Final Documentation

Team 03
07.01.2025

1 Team members

Youssef Dokkar
Dapsara Kapuge
Kashif Raza

2 Introduction

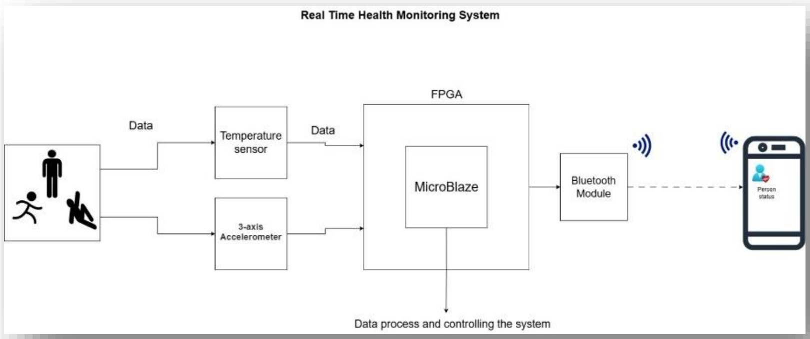
The FPGA-Based Health Monitoring System is an innovative solution designed to address the growing need for real-time and reliable health monitoring. This system utilizes the Nexys A7 FPGA board as its core processing unit, integrating multiple PMOD sensors to capture vital health parameters such as body temperature and movement. The acquired data is processed using custom VHDL IPs, ensuring high accuracy and minimal latency, and transmitted wirelessly to a mobile device via a Bluetooth Low Energy (BLE) module. With its emphasis on real-time data acquisition, efficient processing, and seamless communication, this system provides a robust and scalable platform for healthcare monitoring applications. The project combines the advantages of FPGA technology, custom IP development, and wireless communication to create a user-friendly and adaptable health monitoring solution.

The primary application of this prototype lies in healthcare, targeting use cases such as elderly care, fitness tracking, and post-recovery monitoring. For elderly individuals, the system monitors movement patterns and vital signs, ensuring timely interventions during emergencies, such as falls. Fitness enthusiasts benefit from real-time activity tracking, enabling them to optimize their training and health routines. Additionally, the system supports rehabilitation by monitoring mobility and temperature changes, providing valuable insights for post-recovery care.

FPGA-based IPs are an excellent fit for this project due to their ability to handle real-time data processing with exceptional speed and precision. FPGAs offer unparalleled hardware-level parallelism, enabling simultaneous processing of multiple data streams, which is critical for health monitoring applications. The reconfigurable architecture allows for easy integration of additional sensors and functionalities, ensuring scalability and adaptability. Furthermore, custom VHDL IPs provide precise control over signal processing tasks, reducing latency and enhancing reliability. These features make FPGAs a superior choice compared to traditional microcontroller-based systems for implementing health monitoring systems that demand high performance, low latency, and scalability. [1]

3 Concept description

The FPGA-Based Health Monitoring System is built around the idea of using FPGA technology to efficiently and accurately monitor health parameters in real-time. The system is designed to measure body temperature and detect motion or falls using a combination of PMOD sensors. The temperature sensor records health vitals, while the accelerometer tracks activity levels such as standing, moving, or falling. Data collected from these sensors is processed using custom VHDL IPs integrated into the FPGA's hardware, ensuring fast and reliable performance. A Micro-Blaze processor orchestrates the operation, including signal processing and communication, while a PMOD BLE module wirelessly transmits the processed data to a mobile device for real-time visualization.



The block diagram for the health monitoring system comprises several key components:

- *FPGA (Nexys A7 100T): Central processing unit.*
- *PMOD Sensors:*
    - *Temperature Sensor (PMOD TMP): For recording body temperature.*
    - *Accelerometer Sensor (PMOD ACL): For detecting motion and activity states.*
- *Custom VHDL IPs: Dedicated signal processing units for filtering and analyzing data from sensors.*
- *MicroBlaze Processor: Manages overall system operations, including communication and data handling.*
- *PMOD BLE Module: Facilitates wireless transmission of processed data to a mobile device.*
- *Mobile Application: Receives and displays health metrics in a user-friendly format.*

The following paragraphs describe the essential IPs used in the design:

- ***PMOD Interfaces***

The block design includes IPs for interfacing with the PMOD sensors and communication modules:
PMOD BLE (Bluetooth Low Energy) Module IP: This IP enables the system to transmit health metrics wirelessly to a mobile application. It uses the UART interface for communication with the MicroBlaze processor. [1]
PMOD ACL (Accelerometer) Module IP: This IP handles the accelerometer data, allowing the system to capture movement and activity information, which is critical for detecting standing, moving, or falling states. [1]

- ***AXI IIC for Temperature Sensor***

The AXI IIC (Inter-Integrated Circuit) IP is included for interfacing with the temperature sensor. This IP allows precise data acquisition from the connected temperature sensor, which is crucial for monitoring body temperature in real-time. [1]

- ***AXI UART for Communication***

The AXI UARTlite IP enables serial communication, allowing the system to interface with external devices for debugging or additional data exchange. This IP facilitates seamless communication between the hardware components and the MicroBlaze processor. [1]

- ***MicroBlaze Processor and Supporting IPs***

The MicroBlaze Processor is the main processing unit in the design. It manages all data acquisition, processing, and communication tasks. Supporting IPs connected to the MicroBlaze include:

AXI Interconnect: Ensures efficient communication between the processor and peripheral components, such as the PMOD modules and sensors.
AXI Interrupt Controller: Handles interrupts from the peripherals, ensuring real-time responsiveness.
Local Memory IP: Provides temporary storage for data during processing.

All processing and signal management tasks are handled within the MicroBlaze processor, including filtering temperature data and analyzing accelerometer readings. This integration simplifies the design and ensures efficient operation. [1]

- ***Clocking and Debugging***

The Clock Wizard IP generates clock signals for the FPGA system, ensuring synchronized operation across all components. The MicroBlaze Debug Module (MDM) is included to facilitate debugging during development, helping to identify and resolve issues efficiently. [1]

4 Project/Team management

	Phase 1	***Phase 2***	***Phase 3***	***Phase 4***
Youssef	*Concept*	*Implementation*	*IP block design*	*Implementation in Vitis*
Dapsara	*Concept*	*Implementation*	*IP block design*	*Implementation in Vitis*
Kashif	*Concept*	*Implementation*	*IP block design*	*Implementation in Vitis*

This project was developed collaboratively, with each team member contributing to every phase. The distribution of tasks was structured to ensure equal participation in the conceptualization, implementation, design, and final programming phases.

5 Technologies

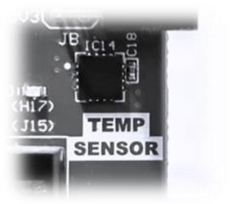
FPGA Board: Nexys A7
The Nexys A7 100T FPGA board is the primary platform for implementing the health monitoring system. It features the following:

- *Programmable Logic: 15,850 logic slices, enabling the creation of custom IP cores.*
- *Memory: 128 MiB of DDR2 RAM, suitable for real-time data processing.*
- *Connectivity: Four PMOD ports for sensor integration and peripheral devices.*
- *Clock Speed: Operates at 450 MHz, providing high-performance computation.*
- *Advantages: Ideal for prototyping due to its midrange performance, cost-efficiency, and compatibility with Xilinx tools like Vivado and Vitis. [1]*

[1] Nexys A7 Board

Temperature Sensor
The Nexys A7 FPGA features an onboard analog temperature sensor, the ADT7420, which is ideal for monitoring real-time body temperature. Key specifications include:

- *Resolution: 16-bit for high precision.*
- *Accuracy: ±0.25°C, suitable for health monitoring applications.*
- *Interface: Integrated into the FPGA, eliminating the need for additional PMOD modules.*
- *Advantage: Simplifies system design by using a built-in component, ensuring reliability and reducing hardware complexity. [1]*



[1] Temperature sensor (ADT7420) on Nexys A7 Board

Accelerometer Sensor (PMOD)
The accelerometer PMOD module detects user activity, including movement and falls. Key features include:

- ***Sensitivity:** Measures acceleration along multiple axes.*
- ***Detection Capabilities:** Capable of identifying static states (e.g., standing) and dynamic activities (e.g., falling).*
- ***Interface:** SPI communication for reliable data transmission to the FPGA. [1]*

[1] PMOD ACL

Bluetooth Low Energy (BLE) Module (PMOD)
The BLE module facilitates wireless communication between the FPGA and a mobile device. Key characteristics:

- ***Bluetooth Version:** 4.2 Low Energy, ensuring efficient and secure communication.*
- ***Interface:** UART for seamless integration with the FPGA.*
- ***Encryption:** AES128 encryption for secure data transmission.*
- ***Configuration:** Remote configuration over-the-air for flexibility. [1]*

[1] PMOD BLE

Xilinx Vivado Design-suite
Vivado serves as the primary tool for hardware design and synthesis, streamlining the creation of the FPGA design for the health monitoring system. It facilitates the integration of MicroBlaze into the Nexys A7-100T Digilent board, thanks to its modular and customizable architecture. This architecture

allows seamless incorporation of various IP cores, such as communication modules, sensor interfaces, and logic blocks tailored specifically for the health monitoring application.
Vivado provides comprehensive support throughout the FPGA development process, including:

- *Hardware Description Language (HDL) Design*
- *Design Entry*
- *Synthesis*
- *Implementation*
- *IP Integration*
- *Debugging and Verification. [2]*

In this project, Vivado 2022.1 was utilized to integrate MicroBlaze and establish connections with onboard and external sensors. It also supported debugging, ensuring the correctness of the implementation and synthesis processes.

Xilinx Vitis IDE
The Vitis Integrated Development Environment (IDE) plays a crucial role in achieving the objectives of the health monitoring system by supporting the development and management of software applications on the MicroBlaze processor embedded within the Xilinx FPGA. Key functionalities of Vitis IDE utilized during this project include:

- *Unified Development Environment:*
  Vitis IDE serves as the central platform for developing and managing embedded software applications, ensuring streamlined integration with the FPGA hardware design.
- *Programming Language Support:*
  The IDE supports multiple high-level programming languages, such as C and C++. For this project, C programming was used to meet the specific requirements of the MicroBlaze processor.
- *Debugging and Performance Analysis:*
  Vitis IDE provides advanced debugging tools and performance analysis features, enabling reliable and efficient implementation of the health monitoring system. [1]

6 IP Implementation

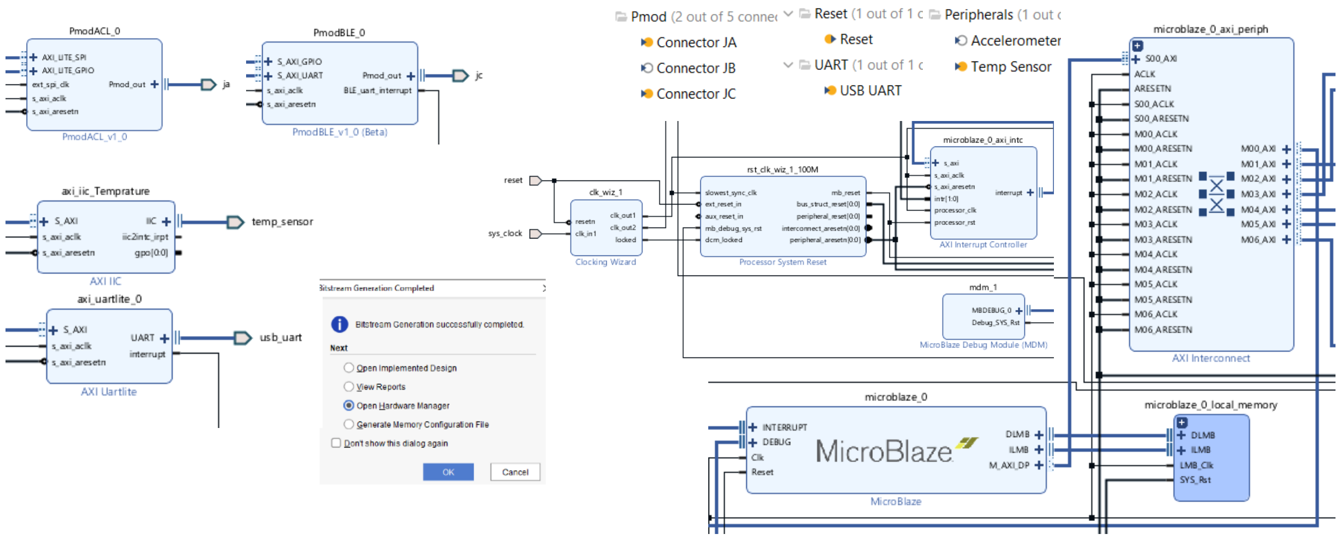
*The hardware implementation starts with the creation of the **IP Block Design** using the **Xilinx Vivado Design Suite**. At the heart of the system is the **MicroBlaze soft processor**, which serves as the main controller for managing sensor data acquisition, processing, and communication. To ensure synchronization across all system components, a **Clocking Wizard** is used to generate the required clock frequencies. The **AXI Interconnect** is employed to enable seamless communication between the MicroBlaze processor and various peripheral devices, ensuring efficient data transfer and system integration.*

*The design incorporates several essential modules. The **ADT7420 onboard temperature sensor**, integrated via the **AXI IIC IP core**, facilitates real-time temperature acquisition using the I2C protocol. This sensor provides highly accurate temperature readings with a 16-bit resolution and an accuracy of ±0.25°C, making it suitable for health monitoring applications. Additionally, the **PMOD ACL accelerometer** is connected to the JA PMOD port and interfaced using the **AXI GPIO IP core**. This module enables the detection of movement and activity, including standing, falling, or dynamic motion, providing critical health-related data. Wireless communication is achieved using the **PMOD BLE module**, which is attached to the JC PMOD port and interfaced via the **AXI UARTlite IP core**. The BLE module enables real-time transmission of processed health metrics, such as temperature and activity status, to a mobile device through Bluetooth Low Energy (BLE) communication. To ensure* *timely and responsive processing of data, an **Interrupt Controller** is integrated to manage sensor-generated interrupts effectively.*

*After finalizing the IP block design, the next phase involves generating the bitstream for the FPGA. This process comprises three main steps. First, **synthesis** converts the HDL design into a netlist, preparing it for further processing. Then, the **implementation** phase maps the netlist onto the FPGA resources, optimizing the placement and routing to improve performance and reduce latency. Finally, the **bitstream generation** step produces the final bitstream file, which is programmed onto the Nexys A7 FPGA board for execution. Once the hardware functionality was validated through simulation and on-board debugging, the completed design was exported to Xilinx Vitis IDE. This marks the transition to the software development phase, where the MicroBlaze processor is programmed to acquire sensor data, process it, and enable real-time communication via the BLE module.*

The design was thoroughly verified in two phases. First, simulation within Vivado was performed to ensure the functionality of the integrated modules and their interactions. Second, on-board debugging validated the real-time acquisition and processing of data, ensuring the system operated as intended. The results confirmed the successful integration of all components. The onboard temperature sensor provided precise real-time readings, the accelerometer reliably detected movement and transitions, and the BLE module enabled effective wireless communication with a mobile device.

*Below are the visuals that represent key aspects of the hardware implementation. The **IP Block Design** highlights the integration of the MicroBlaze processor and the peripheral modules. These screenshots provide a clear overview of the hardware configuration and its functionality. This robust hardware implementation lays the foundation for the software phase, enabling real-time, precise health monitoring and efficient data communication.*



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

( ) Open Implemented Design

( ) View Reports

(●) Open Hardware Manager

( ) Generate Memory Configuration File

☐ Don't show this dialog again

OK Cancel

Here are snippets showcasing the hardware design stages in Vivado, including MicroBlaze integration, peripherals, address mapping, and bitstream generation.

7 Software Implementation

The software implementation begins with the configuration of the MicroBlaze soft processor in Xilinx Vitis IDE, which acts as the central controller for managing sensor data acquisition, processing, and communication. The processor is equipped with 128 KB of local instruction and data memory to enable efficient task execution. Peripheral drivers for the AXI IIC IP core (for the onboard temperature sensor), AXI GPIO (for the accelerometer), and AXI UARTlite (for the PMOD BLE module) are initialized *to facilitate communication with the connected hardware. These drivers ensure that the software and hardware work seamlessly together.*

*Several key libraries are included to manage the functionality of the system effectively. The **xiic.h** library enables communication with the onboard ADT7420 temperature sensor using the I2C protocol. The **PmodACL.h** library is utilized to interface with the accelerometer, while **PmodBLE.h** handles communication with the BLE module. The **xparameters.h** library connects the software with the hardware design by mapping the hardware addresses and IDs. Additionally, the **math.h** library provides support for mathematical calculations, such as computing the magnitude of acceleration to detect falls. For example, the temperature sensor is initialized and configured using the xiic.h library, which allows the software to read and process temperature data in real time.*

```
#include <stdio.h>
#include "xiic.h" // For temperature sensor communication
#include "PmodACL.h"  // For accelerometer control
#include "PmodBLE.h"  // For BLE communication
#include "xparameters.h" // Hardware-software connection
#include "math.h" // For calculations
```

The initialization process is handled by the HealthMonitorInit() function, which sets up all required modules. This function enables instruction and data caches for efficient memory access and configures the accelerometer (PMOD ACL), BLE module, and UART for communication. The BLE buffer is cleared to avoid processing residual data, ensuring that the system starts with a clean slate. During this initialization, the system outputs a green message indicating that the health monitor system is starting, followed by a confirmation message that the system is running.

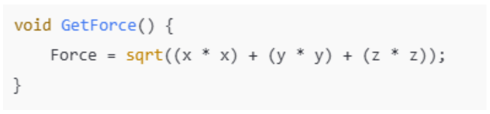
```
void HealthMonitorInit(){
    EnableCaches();
    ACL_initialize();
    SysUartInit();
    BLE_Init();
    emptyBLEbuf();

    printf("\033[32mHealth Monitor system is starting ...\n");
    sleep(3);
    printf("System is running :) \n");
    printf("\033[0m");
}
```

Temperature monitoring is performed by the GetTempValue() function, which reads raw data from the ADT7420 temperature sensor and converts it into Celsius. This data is then processed by the CheckTemp() function, which evaluates whether the temperature is within safe limits. If the temperature exceeds a threshold (below 20°C or above 25°C), warnings are displayed in red on the serial monitor and transmitted to the BLE device. For example, when the temperature is too high, the system sends a warning message such as "Patient's Temperature is too high" through BLE while also printing the message on the serial monitor with appropriate color coding.

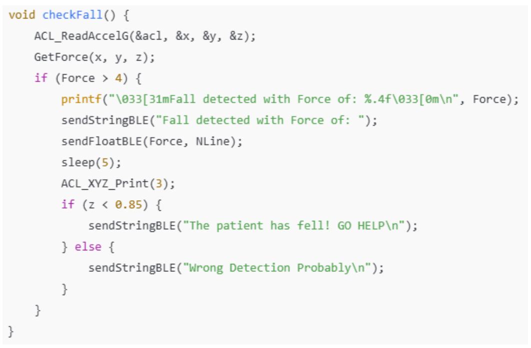
```
void GetTempValue() {
    int16_t temp_raw = 0;
    uint8_t temp_data[2];

    XIic_Recv(IIC_DEVICE_ID, Temp_address, temp_data, 2, XIIC_STOP);
    temp_raw = temp_data[0] << 5 | (temp_data[1] >> 3);
    temperature = temp_raw * 0.0625f; // Convert raw data to Celsius
}
```

```
void CheckTemp() {
    GetTempValue();
    if (temperature < 20) {
        printf("Patient's Temperature dangerously \033[31mlow\033[0m: %.4f °C\n", temperature);
        sendStringBLE("Patient's Temperature is too low: ");
        sendFloatBLE(temperature, NLine);
    } else if (temperature > 25) {
        printf("Patient's Temperature dangerously \033[31mhigh\033[0m: %.4f °C\n", temperature);
        sendStringBLE("Patient's Temperature is too high: ");
        sendFloatBLE(temperature, NLine);
    }
}
```

Fall detection is a critical feature of the system, managed by the checkFall() function. This function calculates the magnitude of acceleration (force) using the GetForce() function, which relies on the math.h library. If the force exceeds 4 G, the system identifies a potential fall and sends a warning through BLE and the serial monitor. The warning includes the force value and the X, Y, Z acceleration components. After a 5-second delay, the system rechecks the orientation to confirm the fall. If the fall *is confirmed, a repeated "GO HELP" message is sent through BLE, ensuring that the alert is noticed. If the fall is determined to be a false positive, the system sends messages indicating that the detection was likely incorrect but advises checking the patient nonetheless.*

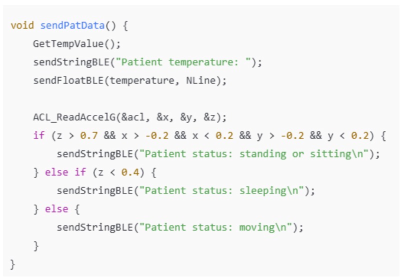


```c
void GetForce() {
    Force = sqrt((x * x) + (y * y) + (z * z));
}
```



```c
void checkFall() {
    ACL_ReadAccelG(&acl, &x, &y, &z);
    GetForce(x, y, z);
    if (Force > 4) {
        printf("\033[31mFall detected with Force of: %.4f\033[0m\n", Force);
        sendStringBLE("Fall detected with Force of: ");
        sendFloatBLE(Force, NLine);
        sleep(5);
        ACL_XYZ_Print(3);
        if (z < 0.85) {
            sendStringBLE("The patient has fell! GO HELP\n");
        } else {
            sendStringBLE("Wrong Detection Probably\n");
        }
    }
}
```

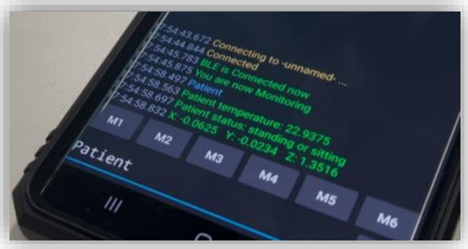
The system can also respond to patient data requests via BLE. The sendPatData() function verifies whether the correct patient name is received. If the name matches, the system sends the temperature, activity status, and accelerometer values (X, Y, Z) through BLE. For instance, if the patient is standing or sitting, the system sends "Patient status: standing or sitting" along with accelerometer readings. If the name does not match, the BLE device receives a feedback message indicating that the name was not found.



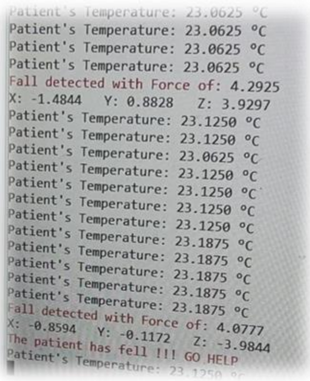
```c
void sendPatData() {
    GetTempValue();
    sendStringBLE("Patient temperature: ");
    sendFloatBLE(temperature, NLine);

    ACL_ReadAccelG(&acl, &x, &y, &z);
    if (z > 0.7 && x > -0.2 && x < 0.2 && y > -0.2 && y < 0.2) {
        sendStringBLE("Patient status: standing or sitting\n");
    } else if (z < 0.4) {
        sendStringBLE("Patient status: sleeping\n");
    } else {
        sendStringBLE("Patient status: moving\n");
    }
}
```

The software also manages BLE connection status, ensuring real-time feedback for the user. When the BLE device connects, the system sends a confirmation message to the BLE device, and the serial monitor displays "BLE is Connected to a device" in green. Similarly, when the BLE device disconnects, the serial monitor displays a red warning message, "BLE device is disconnected."

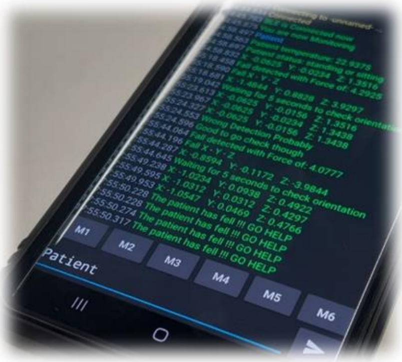
The output of the system reflects the seamless integration of its components. Under normal conditions, temperature readings are displayed on the serial monitor every 1.5 seconds. If the temperature is dangerous, warnings are printed on the serial monitor and sent via BLE with increased frequency and color-coded alerts. For fall detection, the system provides detailed force values and accelerometer readings to help users assess the situation manually. Additionally, patient-specific data requests ensure personalized monitoring, while BLE feedback keeps the user informed about the system's status. The implementation effectively combines data acquisition, real-time processing, and communication to create a robust health monitoring system.



```
7:54:43.672 Connecting to -unnamed- ...
7:54:44.844 Connected
7:54:45.783 BLE is Connected now
7:54:45.875 You are now Monitoring
7:54:58.497 Patient
7:54:58.563 Patient temperature: 22.9375
7:54:58.697 Patient status: standing or sitting
7:54:58.832 X: -0.0625  Y: -0.0234  Z: 1.3516
```



```
Patient's Temperature: 23.0625 °C
Patient's Temperature: 23.0625 °C
Patient's Temperature: 23.0625 °C
Patient's Temperature: 23.0625 °C
Fall detected with Force of: 4.2925
X: -1.4844   Y: 0.8828   Z: 3.9297
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.0625 °C
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.1250 °C
Patient's Temperature: 23.1875 °C
Patient's Temperature: 23.1875 °C
Patient's Temperature: 23.1875 °C
Patient's Temperature: 23.1875 °C
Patient's Temperature: 23.1875 °C
Fall detected with Force of: 4.0777
X: -0.8594   Y: -0.1172   Z: -3.9844
The patient has fell !!! GO HELP
Patient's Temperature: 23.1250 °C
```



```
Connecting to -unnamed- ...
Connected
BLE is Connected now
You are now Monitoring
Patient
Patient temperature: 22.9375
Patient status: standing or sitting
X: -0.0625  Y: -0.0234  Z: 1.3516
55:18.458 Fall detected with Force of: 4.2925
55:18.590 Fall X - Y - Z
55:18.681 X: -1.4844  Y: 0.8828  Z: 3.9297
55:19.042 Waiting for 5 seconds to check orientation
55:23.613 X: -0.0625  Y: -0.0156  Z: 1.3516
55:23.967 X: -0.0625  Y: -0.0234  Z: 1.3438
55:24.327 X: -0.0625  Y: -0.0156  Z: 1.3438
55:24.553 Wrong Detection Probably
55:24.596 Good to go check though
55:44.064 Fall detected with Force of: 4.0777
55:44.196 Fall X - Y - Z
55:44.287 X: -0.8594  Y: -0.1172  Z: -3.9844
55:44.645 Waiting for 5 seconds to check orientation
55:49.238 X: -1.0234  Y: 0.0938  Z: 0.4922
55:49.595 X: -1.0312  Y: 0.0312  Z: 0.4297
55:49.953 X: -1.0547  Y: 0.0469  Z: 0.4766
55:50.228 The patient has fell !!! GO HELP
55:50.228 The patient has fell !!! GO HELP
55:50.274 The patient has fell !!! GO HELP
55:50.317 The patient has fell !!! GO HELP
```

8 Implementation Challenges

During the project development, several challenges were encountered, primarily related to version compatibility, memory allocation, and platform configuration. These issues required detailed debugging and configuration updates to ensure the system functioned as intended.

Version Compatibility Issues
One significant challenge was the compatibility between different versions of Vivado and Vitis. When using newer versions, errors appeared in the parameter files during platform creation. Additionally, example code often failed to run due to incompatibilities. To resolve this, Vivado 2022.1 was used, which eliminated most issues except for makefile-related errors. These were addressed by manually updating the makefile for each PMOD, regenerating the bitstream, and rebuilding the platform in Vitis. Guidance from AMD Adaptive Computing Support helped resolve these issues effectively.

Memory Allocation Challenges
The default BRAM size for the MicroBlaze processor was insufficient for the project's needs. To address this, new BRAM controllers were added in Vivado's block design and connected to the system. The Address Editor was used to assign addresses, and memory mapping was reorganized to expand the memory to 128 KB. The lscript.ld file in Vitis was updated to reflect the new memory configuration:

```
microblaze_0_local_memory_ilmb_bram_if_cntlr_Mem_microblaze_0_local_memory_dlmb_bram_if_cntlr_Mem : ORIGIN = 0x50, LENGTH = 0x3FFB0
```

This adjustment ensured sufficient memory for the application while maintaining stability.

Outcomes
After resolving these challenges, the platform was successfully built using Vivado 2022.1, and the updated memory configuration eliminated allocation issues. The system now supports robust data acquisition and processing, with stable performance for real-time health monitoring.

9 Sources/References

[1] *"Digilent reference," accessed: 2024-01-10. [Online]. Available: https: //digilent.com/reference/*
[2] *S. Alonso, J. L´azaro, J. Jimenez, U. Bidarte, and L. Muguira, "Evaluating latency in multiprocessing embedded systems for the smart grid," Energies, vol. 14, no. 11, p. 3322, 2021.*