|  |  |  |
| --- | --- | --- |
| Technical Design (TD)  Project: integration | | |
| **Enschede, 11 April 2025** | | |
| **Version 0.**1 | | |
| **Made by :** | **Group: 5** |  |
|  | **Mo van Elk** | 462818 |
|  | **Konstantin Apostolov** | 552608 |
|  | **Jort Bolscher** | 556491 |
|  | **Artur Golovko** | 547206 |
|  | **Tom te Lintelo** | 532911 |

Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| TD | Technical Design |
| FR | Functional Requirement |
| NA | Not Applicable |
| PCB | Printed Circuit Board |
| TR | Technical Requirement |
| API | Application Programming Interface |
| ESP32 | Microcontroller with Wi-Fi and Bluetooth capabilities used for IoT applications |

Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| FreeRTOS | A real-time operating system used on microcontrollers for managing multiple tasks. |
| LiPo Battery | A rechargeable lithium-polymer battery commonly used in wearable electronics. |
| BOM | Bill of Materials; a complete list of components used in a design. |
| I2C | A communication protocol that allows multiple sensors to connect to a microcontroller using only two wires. |

References

|  |  |  |
| --- | --- | --- |
| **Number** | **Author(s)** | **Title** |
| [1] | J.W. van Dijk | Project Based and Methodological Design: A Practical Approach |
| [2] | InvenSense | MPU6050 Datasheet |
| [3] | Analog Devices | ADXL345 Datasheet |
| [4] | Texas Instruments | AFE4490 Datasheet |
| [5] | World Famous Electronics | Pulse Sensor Technical Information |
|  |  |  |
|  |  |  |

## **Table of Contents**

1. **Introduction**............................................................................................................ 3  
     1.1 Introduction to Project.................................................................................3  
     1.2 Overview of Document ..................................................................................3
2. **Selecting Components** ............................................................................................ 4  
     2.1.1 Comparison Alternatives for Component 1: Accelerometer ....................... 4  
     2.1.2 Comparison Alternatives for Component 2: Optical Heart Rate Sensor ..... 4  
     2.1.3 Comparison Alternatives for Component 3: OLED Display ......................... 5
3. **Schematics and Drawings** ..................................................................................... 6  
     3.1 Electronic Schematics .................................................................................. 6  
       - Sensor Connections .................................................................................. 7  
       - Microphone Amplifier ................................................................................ 7  
       - Auto Programmer ...................................................................................... 8  
       - Voltage Regulators .................................................................................... 8  
       - Battery Charger ........................................................................................ 9  
       - Buttons and Switches ................................................................................ 9  
     3.2 PCB Design ................................................................................................. 10  
     3.3 Component List (Bill of Materials) ............................................................... 11  
     3.4 Software Design .......................................................................................... 12  
       3.2.1 Software Architecture Overview ..................................................... 12  
       3.2.2 List of Software Processes and Descriptions ................................. 13  
       3.2.3 API Description and Endpoints ....................................................... 14  
       3.2.4 Communication Between Parallel Processes ................................. 15  
       3.2.5 Flow of Control .................................................................................. 15
4. **Conclusions** ......................................................................................................... 16
5. **Abbreviations** ....................................................................................................... 1
6. **Glossary** ............................................................................................................... 1
7. **References** ............................................................................................................ 1

# Introduction

## Introduction to project

This project focuses on developing a smart healthcare device that monitors heart rate, fall detection, provides emergency alerts on falling and medication reminders. The device functions as a wearable watch that ensures patient safety through real-time monitoring and communication with caregivers in the given moment through desktop application and web applications. The user gets audible alerts if it is time to take in their medication and can see their own heartbeat on a display.

## Overview of document

In the previous functional design (FD) the concept is chosen.

In this document this concept is worked out in more depth.

In chapter 2 a couple of alternative components will be compared to each other to select which exact component will be chosen for the concept.

In chapter 3 the component list, electronic schematic and PCB design will be shown.

For the software part there will be a list of processes and functions, description of API’s, communication between parallel processes and functional descriptions, such as low-level flowcharts and pseudo code.

Chapter 3 ends with some considerations regarding the integration of different components to the whole system

The document ends with chapter 4, which gives a conclusion to the chosen components and worked out design.

# Selecting components

In this chapter the components will be selected. This is done by comparing the different alternatives to each other. In a table the alternatives will be scored according to a weighted criterion. The more +’s an alternative has the better it scores on the criteria.

Finally, the best component will be selected

### Comparison possible alternatives for component 1: “accelerometer”

The accelerometer is the chosen option for the fall detection parameter in the FD.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Criteria 4** | **Criteria 5** |
|  | Accuracy/sensitivity | I2C protocol | Size | Power consumption | Cost |
| **MPU6050** | 5 | 5 | 3 | 2 | 3 |
| **ADXL345** | 4 | 5 | 4 | 3 | 2 |
| **LIS3DH** | 4 | 5 | 5 | 5 | 5 |

Table 1 – Comparison possible alternatives for component 1

*The ADXL345 comes as best out of the test, because it is quite accurate and supports I2C, furthermore it has a low power consumption and is not too big. Compared to the others.*

### Comparison possible alternatives for component 2: “Optical heart rate sensor”

The optical heart rate sensor is the chosen option for the heartbeat detection parameter in the FD.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Criteria 4** | **Criteria 5** |
|  | Accuracy | I2C protocol | Size | Power consumption | Cost |
| **MAX 30102** | 5 | 5 | 5 | 4 | 3 |
| **AFE4490** | 3 | 5 | 3 | 5 | 2 |
| **Pulse sensor** | 5 | **1** | 3 | 3 | 5 |

Table 2 – Comparison possible alternatives for component 2

*The MAZ 30102 comes as best out of the test. It is small, accurate and supports I2C.*

### Comparison possible alternatives for component 3: “OLED display”

The OLED display is the chosen option for the User interface parameter in the FD.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Criteria 1** | **Criteria 2** | **Criteria 3** | **Criteria 4** |
|  | Readability/resolution | Size | Power consumption | Cost |
| SSD1305 (0.96") | 5 | 5 | 5 | 5 |
| SH1106 (1.3") | 5 | 4 | 3 | 5 |
| SSD1331 (Color) | 4 | 3 | 2 | 2 |

The SSD1305 comes as best out of the test, it has a high resolution, is big and low power consumption. It is also the cheapest.

Table 3 – Comparison possible alternatives for component 3

For the rechargeable LiPo battery it is not needed to do an Indepth comparison, as the form of battery is already quit low level. For the other chosen options, the audible ping and message on phone, a component selection is not needed. For the ESP32 it has already determined by the stakeholders which is going to be used.

# Schematics and drawings

In this chapter the electronic schematics and drawing will be shown.

For the software there will be descriptions of software processes, functions, API’s, communication and pseudo code/low level flowcharts.

## Electronic

This section provides detailed schematics and drawings to illustrate the electronic design clearly. Everything is drawn in Altium.

* **Electronic Schematics:**

Detailed circuit schematics illustrating the connectivity and interactions between sensors, microcontroller, and power management components.

A diagram of a computer

AI-generated content may be incorrect.The total electronical schematic:

Figure 1 electronical schematic

On the top left the ESP32 is placed. On the left side decoupling capacitors are placed. The enable pin is connected with the 3.3V on the right side.

Then on the top the sensors are placed. Here is described to which pin of the ESP they are connected. These are to measure, store and display data.

A diagram of a computer chip

AI-generated content may be incorrect.

Figure 2 electronical schematic sensors

Then we have the microphone amplifier to boost the sound and to be able to play sound.

A diagram of a circuit

AI-generated content may be incorrect.

Figure 3 electronical schematic mic amplifier

On the right side the self-designed microphone preamplifier is placed. The microphone is connected to the MCP602 OPAMP. On the left the backup module is placed for the microphone amplifier. The I2S amp is also placed here, connected to a speaker.

Then the auto programmer uploads the code to the ESP-32.  
A computer diagram of a circuit board

AI-generated content may be incorrect.

Figure 4 electronical schematic auto programmer

On top the FT232RL is placed, which is connected to a USB port where the data cable connects to. On te right side 2 LEDs are placed to see if it receives data. Above the LEDs the connections to the ESP are shown. On the bottom right the auto-reset circuit is placed. So, you don’t manually need to press buttons to upload a new code. On the bottom left the backup module is placed with the connections to the ESP32.

Then we have to voltage regulators. The left one converts the battery power to 3.3 V for the ESP-32. The left one converts USB power to 3.3 V. This one is a back-up in case the battery does not work.

A diagram of a circuit

AI-generated content may be incorrect.

Figure 5 electronical schematic voltage regulators

Then the battery charger

A diagram of a circuit

AI-generated content may be incorrect.

Figure 6 electronical schematic battery charger

On the left side the battery charger IC MAX1811 is placed. On the left it is connected to a USB port to deliver power. With an LED to check if it is charging. On the right side you can connect the battery to the circuit for charging. On the right side a backup module is placed. On the left side the power is delivered and on the right side the battery is connected.

Lastly two buttons

A diagram of a circuit

AI-generated content may be incorrect.

Figure 7 electronical schematic switches

These can be used to switch the display screen. The capacitors are used for debouncing

* **PCB Designs:** Complete PCB layout diagrams and circuit routing

A green circuit board with many small chips

AI-generated content may be incorrect.

Figure 8 PCB design

Here the total PCB is shown. This is the prototype variant where all the modules and back-up modules are placed on top of the PCB. Remember it is proof of concept. Because this size would be pretty unhandy to have on top of your wrist.

All the modules are placed on the right side. On the bottom left the battery charger circuit is placed and next to it the voltage regulator for the battery is placed.

Then under it the two buttons are placed. Then comes the self-designed microphone pre-Amplifier. In the right middle the voltage regulator for the back-up power is placed for 5V to 3.3V. Above it the programming circuit is placed. With the back-up programmer above it.

Beneath here the connections are shown between the components and modules. The blue is a ground plane to decrease the magnetic field, which decreases the noise and interference of the signals.

A blue circuit board with many different colored lines

AI-generated content may be incorrect.

Figure 9 PCB connectins

* **Component List (Bill of Materials - BOM):**

Comprehensive list with specifications, quantities.

Here all the components placed on the PCB are listed in an Altium made BOM.

The first collum describes what component the last one gives the quantity of the component.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 10 BOM

## Software

This section documents the software design for the smartwatch firmware, backend API, and user interfaces, following the modular architecture and algorithms discussed in FD Section 4.3 and the integration approach from FD Section 4.4.

**3.2.1 Software Architecture Overview:** The system employs a modular architecture consisting of:

* + **Smartwatch Firmware (Embedded C++):** Runs on the ESP32, responsible for sensor data acquisition, processing (HR, Fall Detection), UI management, alert generation (audio/visual), power management, and communication with the backend API via Wi-Fi. Arduino framework with FreeRTOS for task management.
  + **Backend API (C# using ASP.NET Core Web API):** Manages data storage (likely using Entity Framework Core with a database like SQL Server, as mentioned in FD 4.4), user authentication (TR0504), processes incoming health data, triggers notifications (e.g., to practitioner apps/SMS gateway - FD 3.2.6), and serves data to frontend applications.
  + **Frontend Applications (C# for Windows/Web, potentially Xamarin/MAUI for Mobile):** Provide user interfaces (FR0101) for data visualization (EIR0202), device configuration (EIR0203, EIR0204), and viewing alerts. Interacts with the Backend API.

**3.2.2 A complete list of processes + descriptions:** Key software processes include: SensorDataManager, FallDetectionAlgorithm, HeartRateMonitor, UIManager, ConnectivityManager, AlertManager, TimeKeeper, PowerManager.

|  |  |
| --- | --- |
| **Process** | **What it does** |
| **SensorDataManager** | Reads **ADXL345** (accel) and **MAX30102** (heart‑rate) every 50 ms and keeps the last 8 samples in memory. |
| **HeartRateMonitor** | Turns 8 infra‑red samples into a single **BPM** value. |
| **FallDetectionAlgorithm** | Calculates a = √(x²+y²+z²); if it jumps above a threshold it flags a fall. |
| **UIManager** | Shows either the time or the live BPM on the 0.96″ OLED; reacts to the two side‑buttons. |
| **AlertManager** | Plays a short beep and/or vibration when medication time or a fall is detected. |
| **ConnectivityManager** | When Wi‑Fi is available, bundles the latest BPM + fall events into JSON and **POSTs** to the backend. |
| **TimeKeeper** | Keeps track of the real‑time clock and syncs over NTP once per hour. |
| **PowerManager** | Monitors battery voltage and puts the ESP32 into light‑sleep when nothing is happening. |

* 

**3.2.3 Description of API’s (Application Programming Interfaces):**

* **Backend API:** A RESTful API, implemented using **ASP.NET Core Web API**, is defined for communication between the smartwatch/frontend applications and the server. Key endpoints (as per FD 4.4):
  + POST /api/health-data: Receives time-stamped heart rate, activity level, and fall detection events from the smartwatch.
  + POST /api/time: (Potentially used by device if NTP is unavailable) To get/sync time.
  + POST /api/notifications: Endpoint possibly used *by* the backend to trigger external notifications (e.g., SMS via Twilio) based on received health data. (Or GET for device to pull config).
  + GET /api/health-data?user\_id=...&timespan=...: Allows frontend apps to retrieve historical data for visualization.
  + GET/POST /api/config?user\_id=...: Allows frontend apps to manage medication schedules and alert thresholds.
* Data is exchanged in JSON format. Authentication (e.g., JWT bearer tokens managed via ASP.NET Core Identity) secures access.

**3.2.4 Description of communication between parallel processes:** *(No changes needed, protocols remain the same)*

* + **Intra-Device (ESP32 Firmware):** FreeRTOS tasks are used for concurrent handling of sensor reading (interrupt-driven where possible - TR0802), data processing, UI updates, and Wi-Fi communication. Queues and Semaphores manage inter-task communication and resource sharing.
  + **Device-to-Backend:** Asynchronous communication via HTTPS POST requests over Wi-Fi to the **ASP.NET Core API**. Data is buffered locally (e.g., on SD card or internal flash - TR0503, TR0809) if the network is unavailable, with a retry mechanism upon reconnection (addresses FD 4.4 challenges). WebSockets (via ASP.NET Core SignalR) or MQTT might be considered for lower latency real-time updates if required, but the current design uses REST.
  + **Frontend-to-Backend:** Standard HTTPS RESTful API calls (GET, POST) from the C# application to the **ASP.NET Core API** to retrieve data and post configuration changes. Libraries like HttpClient in .NET are used.

**3.2.5 Flow of Control (text version)**

SensorDataManager collects data.

HeartRateMonitor & FallDetectionAlgorithm crunch numbers.

UIManager + AlertManager react immediately.

ConnectivityManager ships data when the Wi‑Fi icon is green.

TimeKeeper & PowerManager run quietly in the background.

*Possible documentation that can be added here:*

* *A complete list of processes + descriptions.*
* *A complete list of functions + descriptions.*
  + *Short description of functional behaviour.*
  + *Parameters*
  + *Return values*
* *Description of API’s (Application Programming Interfaces).*
  + *A list of functions, parameters and return values (+ descriptions).*
* *Description of communication between parallel processes*
  + *Descriptions in text*
  + *Sequence diagrams*
    - *See for examples:* [*https://www.uml-diagrams.org/sequence-diagrams.html*](https://www.uml-diagrams.org/sequence-diagrams.html)
* *Functional descriptions*
  + *Flowcharts*
  + *Pseudo code*
  + *Nassi-Shneidermann diagram (not very common)*

# Conclusions

All components were carefully selected based on clear criteria to ensure the device is accurate, efficient, and low power. The electronics, PCB, and software were designed to work smoothly together, using proven tools and reliable frameworks. The ESP32 handles real-time data processing, alerts, and communication with the backend, while the API and apps ensure data is accessible to both users and caregivers.

This proof-of-concept successfully meets all core requirements—heart rate monitoring, fall detection, medication alerts, and user interface. The design is modular, future-proof, and ready for testing and improvements. The project shows strong potential for real-world use in smart healthcare.