Software Requirements Specification

for Embedded Systems Development - E8408 course project:

Hiking Tour Assistant

Group B authors:

Markus Mattsson 102659867

Teemu Rauha 902988

Vilma Väisänen 1025958

Version history:

Revision	Submission date	Reason for changes / comments
0.1	21.2.2025	First draft
1.0	24.3.2025	Final submission. Improvements made according to recommendations by Kumar Dheeraj

Table of Contents

1. Introduction

Hiking is a very popular way to spend free time in Finland, in every season of the year. Being outdoors is a popular leisure for all ages, regardless of fitness level. According to the EU barometer, over 66% of Finns spend time in outdoor activities, on average 182 times per year. [1] Finland offers breathtaking routes and pure nature for everyone, and it has been studied that being outdoors has many benefits for health. Being outdoors is also linked with happiness, and no wonder that Finland has been awarded for the happiest country for six years in a row [1].

With wearable technology, hiking can be tracked and analyzed, providing information about outdoor experiences, which makes hiking safer for tourists and children, and makes the hiking more interesting for unmotivated people, or people who want to be in the nature only for fitness.

Helsinki City council ordered an application from us to nature lovers. This project is designated to cater the needs of the Finnish Hiking Tourism sector. Our team is developing a **Hiking Tour Assistant 1.0 (HTA)**, an application for a smart wristwatch, that would be able to record and display trip statistics, such as step count, distance traveled, and calories burned. These statistics could promote healthy outdoors activities in Finland, encourage people to go outdoors and grow hiking tourism in Finland. According to Visit Finland, Finnish nature attracts tourists as more than 3.5 million tourists visit to Finland's nature parks annually. [2]

This software requirements specification (SRS) document outlines the functional and technical requirements to ensure the development and delivery of a complete software and hardware solution.

1.1 Document Purpose

This SRS is used for providing the basis for our project's software development. As it is mentioned in the IEEE Std 830-1998, an SRS "establish *the basis for agreement between the customers and the suppliers on what the software product is to do*". The document will describe all functions and requirements we have selected for this system. A thorough SRS will help streamline software development, reducing effort in later project stages, so that creating this document thoroughly will help us develop the actual software and reduce the amount of work in the later stages of the project.

This document will also ensure alignment with the goals set by Helsinki City council and the Finnish hiking Tourism sector. This SRS will be maintained and updated regularly to reflect any changes. Updates will be based on the feedback, new developments or modifications in technology. Updates will be clearly marked to the version history to ensure all stakeholder have access to the most current version.

1.2 Product Scope

The product covered by this document is the Hiking Tour Assistant 1.0. The product consists of a smartwatch and a mobile application. Key features of the HTA are to be able to record and display statistics, such as travelled distance, step count and trip route record real-time. These features ensure safety and convenience throughout the hiking. The product is designated to be suitable, accessible and user-friendly for people of all ages. Goals for this product are promoting hiking tourism in Finland, motivating people for outdoors and supporting and tracking their hiking experiences, ensuring a seamless and enjoyable hiking experience. The product is intended for use by hikers and outdoor enthusiasts in Finland, particularly on hiking trails and in natural parks.

The HTA follows a distributed embedded architecture, integrating a wearable smartwatch and a mobile application. The smartwatch module serves as the data collection unit, while the mobile application processes, stores, and displays the hiking session details. Key components of the system are smartwatch module, mobile application, data synchronization and user interface (UI).

The HTA is funded through a combination of government grants from the Finnish Tourism Board and partnerships with local tourism organizations. Key stakeholders include the end users, the ones who use the product, hikers and tourists. Also, the Helsinki City Council, the Finnish Tourism Organization, the development team, and tech partners.

1.3 Document Overview

Rest of the document gives a comprehensive picture of the Hiking Tour Assistant 1.0. This document is organized into the following chapters:

- **Chapter 2** makes a product overview. We present one by one functions of the product and user characteristics. We also discuss the limitations and the external factors affecting to the product.
- **Chapter 3** focuses on the functional and non-functional requirements of the product. Design and implementation of the product, such as installation, distribution and reusability will be discussed in the end of the Chapter 3.
- **Chapter 4** focuses on the verification methods and process

1.4 Definitions, Acronyms and Abbreviations

Accelerometer: A sensor measuring motion, orientation, and acceleration to track activities like steps.

Bluetooth: A wireless communication technology for low-power and short-range applications.

BMA424: Built-in accelerometer by Bosch Sensortec.

FPS: Frames Per Second. A measure of the refresh rate of the watch's user interface.

GPS: Global Positioning System. A satellite-based navigation system providing location and time information.

GUI: Graphical User Interface.

GNSS: Global Navigation Satellite System. Includes systems like GPS, GLONASS, Galileo, etc.

HTA: Hiking Tour Assistant.

HW: Hardware.

IDE: Integrated Development Environment. A software suite used for developing and testing software.

IoT: Internet of Things. A network of physical objects embedded with sensors and software to connect and exchange data.

LCD: Liquid Crystal Display. Volatile memory used in microcontrollers to store variables and data during program execution.

MET: Metabolic Equivalent of Task. A measure expressing the energy cost of physical activities.

SRAM: Static Random Access Memory.

SRS: Software Requirements Specification.

SW: Software.

UI: User Interface.

UN: United Nations.

1.5 References

[1] Forest.fi, "For the sixth time, the happiest nation in the world – Finns are free to roam anyone's forests," [Online]. Available: https://forest.fi/article/for-the-sixth-time-the-happiest-nation-in-the-world-finns-are-free-to-roam-anyones-forests/. [Accessed: 10-Mar-2025].

[2] Finland.fi, "Natural attraction: Finland's national parks draw people to popular peaks and hidden gems," [Online]. Available: https://finland.fi/life-society/natural-attraction-finlands-national-parks-draw-people-to-popular-peaks-and-hidden-gems/. [Accessed: 10-Mar-2025].

[3] ELEC-E8408, "Embedded systems development course material," [Online]. Available: Aalto MyCourses. [Accessed: 10-Mar-2025].

[4] YK Liitto, "Kestävää teollisuutta, innovaatioita ja infrastruktuureja," [Online]. Available: https://www.ykliitto.fi/kestavaa-teollisuutta-innovaatioita-ja-infrastruktuureja. [Accessed: 10-Mar-2025]

2. Product Overview

Hiking Tour Assistant (HTA) is a smartwatch application designed to enhance hiking experience by presenting and analyzing hiking data. The watch gathers necessary inputs from the user, records hiking data with sensors and syncs the information with Arduino for further analysis.

This chapter describes the general factors that affect our product and its requirements. It provides a background for detailed requirements, which are defined in detail in chapter 3.

2.1 Product Perspective

The HTA is a standalone application designed for integration with the LiLyGo smartwatch (ESP32-based). The application utilizes data from user and step count recorded by the watch, to determine hike distance and calories burned - among other things. Users may start a hiking session by pressing the side button of the watch and observe the hiking data live on the touch screen. Users may further interact with the application via the touch screen.

After a hike, the user receives a more detailed analysis of the session. The analysis is done with Arduino Mega, which downloads hiking- and user data from the watch via Bluetooth and performs necessary calculations before presenting the hiking data on an LCD screen.

The connections between each component and user are presented in the context diagram in the figure 1 below. Each function of the diagram is introduced further in the section *2.2*.

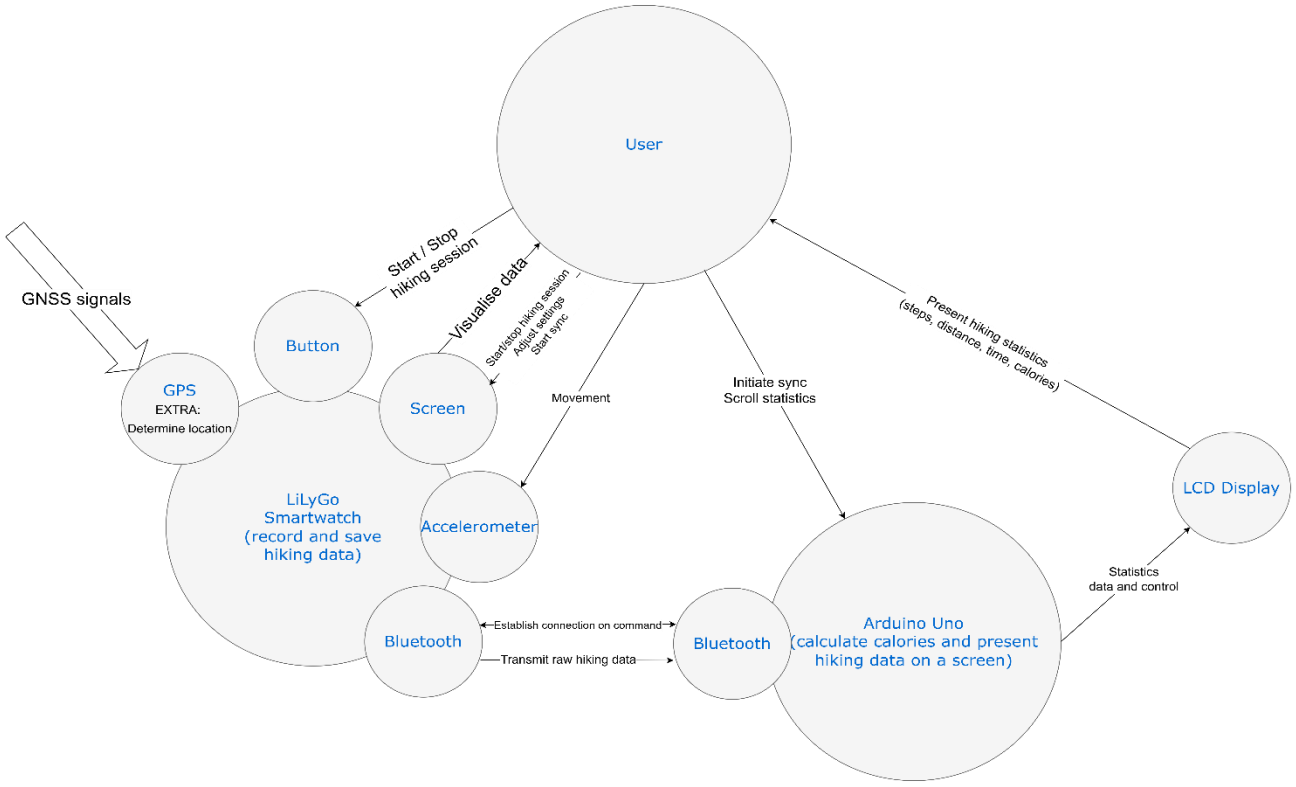

Figure 1: Context diagram

2.2 Product Functions

This section summarizes the major functions of the product. Functions are presented on a high level with short description of the action and how it is initiated (either by the devices or the user).

2.2.1 Watch: Start and stop hiking session

Watch shall have method of starting and stopping the hiking session by the user. This may be done via the watch side button. Watch shall save the total duration of the hike at the end of the hike based on the time elapsed between start and stop signals.

2.2.2 Watch: Record steps count and convert into travelled distance

Once a hiking session is started, the watch shall continuously monitor and count the travelled distance, by counting the wearers steps using an integrated accelerometer. Distance shall be calculated by multiplying the steps count with stride-length. This length may be configurable by the user and may be based on the user height.

2.2.3 Watch: Display steps and distance data on screen

The total step count and distance travelled during the hike (calculated by the 2.2.2 function) shall be presented on the screen during the hiking session. Watch may also display the time elapsed since the session start.

Information shall be presented in English and in numerical format with corresponding units. The screen background, brightness, contrast and text size, font and color shall be selected so that they maximize the information readability and allow optimal accessibility in both dark and bright environment. Reading distance from the screen is 30 cm.

2.2.4 Watch - Arduino: Synchronize and store data with Arduino via Bluetooth

Watch shall have the ability to connect to Arduino via Bluetooth and upload (synchronize) the collected hike session data (steps, distance, duration, user information and possibly route). This feature may be initiated at the end of the hike session, or it may be started by a separate command by the user (via button or touch screen).

Arduino shall have the capability to respond to the sync call by the watch, receive the collected data and store it in a memory for later use.

2.2.5 Arduino: Calculate estimated number of calories burned during the session

Arduino shall use the data received in the sync (2.2.4) to calculate the estimated number of calories burned during the hiking session. Approximate of the user weight (W) in kilograms and hike duration (T) in hours are also needed for the calculation: $\text{Calories} = T \times MET \times 3.5 \times W / (200 \times 60)$. MET is the metabolic equivalent, which is 6 for hiking.

In case there is elevation data available, the calories-algorithm may include it to present more precise estimate of the total burned calories.

2.2.6 Arduino: Show last session statistics on LED screen

Arduino shall display the session statistics on the screen: steps, distance and calories. This may be shown immediately after the sync, or it may be initiated by user via button press.

2.2.7 Watch - Arduino: BONUS: Establish automatic discovery and synchronization via Bluetooth

While not recording a hiking session, the watch may continuously search for the Arduino and attempt to connect to it via Bluetooth. If it is in range of the Arduino, the watch shall establish a connection and perform sync as presented in 2.2.4.

2.2.8 Watch - Arduino: BONUS: Establish communication over Wi-Fi

Watch may have the ability to connect to the Arduino via Wi-Fi. If enabled, this connection shall be initiated by the user from watch settings, via touch screen and/or button UI.

2.3 Product Constraints

Here we briefly present the relevant limitations for the product. This includes hardware (HW) and software (SW) constraints, as well as user interface (UI) limitations.

On the HW side, the main constraints are the limited display size of the watch and also its low processing power. Also, using Bluetooth module limits the range of the data sync. There is also limited storage capacity for the hiking data in the watch and Arduino. Battery capacity on the watch limits the duration of hiking sessions.

As for the SW, the application must be developed on C++ language on the ESP32 platform, and the Arduino program must be written on Arduino IDE's C++ variant.

UI of the watch is restricted to the small touch screen and a push button. Arduino has limited UI capabilities: Plugging in the power cable and using a push button. Final hiking data may be presented only on a 2x16 character LCD-screen, which limits the data format and readability of the information

2.4 User Characteristics

Users of the hiking app can be categorized to two groups: casual hikers and fitness users. Casual hikers are interested in recording occasional trip statistics for personal use. They might want to share the information on social media, but they don't keep long-term track on the hiking data.

Fitness-oriented users are more interested in the long-term performance increase and thus want to keep records of their past hikes. They also want to see the burned calories to match their personal dietary/activity goals. Fitness users are more likely to be using all the features of the app, as where the casual hikers might only be interested in distance and duration data.

Users can use the watch during both night and day, and they understand English and metric units. Using the application should require minimal technical expertise to operate and have a simple UI for navigating different functions and data views.

2.5 Assumptions and Dependencies

The watch model is LiLyGo V2 and the Arduino used is the Mega-version. The LCD screen connected to Arduino is Grove-LCD RGB Backlight version 4.0. Bluetooth unit connected to the Arduino is Opencircuit HM-10 v4.0.

This section outlines the key assumptions and dependencies that could impact the requirements of the Hiking Assistant project. These factors include required drivers and libraries (based on the used HW), network and Bluetooth stability assumptions, and HW reliability considerations.

2.5.1 Required Drivers and Libraries

We are assuming that all the libraries and drivers provided are functioning as supposed with the HW. The following SW components are required to support the functionality of the HW:

- LilyGo Smart Watch Drivers: The watch relies on firmware that is compatible with its power unit, Bluetooth module, acceleration sensor, display, and other essential component required by the product.
- Arduino Mega Libraries: The Arduino must support Bluetooth communication, LCD display output, and user input handling.
- Bluetooth Communication Library: Both the LilyGo watch, and the Arduino Mega require appropriate libraries to establish and maintain a stable Bluetooth connection (e.g., serial communications for Arduino and Bluetooth libraries for LilyGo).
- Power Management Libraries: The system may require libraries to optimize battery consumption for prolonged operation.
- Data Processing Libraries: The Arduino requires computational support for step count-based distance calculations and calorie estimation algorithms.

2.5.2 Hardware Reliability Concerns

The reliability of the Hiking Assistant depends on the durability and performance of its components:

- LilyGo Smart Watch: The watch must withstand outdoor conditions, including temperature variations and potential impacts in normal hiking usage.
- Arduino Mega: We assume the sync will be done indoors, so Arduino and related HW must function in normal indoors environment.
- Power Supply: Battery life is a critical factor. The watch should operate efficiently without frequent recharging, at least during a complete hike. Arduino is powered by AC adapter and does not have battery considerations.
- Button Durability: Both the Lilygo watch power button, and the Arduino button must endure repeated use without mechanical failure.
- Screen Readability: The Lilygo watch screen should be visible in varying light conditions, including direct sunlight and nighttime usage.

2.5.3 Network and Bluetooth Stability Assumptions

The system relies on stable communication between the LilyGo watch and Arduino via Bluetooth:

- Bluetooth Range: The devices assume a stable connection within a reasonable range. In our case the required range in open space is at least 5 meters.
- Interference Considerations: Bluetooth signal integrity may be affected by environmental factors such as physical obstacles and electromagnetic interference.
- Automatic Reconnection: If the connection is lost, the system should be able to attempt automatic reconnection at least once before requiring manual reactivation.
- Data Integrity: Data synchronization assumes that the Bluetooth transmission will be error-free or that any transmission errors can be handled by retry mechanisms.

We assume that all functions not programmed or developed by the project team function well and are stable during normal usage and do not need further improvement work from the project team.

Any changes or failures related to the above assumptions may impact the system's ability to function as specified. Future design iterations should consider mitigation strategies for potential risks associated with hardware and network dependencies.

3. Requirements

This chapter presents the detailed requirements for interfaces, functions, quality, design and implementation.

3.1 External Interfaces

The Hiking Tour Assistant (HTA) integrates various external interfaces to ensure effective communication between system components and users. This section outlines the interfaces that enable interaction with the HTA, supporting its functionality and user experience.

The below data flow diagram illustrates the data exchange between the HTA's primary components; Arduino Mega and LiLyGo watch. It depicts how data is collected, processed, and displayed, emphasizing the system's operational flow and data management.

The subsequent subsections will elaborate on the user, hardware, and software interfaces, detailing their roles and requirements within the HTA system.

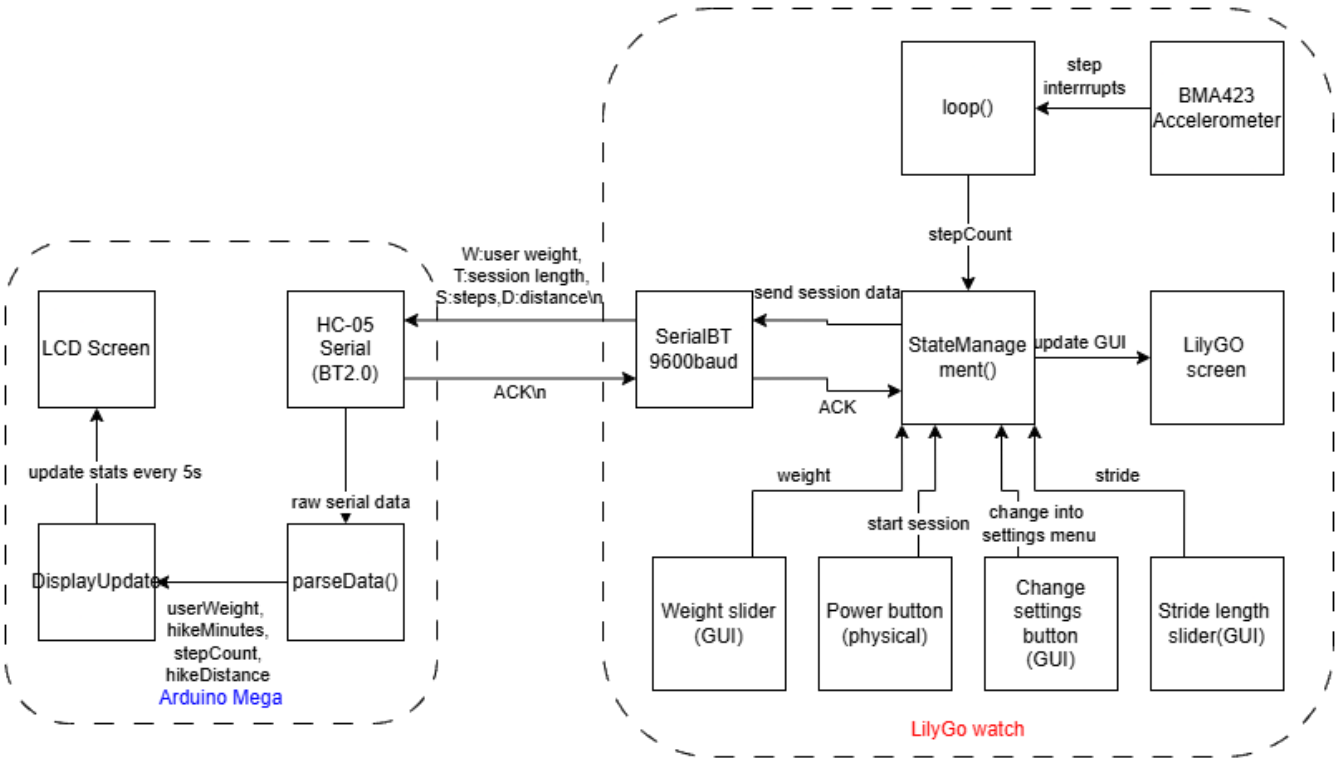

Figure 2: Data flow diagram

3.1.1 User interfaces

The watch has 1,54-inch screen that shall be used for displaying statistics during an active session. Screen is also touch-sensitive, which shall be used for inputting user parameters for more accurate measurements. The button on the side of the watch shall be used for starting and stopping tracking sessions.

Watch also has an internal accelerometer, which detects user arm movement during hike and records steps count. This step count data is then turned into total hike distance based on the user set stride length.

Arduino is used for showing the hike statistics on an attached LCD display. User turns the power on for the Arduino, which then automatically starts polling for hiking data from the watch. After successful sync, the User may scroll different hiking statistics on the display by pressing the Arduino button.

Table 1. User Interface requirements

Component	Inputs	Outputs	Interaction Method	Performance Constraints
LiLyGo Smartwatch UI	Button press, Touchscreen taps	1,54-inch screen showing the GUI	Button & touchscreen	Max UI update delay: 200ms
Watch Accelerometer (BMA423)	User moving the watch	Step count data	Walking, jogging, running, etc	Step count accuracy ±2%
Arduino UI	Power on, button press	LCD screen displays session statistics	Turning power on & button press	Response time < 50ms

3.1.2 Hardware interfaces

The product has two main hardware interfaces: ESP32 inside the LiLyGo v2 -watch, and an Arduino Mega. The ESP32 shall handle all of the user inputs and data gathering from the sensors. The Arduino Mega is only used for data visualization and storing previous sessions.

ESP32-chip inside the smartwatch will handle most of the functionality and input. Connected to the ESP32 is the accelerometer- and GPS-sensors which are used to compute the step count. The smartwatch communicates with an Arduino Mega via Bluetooth and optionally WiFi.

The Arduino Mega communicates with the screen via I2C, and it shall save session statistics into EEPROM-memory.

Table 2. Hardware interface requirements

Component	Inputs	Outputs	Timing Constraints	Accuracy Metrics
LilyGo Smartwatch (ESP32)	Button press, Accelerometer (BMA423), touch screen	BT data to Arduino, Display updates	Max response delay: 200ms	Step count: ±2% accuracy
Arduino Mega	Bluetooth data from smartwatch, buttons	LCD updates, EEPROM storage	Sync under 5 sec	N/A
LCD Screen (16x2, I2C)	I2C data from Arduino	Displays session data	Max update delay: 50ms	Readability at 30cm distance

3.1.3 Software interfaces

The programming interface used for the project is Arduino IDE 2.3.4, and the codebase is a mix of C and C++. The main libraries used are TTGO_TWatch_Library V1.4.3, esp32 2.0.5, BluetoothSerial and Grove- LCD RGB Backlight.

Data will be stored in EEPROM in the Intel hex-file format.

Table 3. Software interface requirements

Module	Inputs	Outputs	Data Format	Performance Constraints
Watch Firmware (ESP32)	Button press, Accelerometer	BT sync data, UI display updates	JSON-like BT payload	Max UI update delay: 200ms
Arduino Firmware	BT session data	LCD output, EEPROM storage	EEPROM-stored stats	Sync within 5 sec
LCD Module	I2C data from Arduino	Displayed text	16x2 character format	Response time < 50ms

3.2 Functional Requirements

As a user starts using the product, they should first input their stats into the watch screen, which will be saved for later calculations.

The watch should stay in idle state until a button is pressed which starts the session. Once a session is started, the software records the step count by listening to interrupts from the BMA423-sensor. Distance is then calculated by multiplying steps by stride-length, which is inputted by the user during setup and the watch also updates the screen with current statistics after every interrupt.

The session is ended by pressing the button on the watch again. After the session ends, the watch starts to synchronize with the Arduino via Bluetooth. If the synchronization is unsuccessful, the watch will attempt 3 more times with 2 second intervals.

After a Bluetooth-connection is established, the watch sends the data from the previous session. To ensure that no data is lost, Bluetooth has built-in error detection. The Arduino then calculates the estimated calorie consumption from the data. After processing, the session statistics are then displayed on the separate LCD-screen and stored in SRAM.

3.2.1 Watch Requirements (FR-1-XX)

FR-1-01: When turned off, after user presses the power button for 3 s (seconds), the watch shall turn on and start the initial program.

FR-1-02: Watch shall show the program standby view after turning on and keep the view on the screen until other required function is initiated.

FR-1-03: The standby view shall state the application name - “Hiking Assistant” and also present a touch-screen button for accessing the settings.

FR-1-04: While in the standby view, pressing the watch power button shall start the hiking session.

FR-1-05: During an active hiking session, the watch shall count the steps taken by the user by listening to the interrupts from the BMA423 acceleration sensor.

FR-1-06: During an active hiking session the watch shall calculate distance traveled by multiplying the stride length with the steps count.

FR-1-07: Watch shall display steps count, distance, and time elapsed during an active hiking session. This session statistics shall be updated after each interrupt from the acceleration sensor.

FR-1-08: During an active hiking session, the watch shall allow the user to stop the hiking session by pressing the power button.

FR-1-09: The watch shall save the hiking data to EEPROM memory after the hike has been stopped.

FR-1-10: After stopping the hiking session, the watch shall attempt to synchronize with the Arduino via Bluetooth. If the synchronization is unsuccessful, the watch will attempt 3 more times with 2 second intervals. If unsuccessful after re-attempts, the watch shall notify the user with a text or icon on the screen.

FR-1-11: Watch shall allow user configuration of stride length based on height in the settings.

FR-1-12: Watch shall display information in a readable format, ensuring optimal visibility in both dark and bright conditions.

FR-1-13: Watch shall allow the user to enable/disable automatic Bluetooth discovery for synchronization with Arduino.

FR-1-14: Watch shall have an option for manually initiating the Bluetooth synchronization.

FR-1-15: When turned on, the watch shall turn off after user presses the power button for 3 seconds.

3.2.2 Watch-Arduino Sync Requirements (FR-2-XX)

FR-2-01: Watch shall initiate synchronization with Arduino at the end of a hiking session or upon user request.

FR-2-02: Watch shall transmit steps, distance, duration, and user information to Arduino via Bluetooth.

FR-2-03: Arduino shall acknowledge data receipt and confirm synchronization status to the watch.

FR-2-04: Watch shall notify the user whether synchronization was successful or failed.

FR-2-05: Watch may attempt automatic synchronization when in range of the Arduino, if enabled.

FR-2-06: Watch may establish a Wi-Fi connection with the Arduino if enabled from settings.

3.2.3 Arduino Requirements (FR-3-XX)

FR-3-01: Once turned on by attaching the power cable, the Arduino shall begin to poll the Bluetooth sensor HM-10 for incoming transmissions.

FR-3-02: Arduino shall parse the received hiking session data into separate variables and store them into SRAM memory.

FR-3-03: Arduino shall calculate estimated calories burned using the stored hiking variables.

FR-3-04: Arduino shall adjust the calorie calculation formula if elevation data is available.

FR-3-05: Arduino shall display session statistics (steps, distance, duration, and calories) on the LED screen. All statistics may not be shown at the same time, and user shall have a way to scroll between different statistics.

3.3 Quality of Service

As nothing about the operation of the software is safety-critical, the main requirements are centred around performance, ease-of-use and data privacy.

3.3.1 Performance

Here are performance requirements for our product under various circumstances as specific as possible. Requirements are needed for the smartwatch to ensure smooth use and reliability in all Finnish seasons.

Table 4. Performance requirements

Requirement	Value
Step count accuracy	**±2%**, comparable to commercial smartwatches.
UI performance	Watch screen refresh rate at least 30 FPS. Maximum response time for user inputs (touch interactions) should be 200ms.
Battery	At least 12 hours continuous use.
Data Synchronization	BT synchronization between smartwatch and Arduino should be done in less than 5 seconds after initialization.
Environmental conditions	Temperature range to function in Finnish nature, -20°C to + 40°C.

3.3.2 Security

We must apply **GDPR (General Data Protection Regulation)** because the software has GPS location data and user profiles. The user has a right to request data deletion at any time and enable to be anonymous. GPS data should be handled with care.

The system must comply with **EU cybersecurity standards for IoT devices**.

Bluetooth connections should use secure pairing.

3.3.3 Availability

Factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

Table 5. Availability requirements for Hiking Tour Assistant 1.0

Checkpoint	**Auto-save data (*Step count, distance traveled, calories burned, GPS*) every 30 seconds to prevent data loss.**
Recovery	Resume from the last saved location and trip statistics after shutdown
Restart	Automatic restart after shutdown

3.5 Design and Implementation

3.5.1 Installation

For the first time installation and set-up, guided tutorial for pairing watch with the mobile application. User needs to accept permissions, such as use of Bluetooth.

The software is uploaded to the watch and Arduino Mega via the Arduino IDE upload-function while the devices are connected to a computer with USB-cables.

3.5.2 Maintainability

For the maintainability, the software shall be clearly divided into modules based on different functionalities, and the relationship between modules shall be as clear as possible and well documented. The libraries used by each module should be kept at a minimum for easy updatability.

3.5.3 Reusability

Documentation: Comprehensive documentation ensures that components can be easily adapted or extended for new projects.

The design of the software for maintainability also enhances reusability. Additionally, the hardware and libraries used have a large number of functionalities that aren't used in the product but could be used in the future to expand the product's capabilities.

As we make the device reusable and design it carefully, we are participating in the **UN Sustainable Development goals. [4]**

Figure 3. UN Sustainable Goal 9.

Goal 9 promotes sustainable industrialization and innovation includes designing technology that is maintainable, adaptable and reusable. Our software and hardware reusability support this goal by ensuring efficient use of resources and long-term innovation.

Figure 4. UN Sustainable Goal 12.

Goal 12 promotes responsible consumption and production. Our product aligns with responsible resource use as efficient software design and reusable hardware components reduce waste and extend the lifecycle of technology.

3.5.4 Portability

As the software is built on libraries specifically for the LilyGo T-Watch 2020, porting it to other systems would require rewriting most of the codebase. The Arduino Mega used for displaying information to the user could be replaced with minimal software changes, as Bluetooth is the only interface between the two.

3.5.5 Cost

Below is a cost estimation, based on a comparable products and sources. The estimate will vary due to a required testing, development, team size and other resources.

Table 6. Specify monetary cost of the software product

Cost category	Estimated cost	Includes
Development	50 000-100 000 € *(Typical Software development costs, source: Deloitte)*	Coding, testing, design
Deployment	50 000 € *(Typical deployment costs, source: AWS pricing)*	Cloud system
Maintenance	10 000 € / year *(Typical maintenance costs, source: GoodFirms)*	Customer support, updates, bug fixes
Licensing	10 000 € *(source: GDPR.eu & ISO.org)*	GDPR, ISO standards
Marketing	10 000 € *(source: Skift.com)*	Tourism specific promotion
Total:	130 000 € - 180 000 €	

3.5.6 Deadline

21.2.2025 Deadline for part A – requirement engineering (SRS)

24.3.2025 Deadline for part B – final document + SRS updated

28.3.2025 Demo & Presentation Day (10 min presentation)

4. Verification

This chapter presents the verification approaches and methods planned to qualify our HTA software. The purpose of the verification process is to provide objective evidence that our system fulfills its requirements and characteristics to ensure high-quality product for customer (Helsinki City Council).

4.1. Verification process

Verification process consists of multiple stages. On the first stage, we identify the functional and nonfunctional requirements, from the chapter 3. Next step is to create test cases for each requirement. Verification is conducted through unit testing, integration testing, and system testing. Unit testing ensures that individual **software** components function correctly. Integration testing verifies that different **modules communicate and work together as expected**. Finally, system testing evaluates the **entire system's** functionality, performance, and reliability in real-world scenarios.

If the system passes all tests, it is considered verified. If any issues arise, they are addressed, documented and the affected components are retested to ensure compliance with the requirements. In the figure 4. this verification process is demonstrated.

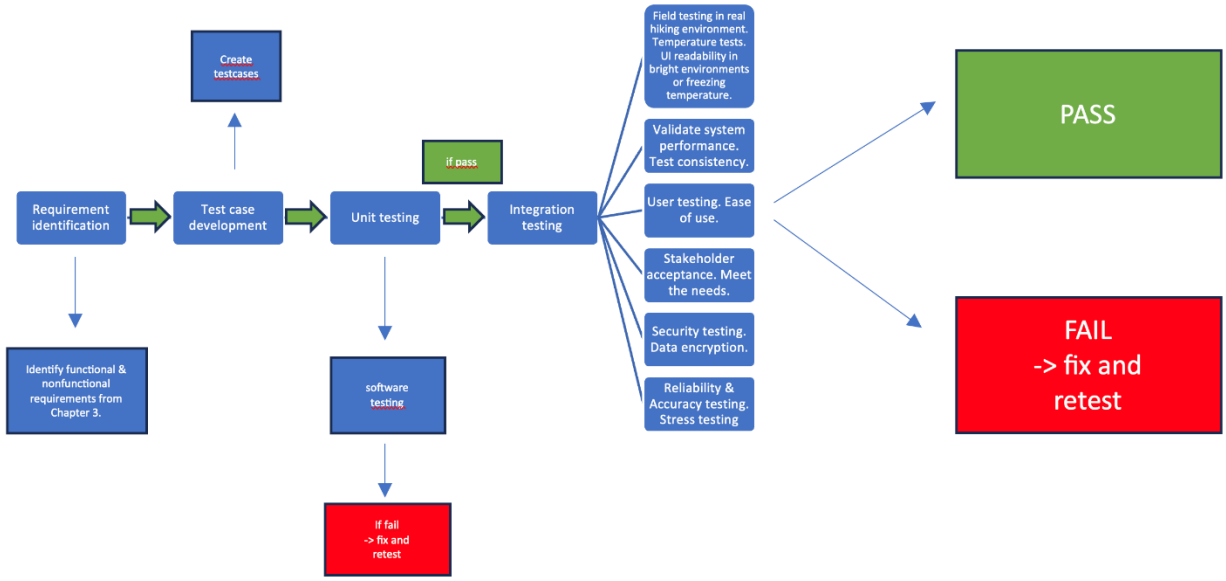

Figure 5. Process flowchart

4.2. Verification methods

The verification of the requirements outlined in chapter 3 is essential to ensure the correct functioning of the system. For each functional requirement, corresponding test cases have been

defined to verify the expected behavior of the system components, both individually (unit testing) and in integration (system testing). These test cases are listed in the table 7 below.

Table 7. Test cases

Test case ID	Description
TC-1	Verify that the watch turns on when the power button is pressed for 3 seconds and starts the initial program.
TC-2	Verify that after the watch turns on, the standby view is shown with the application name and a touchscreen button for accessing settings.
TC-3	Verify that in the standby view, pressing the power button for 1 second starts the hiking session and pressing it for 1 second again stops the session.
TC-4	Verify that during an active hiking session, the watch counts steps using the BMA423 sensor and calculates distance using stride length and step count.
TC-5	Verify that the watch displays steps, distance, and time elapsed during an active hiking session, updating with each interrupt from the acceleration sensor.
TC-6	Verify that after stopping the hiking session, the watch saves the data and attempts to synchronize with Arduino up to 3 times, notifying the user if unsuccessful.
TC-7	Verify that the watch allows the user to configure stride length based on height in the settings
TC-8	Verify that the watch displays information in a readable format, ensuring visibility in both dark and bright conditions.
TC-9	Verify that the watch allows the user to enable/disable automatic Bluetooth discovery and manually initiate Bluetooth synchronization.
TC-10	Verify that the watch initiates synchronization with Arduino at the end of a hiking session, transmits session data, and notifies the user whether synchronization was successful or failed.

To facilitate this process, the following test case mapping table has been created. This table associates each test case with the specific requirements from both the watch and Arduino components, ensuring comprehensive coverage of all functional requirements. The figure 5 below highlights the relationship between the test cases and the associated requirements to be verified during the testing phase.

Requirement ID (Chapter 3)	Requirement Description	Test Type	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7	TC-8	TC-9	TC-10
FR-1-01	Watch turns on after holding power button for 3s	Unit	X									
FR-1-02	Watch displays standby view after turning on	Unit		X								
FR-1-03	Standby view displays app name and settings button	Unit			X							
FR-1-04	Pressing power button for 1s starts hiking session	Unit				X						
FR-1-05	Watch counts steps using BMA423 sensor	Unit					X					
FR-1-06	Watch calculates distance using stride length and step count	Unit					X					
FR-1-07	Watch displays updated step count, distance, and time	Integration						X				
FR-1-08	User can configure stride length in settings	Integration							X			
FR-1-09	Display remains readable in all lighting conditions	System								X		
FR-1-10	Watch syncs with Arduino via Bluetooth	Integration								X		
FR-1-11	Holding power button for 3s turns watch off	Unit									X	
FR-1-12	Watch saves hiking data to memory after stopping session	System									X	
FR-1-13	Watch attempts BLE sync 3 times if unsuccessful	Integration										X
FR-1-14	Watch notifies user if sync fails	System										X
FR-1-15	User can manually initiate BLE sync	Integration										X
FR-2-01	Watch initiates sync at the end of a session	Integration							X			
FR-2-02	Watch transmits session data via Bluetooth	Integration								X		
FR-2-03	Arduino acknowledges received data	Integration									X	
FR-2-04	Watch notifies user about sync status	System									X	
FR-2-05	Watch syncs automatically when in range	System										X
FR-2-06	Watch can connect to Arduino via Bluetooth	Integration										X
FR-3-01	Arduino listens for incoming Bluetooth data	Unit										X
FR-3-02	Arduino parses received data and stores it	Integration								X		
FR-3-03	Arduino calculates calories burned	Unit									X	
FR-3-04	Arduino adjusts calorie formula if elevation data is available	Integration									X	
FR-3-05	Arduino displays session statistics on LED screen	System										X

Figure 6. Test case mapping table

5. Conclusion and Acknowledgments

We would like to extend our thanks to all reviewers and course staff who have provided valuable input and support during the writing of this Software Requirements Specification (SRS). We welcome any additional feedback or suggestions you may have regarding this SRS. Your guidance is instrumental in helping us learn and improve our skills in crafting high-quality SRS documents.

We look forward to continuing our collaboration and successfully bringing the Hiking Tour Assistant to life in the demo session. Thank you for your support and for providing such an enlightening course.

Best regards,

The Hiking Tour Assistant Development Team