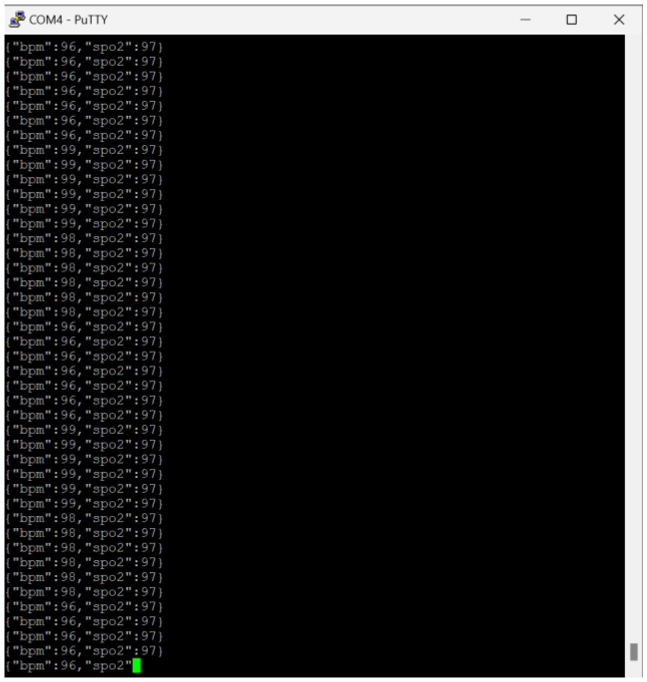

```
COM4 - PuTTY
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":99,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":98,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2":97}
{"bpm":96,"spo2"
```

Fig. 8. HC-05 JSON UART Output

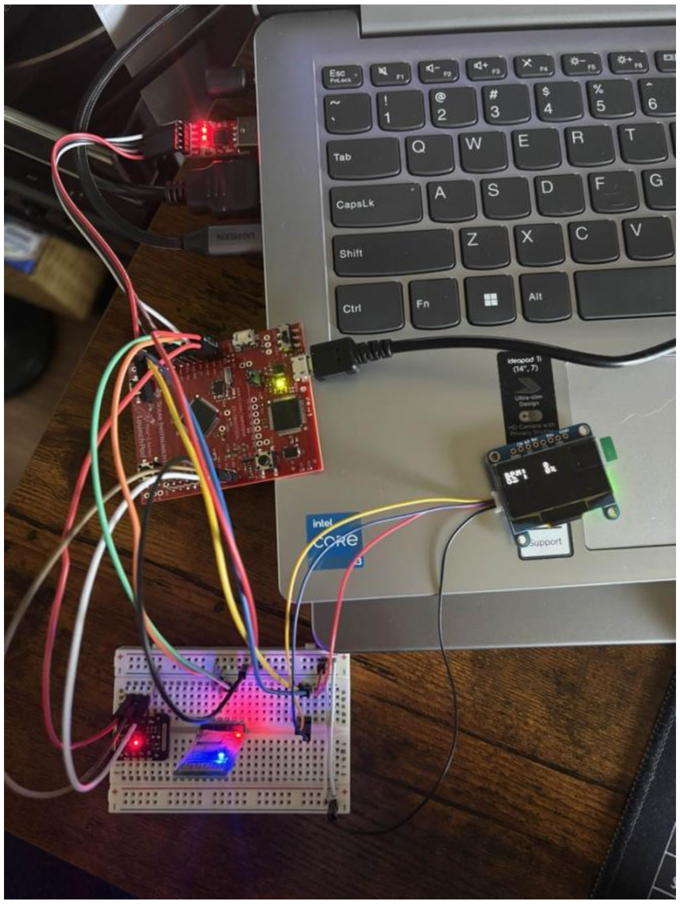


Fig. 9. HC-05 Prototype

d) Final Demonstration Malfunctions

On the day of the project demonstration, the heartbeat detection algorithm encountered unexpected failures, resulting in BPM readings defaulting to zero despite ongoing IR signal acquisition. It is hypothesized that either last-minute code revisions or peripheral instability led to data corruption in the beat detection module. Nevertheless, the IR reflectance signal continued to display expected rise-and-fall patterns, allowing a partial demonstration of live pulse monitoring. The BPM malfunction is shown in Fig 10.

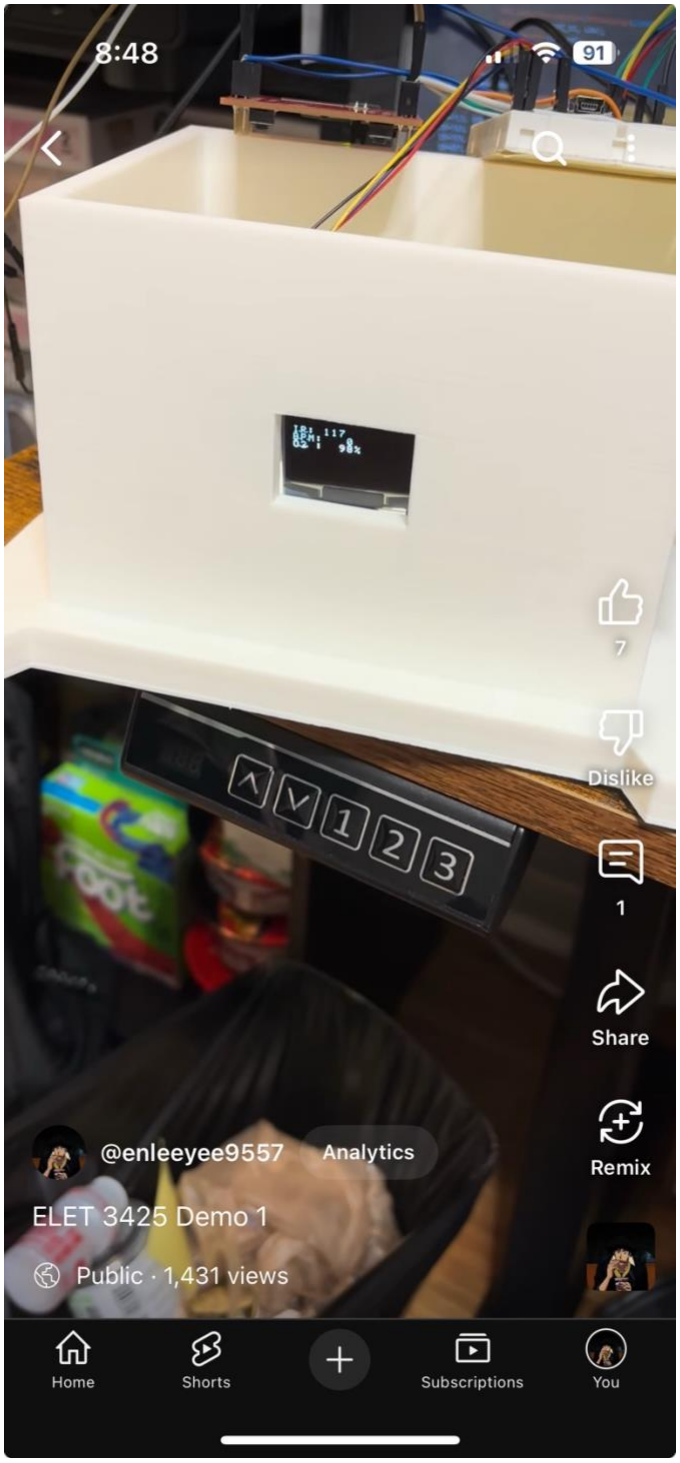


Fig. 10. BPM Malfunction OLED Display (from Demo Video)

e) Code Development and Repository Management

The development cycle for the project spanned approximately two weeks. Progressive commits and iterations can be reviewed in the project's public Github repository [8]. Regular code backups and modular testing were essential in mitigating critical errors prior to final integration.

The final assembled prototype is shown in Fig 11.

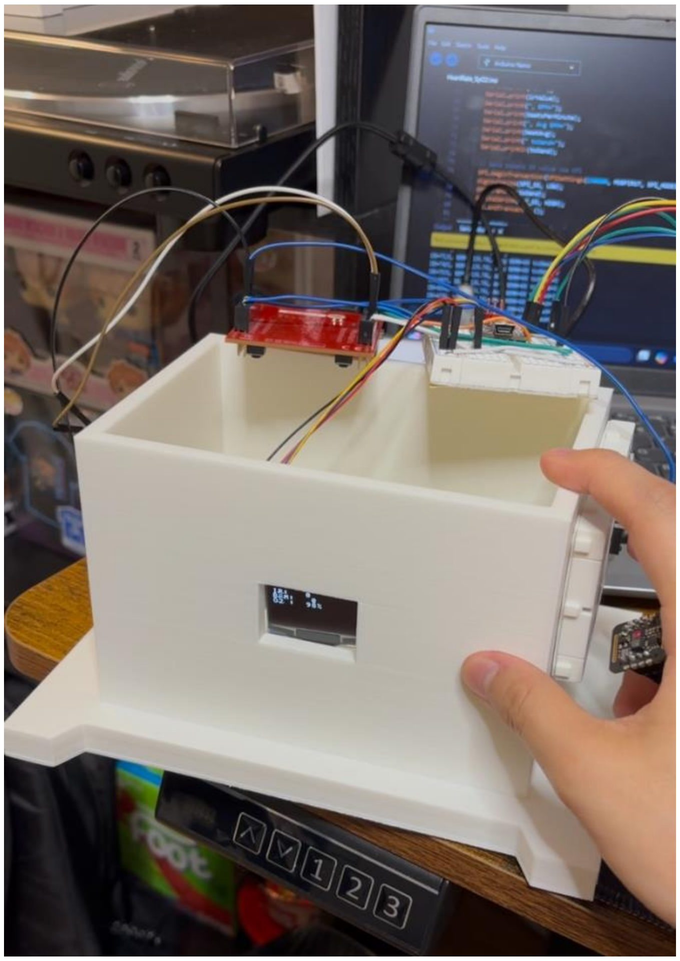
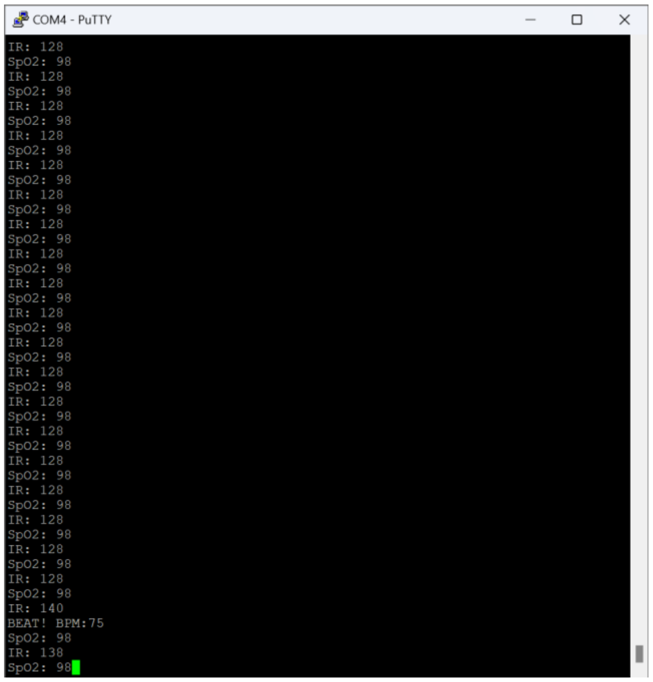


Fig. 11. Final Assembled Prototype of Sleep Apnea Monitoring System

D. *Test Procedures and Results*

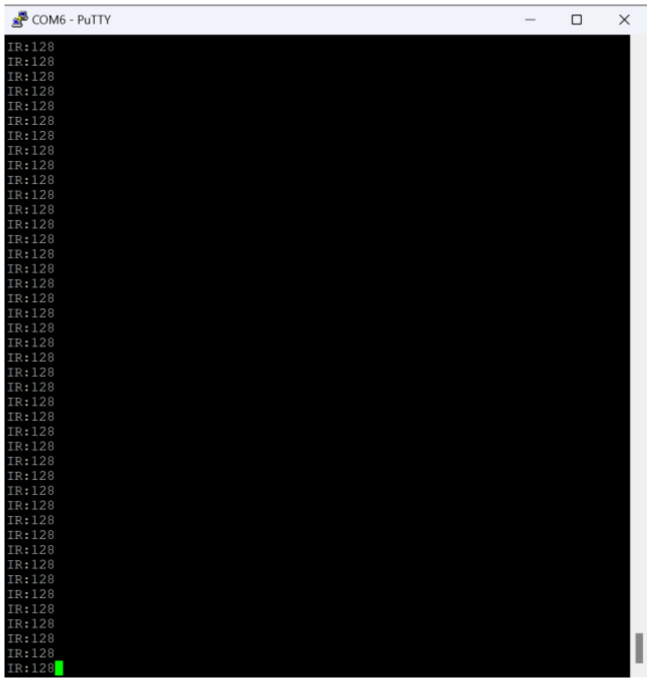
Comprehensive testing was conducted to validate system functionality and performance:

- Heart Rate Monitoring: IR reflectance values were collected using the MAX30102 sensor with a finger placed consistently over the sensing window. Detected BPM values ranged from 60 to 120 BPM depending on finger pressure and ambient conditions. Fig. 12 shows sample UART output logs.



```
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 128
SpO2: 98
IR: 140
BEAT! BPM:75
SpO2: 98
IR: 138
SpO2: 98
```

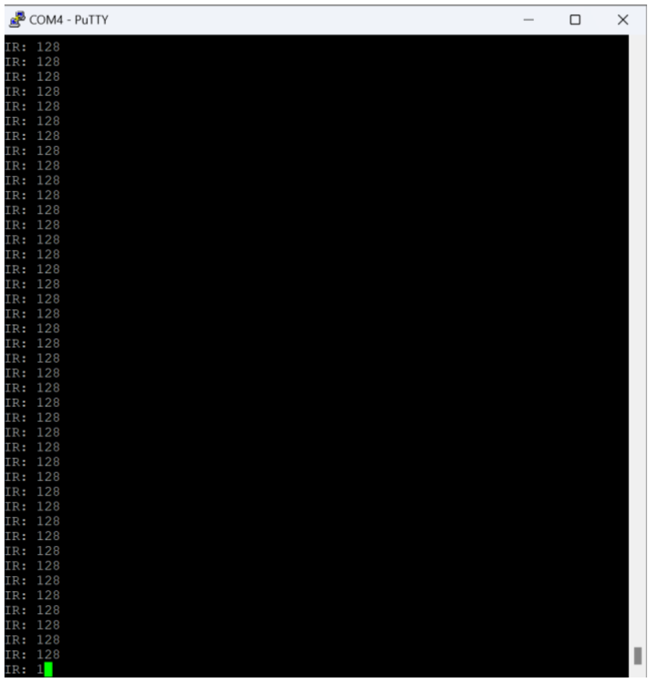
Fig. 12. UART IR, BPM, SpO2 output



```
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
IR:128
```

Fig. 13. Arduino Serial Output

- Apnea Event Simulation: Simulated apnea events by manually forcing a sustained low SpO2 condition. The system reliably trigged "Apnea Event!" notifications on both OLED and Bluetooth. Fig # shows notification via UART logs.
- SPI Communication Validation: SPI frame integrity was confirmed by reading Tiva UART logs and Arduino Serial logs after every frame reception. No frame drops or data corruption were observed at the 100 Hz sampling rate. Fig 13 and Fig 14 shows both Arduino and Tiva output logs.



```
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 128
IR: 1
```

Fig. 14. Tiva UART Output

- Power Measurement: Current consumption was measured at approximately 65 mA under normal operation, aligning with system power estimates.

E. Analysis of Findings

The experimental results support the following conclusions:

- Reliable Communication: SPI communication at 100 Hz operated reliably without frame loss, validating the design choice of polling-based SPI management.

- Accurate Beat Detection: The slope-based algorithm demonstrated consistent beat detection, particularly when IR signal smoothing was employed via a circular buffer averaging technique.
- Resilience to Interruptions: The system demonstrated robust handling of transient UART or Bluetooth interruptions without affecting real-time monitoring or OLED updating.
- Low Power Operation: System power consumption remained within acceptable bounds for portable, battery-powered health monitoring applications.

A summary of measured performance metrics is provided in Table V.

TABLE V. PERFORMANCE METRICS

Metric	***Measured Result***
Heart Rate Detection Range	60-120 BPM
SPI Communication Integrity	0% data loss at 100 Hz
Apnea Detection Threshold	10 seconds no beat / low SpO2
Power Consumption	~65 mA typical

V. SUMMARY AND CONCLUSION

This project demonstrates the feasibility of a real-time, embedded system-based health monitor for detecting sleep apnea. The combination of SPI-driven data acquisition, slope-based beat detection, real-time display, and Bluetooth alerting enables a fully embedded portable diagnostic system. Future directions include power optimization, live SpO2 sensing, and mobile app integration.

A. *Future Work and Improvements*

While the current prototype successfully demonstrates real-time sleep apnea monitoring capabilities, several avenues for future improvement and system enhancement were identified throughout the development and testing phases. These enhancements aim to increase the accuracy, reliability, usability, and portability of the device.

The current system approximates SpO_2 values without fully leveraging the red-to-infrared absorption ratio technique used in clinical-grade pulse oximeters. Future iterations should integrate real-time SpO_2 calculation by capturing both red and IR signals from the MAX30102 sensor. This will require implementing a more sophisticated algorithm based on the Beer-Lambert Law, including calibration against known oxygen saturation levels. Accurate SpO_2 monitoring would greatly enhance the diagnostic reliability of the system for detecting hypoxia-related apnea events.

One major limitation in the current design is the reliance on a secondary microcontroller (Arduino Nano) to manage the MAX30102 sensor. Future versions should fully integrate the sensor directly with the TM4C123GXL to reduce system complexity, power consumption, and cost. This would necessitate the development of a robust I2C driver and MAX30102 library compatible with TivaWare, addressing initialization timing, register configurations, and interrupt-based data handling to ensure reliable acquisition under real-world conditions.

The existing HC-05 Bluetooth module operates under Bluetooth Classic, limiting compatibility with iOS devices and increasing power consumption. Future versions should adopt a BLE module, such as the HM-10 or a Nordic nRF52832 module, to enhance wireless communication reliability, security, and energy efficiency. BLE would allow seamless integration with smartphones and tablets, opening opportunities for real-time app-based monitoring and notifications.

Building a dedicated mobile application would enable users to receive apnea alerts, visualize real-time heart rate and SpO_2 trends, and store historical sleep data for longitudinal analysis. The app could be developed for both Android and iOS platforms and could include features such as customizable alert thresholds, data export to healthcare providers, and daily/weekly health summaries. Integration with cloud services for optional data backup could further enhance the utility of the system.

Although the system's current power consumption is suitable for several hours of operation, longer battery life is essential for overnight monitoring sessions. Future improvements could include dynamic power management strategies such as MCU sleep modes, duty-cycled Bluetooth transmissions, and OLED partial updates to conserve energy. Incorporating a rechargeable lithium-ion battery with integrated charging circuitry (e.g., via USB) would also enhance portability and user convenience.

Real-world use introduces motion artifacts that can cause false heartbeat detections or missed apnea events. Future work should integrate accelerometer-based motion sensing (e.g., using an MPU-6050 module) to distinguish between true physiological signals and motion-induced noise. Data fusion techniques combining IR signals and motion data could significantly improve beat detection accuracy and event classification.

To increase user comfort and system adoption, miniaturizing the hardware into a compact, wearable form factor is an important future objective. Potential designs include a wristband, finger clip, or chest strap form factor, depending on sensing and comfort trade-offs. Flexible PCB technology, 3D-printed enclosures, and low-profile components would facilitate this transition.

Currently, the system relies on real-time visualization and transmission of physiological data. Adding local non-volatile memory (such as an SD card) would allow the device to record extended sleep sessions for later review. Offline data analysis could provide additional insights, such as apnea event frequency, duration, and severity trends over multiple nights.

REFERENCES

[1] MAX30102 Pulse Oximeter and Heart-Rate Sensor Datasheet, Analog Devices, [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/100/779/original/2011241005_UG-Univision-Semicon-UG-2864KSWLG01_C113322.pdf?1616084674

[2] TM4C123GH6PM Microcontroller Datasheet, Texas Instruments, [Online]. Available: https://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf

[3] Arduino Nano Datasheet, Arduino Documentation, [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

[4] HC-05 Bluetooth Module Datasheet, Components101, [Online]. Available: https://components101.com/sites/default/files/component_datasheet/HC-05%20Datasheet.pdf

[5] UG-2864KSWLG01 OLED Display Datasheet, Univision Technology Inc., [Online]. Available: https://cdn-learn.adafruit.com/assets/assets/000/100/779/original/2011241005_UG-Univision-Semicon-UG-2864KSWLG01_C113322.pdf

[6] American Sleep Apnea Association, "Sleep Apnea Information," [Online]. Available: https://www.sleepapnea.org/learn/sleep-apnea/

[7] Mayo Clinic, "Sleep Apnea - Symptoms and Causes," [Online]. Available: https://www.mayoclinic.org/diseases-conditions/sleep-apnea/symptoms-causes/syc-20377631

[8] Figma Design Platform, Sleep Apnea Graphs and Design, [Online]. Available: https://www.figma.com/board/jJCEzf1YQdYQ9xFIipiRPd/Sleep-Apnea-Graphs-and-Design?node-id=0-1&t=ZVviEvkGPpEDKuSb-1

[9] Enlai Yii et al., "Sleep Apnea Monitoring System," GitHub Repository. [Online]. Available: https://github.com/enleeyee/Sleep-Apnea-Monitor-System

APPENDIX

CODE Github Repository: https://github.com/enleeyee/Sleep-Apnea-Monitor-System