2023

Hiking Tour Assistant Software Requirements Specification

ELEC-E8408 EMBEDDED SYSTEMS DEVELOPMENT

GROUP K: PYRY AHO, VILLE SYNKKÄNEN, SEPPO KOIVISTO

VERSION	DATE
1.0	28.02.2023
2.0	21.03.2023

Contents

1. Introduction 2
1.1 Purpose 2
1.2 Scope 2
1.3 Definitions, acronyms, and abbreviations 2
1.4 References 3
1.5 Overview 3
2. Overall description 3
2.1 Product perspective 3
2.2 Product functions 6
2.2.1 Context diagram .... **Virhe. Kirjanmerkkiä ei ole määritetty.**
2.3 User characteristics 7
2.4 Constraints 7
2.5 Assumptions and dependencies 7
3. Specific requirements 8
3.1 External interface requirements 8
3.1.1 User interfaces 8
3.1.2 Hardware interfaces 9
3.1.3 Software interfaces 10
3.1.4 Communications interfaces 11
3.2 Functional requirements 11
3.3 Performance requirements 13
3.4 Design constraints 14
3.4.1 Standards compliance 14
3.5 Software system attributes 14
3.5.1 Reliability 14
3.5.2 Availability 14
3.5.3 Security 14
3.5.4 Maintainability 14
3.5.5 Portability 15

1. Introduction

The Hiking Tour Assistant (HTA) is a software application designed to assist hikers in recording their hiking tours. This Software Requirements Specification for the Hiking Tour Assistant outlines the functional and non-functional requirements of the software. It describes the features and capabilities of the software, as well as the technical constraints and performance requirements.

1.1 Purpose

The purpose of this document is to describe the purpose and functions of the Hiking Tour Assistant and to lay out the requirements in as much detail as possible. This document also serves as a plan for the project and can be used to check requirements related to different aspects of the product.

The intended audience for this SRS is first and foremost the assistants and other students who will evaluate the mini project. As a result, this document is relatively technical and assumes that the reader has some background in software development and/or requirements engineering.

1.2 Scope

Hiking is a popular activity in Finland. The hiking experience can be improved by tracking the activity by using a smartwatch. The purpose of the Hiking Tour Assistant is to help the user to track their hiking session statistics including travelled distance, step count, and burned calories. The smartwatch application has a simple and intuitive UI which allows the user to start recording their session by a single push of a button.

The software implements a smartwatch UI which allows the user to start and stop recording sessions. During a hiking session travelled distance and step count are displayed and saved to the smartwatch. In addition, a web UI for a Raspberry Pi is implemented to display session statistics, including burned calories. The session information can be synchronized between the smartwatch and the Raspberry Pi.

The project is funded by Helsinki City Council, who are interested in developing an app for hikers and nature lovers. The other main stakeholders of the project are us as the development team, the Finnish Hiking Tourism and Finnish Fitness consortiums as the expected customers for the product, and the individual end-users of the product.

The end-users can be expected to be interested in hiking and they probably hike quite a lot if they have an interest in tracking their session statistics. Additionally, the users cannot be expected to be technologically savvy, so the application must be intuitive and easy to use. The application is tailored towards recording personal hiking trips, but the user can also choose to use it to track their daily steps in their everyday life. So, the possibilities of the application are not strictly limited to hiking.

1.3 Definitions, acronyms, and abbreviations

HTA	Hiking Tour Assistant
RPi	Raspberry Pi
SRS	Software Requirements Specification
C++	Programming language
Python	Programming language
Flask	Python framework for web servers

Pybluez	Python library used for Bluetooth connections
Bluetooth	A wireless technology
API	Application Programming Interface
UI	User Interface
ESP32	Series of microcontrollers
OS	Operating system
SRAM	Static Random Access Memory
PSRAM	Pseudostatic Random Access Memory
IMU	Inertial Measurement Unit
LED	Light-emitting Diode
IEEE	Institute of Electrical and Electronics Engineers

1.4 References

LiLyGo TWatch documentation

RPi documentation

IEEE-830 Standard

1.5 Overview

The rest of the SRS contains the overall description of the product, including description of the functions, operation, intended users and potential constraints and assumptions and definitions of all the specific requirements. A context diagram outlining the interactions between external entities and the software system is also provided.

Section 2 contains the overall description of the HTA, consisting of five subsections: product perspective product functions and the context diagram, user characteristics, constraints, and assumptions and dependencies. Section 3 offers a more detailed explanation of the specific requirements. The section is divided into external interface requirements, user interfaces, functional requirements, performance requirements, design constraints, and software system attributes.

2. Overall description

2.1 Product perspective

The Hiking Tour Assistant is meant to serve as a simple solution for logging hiking statistics using a smartwatch. HTA allows the user to track travelled distance and step count during the session and to move these statistics to an external device (Raspberry Pi) where extended statistics including estimated calories burned can be displayed.

The project can be easily separated into two products: the software for recording the session statistics on the smartwatch and a web UI implemented on the Raspberry Pi to display the session data. These two products communicate with each other over a Bluetooth connection.

System interfaces:

Figure 1 shows the deployment diagram of the system, where the smartwatch and RPi are shown to communicate over a Bluetooth connection. The program for the smartwatch consists of *main.cpp* file, where the main functionality of the watch is implemented, *config.h* and *utils.h* header files, which are used to import certain functions to the main file, and *id.txt, steps.txt* and *distance.txt* text files, where the session statistics are saved after the session and read from when sending them to the RPi in the *main.cpp* program.

The RPi program consists of two main scripts: *receiver.py* and *app.py*, which are used to synchronize data with the watch and launch the webUI respectively. Files *bt.py* and *hike.py* contain class declarations for the classes used in the receiver script. The receiver writes the session data it receives from the watch to the *session.txt* file and *app.py* reads the data from the file to display the session statistics to the user. The app also reads/writes the users weight from the *user.txt* file.

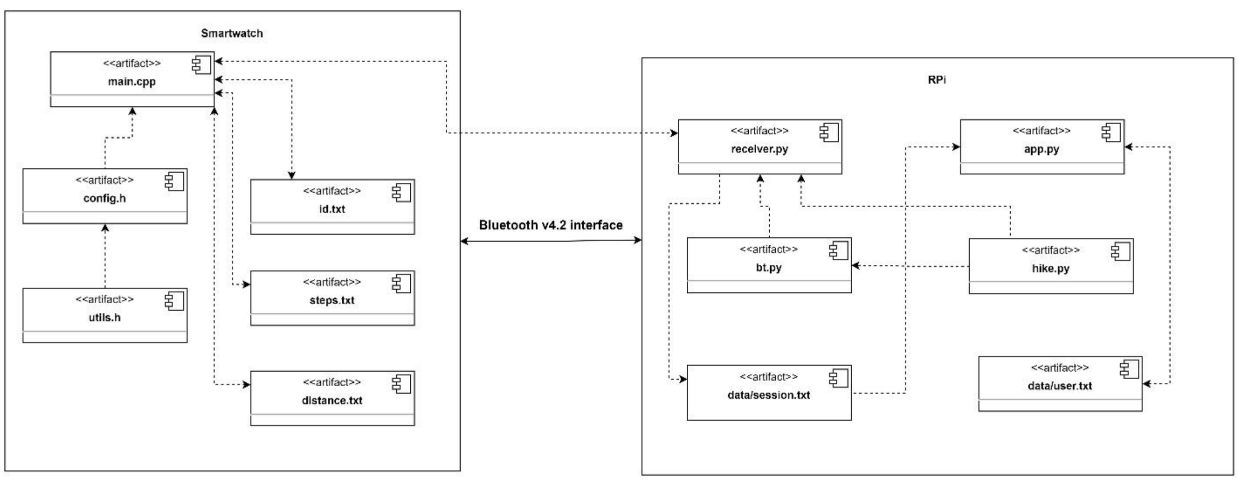

Figure 1: Deployment diagram of the HTA.

User interfaces:

The smartwatch UI shall be intuitive and easy to use. It allows the users to start & stop hiking sessions and input their step length. The current session statistics are displayed during a hiking session. The user can also access the statistics of the last recorded session. The UI shall also have an option to synchronize the data with the RPi on user command.

The web UI on the RPi shall display the extended session statistics on the screen. In addition to travelled distance and step count, the extended statistics include an approximation of the burned calories during the hike which is calculated on the RPi. To calculate the estimated burned calories, the UI should ask the user to input their weight.

Hardware interfaces:

The hardware interfaces on the smartwatch include the screen and buttons which the user can interact with. The hardware interfaces on the RPi include the peripherals such as the mouse, keyboard, and monitor.

Software interfaces:

The two main software interfaces are the LilyGo smartwatch firmware and the Raspberry Pi operating system. On top of the RPi OS, Python and several libraries are used, along with the web server.

Communications interfaces:

The smartwatch should communicate with the RPi via Bluetooth using the ESP32 platform. ESP32 supports Bluetooth v4.2.

Memory constraints:

The Raspberry Pi has 1 GB of LPDDR2 memory and 32 GB of flash storage. This storage and memory are enough to run the RPi operating system and the web server and synchronization app on top of it. The limited amount of RAM could be an issue if there are many browser tabs open at the same time, though this is not necessary for our system.

The LilyGo watch has 520 kB of SRAM, 8 MB of PSRAM and 16 MB of flash storage. This should be more than enough to run the smartwatch program while fulfilling the requirements laid out in this document. As only the latest session statistics are stored, the smartwatch storage capacities should not pose any issues either.

Operations:

The smartwatch has four main modes of operation: idle, recording, stopped, and synchronizing with RPi. The RPi has four main modes of operation: idle, synchronizing with the smartwatch, calculating the estimated burned calories, and displaying the data.

2.1.1 Context diagram

Figure 2 shows the context diagram for the Hiking Tour Assistant (HTA). The diagram shows the basic relationships between the HTA and the main hardware interfaces which interact with it i.e., the smartwatch screen, the smartwatch controls and the RPi peripherals. The smartwatch controls are used as inputs for the system, and they can indicate either starting/ending a session, synchronizing the data of the last session, or inputting the users step length to calculate the travelled distance. The display shows the user the relevant information, for example steps and distance travelled during the session. The RPi accepts synchronizing the latest session data from the smartwatch over a Bluetooth connection. After receiving the data, the user’s weight is requested and estimated burned calories are calculated. Next, the extended statistics are displayed on the external monitor connected to the RPi.

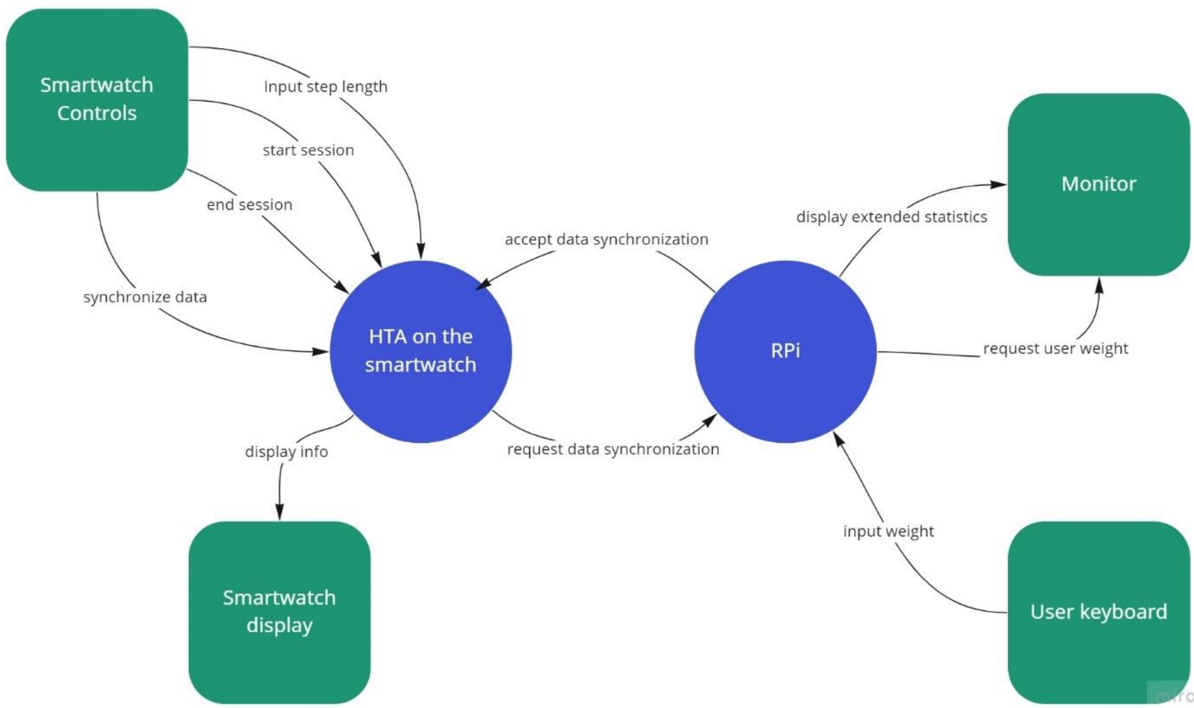

Figure 2: The context diagram of the Hiking Tour Assistant.

2.2 Product functions

The main product functions are listed below. These are explained in greater detail in section 3.2 Functional requirements.

- Start & stop hiking sessions. The user can start/stop the session by pressing the button on the side of the smartwatch.
- Record step count and convert it into travelled distance during the hiking session. The user can input their step length before starting the session, otherwise the default value of 0.8 m is used. Recording the steps and distance is done in the smartwatch HTA module which is shown in the context diagram.
- The session data is displayed on the smartwatch screen during the session.
- Synchronize and store the data with RPi via Bluetooth. The smartwatch tries to synchronize to the RPi whenever it has a session saved and it is in an otherwise idle state. The RPi accepts the synchronization after user input (running the synchronization script).
- Calculate the estimated number of calories burned during the session on RPi. The RPi requests the user to input their weight via the keyboard. After receiving the weight, the RPi calculates calories burned during the session.
- Initialize the web UI and show statistics from the last session (travelled distance, step count and burned calories). The web UI is displayed on the external monitor.

2.3 User characteristics

The expected users are normal people who do not necessarily have technical expertise or higher education. The watch should be easily usable and not require more than basic knowledge of such devices.

2.4 Constraints

Regulatory policies: The software shall follow the data collection laws in Finland.

Hardware limitations: The main limitations of the hardware are the memory capacity and processing power of the LilyGo smartwatch. The limited processing power means that more complex data processing, such as calculating the burned calories, shall not be done on the smartwatch but instead calculated on the RPi.

The LilyGo watch also has an 240x240 pixel display which limits the amount text that can be displayed on the screen. The RPi requires an external monitor to display the web UI. The watch also has a BMA423 IMU sensor which is used to implement the step-counter functionality. It can measure acceleration in 3 axes at 12 bit digital resolution which is plenty for our application.

Data processing: During the session, steps are written into a 32 bit unsigned integer variable which has the maximum value of 4,294,967,295 which is more than plenty for recording the steps. The distance to be displayed is calculated by multiplying the steps with the float type step length variable. The steps and travelled distance, alongside the session id, are written into .txt-files. The data is written only at the end of the session as not to disturb/block the step counting process during the session.

Interfaces to other applications: Other than the interfaces between the smartwatch and RPi, there are no interfaces to external applications or devices.

Parallel operation: The smartwatch and RPi operate independently, except when they are synchronizing. There are no other parallel operation constraints.

Higher-order language requirements: C++ is used for the development of the smartwatch software.

Signal handshake protocols (e.g., XON-XOFF, ACK-NACK): Bluetooth v4.2 implements the 802.11 RSN 4-way handshake between two devices establishing a connection.

Reliability requirements: The main reliability requirement is that the smartwatch saves the session data into non-volatile flash memory straight after ending the session so that if the watch malfunctions (for example the battery runs out) the session data can be recovered.

Criticality of the application: The application is not critical, and possible failures will only be an inconvenience for the user.

Safety and security considerations: The application is not safety-critical.

2.5 Assumptions and dependencies

It is assumed that the software is developed specifically for the LiLyGo smartwatch and RPi platforms, so only the constraints which come with these platforms need to be accounted for. If the goal platforms change the SRS will need to be updated accordingly.

The LiLyGo TWatch Library will be used to implement the smartwatch software. The library provides many useful ready-made functions such as the BMA423_StepCount.

For the RPi, Python is used as a programming language along with the following libraries: Flask for the web server and PyBluez and Pillow for the synchronization application.

The users of our system are assumed to be normal people that have only basic experience with devices such as ours. This means that the system should be straightforward and easy to use. The user manual should provide adequate instructions for using the watch and RPi.

It is assumed that the libraries and drivers used are fit for this purpose and will work without major errors. The hardware is also assumed to operate as intended without any malfunctions or other issues.

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

Name of item	Session controls	HTA screen	Data synchronization command	Display session data command	Session data display	User step length	User weight
Description of purpose	Start and stop recording sessions	Display hiking session information	Send a data synchronization command	Send a command to the RPi to display the saved session data.	Display the session data saved to the RPi	Determine user step length for distance conversion	Determine user weight for calorie calculation
Source of input	Button on the HTA	N/A	Keyboard connected to the RPi	Mouse connected to the RPi	N/A	Watch touch-screen	Keyboard connected to the RPi
Destination of output	N/A	HTA screen	N/A	N/A	Monitor connected to the RPi	N/A	N/A
Valid range & accuracy	N/A	N/A	N/A	N/A	N/A	0.1 –2.0 meters, 0.05 accuracy	0 – 1000 kg, one decimal accuracy
Units of measure	seconds	Hz, Steps, meters	seconds	ms	steps, kilometers, calories, seconds	meters	kilograms
Timing	Starting or ending a session shall have	The screen shall update at 1 Hz	Synchronization attempt shall start	The command shall execute	Displaying the data shall take at most 5	Saving the user step length shall take	Saving the user weight shall take

	at most a 1 s delay	frequency or faster	within 1 second of starting the script	with at most a 500 ms delay	seconds to complete	no longer than 500 ms	at most 500 ms
Relationships to other inputs/outputs	Session results are visible on the HTA screen	Related to the session controls	This command starts the synchronization process	Session data is displayed after executing this command	Related to the display data command	Step length is asked on the HTA screen	When the data is synchronized, calories are calculated based on the user's weight
Screen formats/ organization	N/A	240x240 pixels	N/A	N/A	1920x1080 pixels	N/A	N/A
Window formats/ organization	N/A	LED matrix with steps and distance in this order	N/A	N/A	Data is displayed in the following order: steps, kilometers, calories	N/A	N/A
Data formats	N/A	Floating point numbers	List of sessions	Text	Floating point numbers	Floating point numbers	Floating point numbers
Command formats	Boolean input	N/A	Boolean input	Boolean input	N/A	N/A	N/A
End messages	Session end message	Session end message	N/A	N/A	N/A	User step length saved message	User weight saved message

3.1.2 Hardware interfaces

Name of item	Smartwatch Button	Smartwatch Screen	RPi Mouse	RPi Keyboard	RPi Display
Description of purpose	Start / Stop hiking session recording	Show session information to the user	Control the Web UI	Control the Web UI	Show Web UI to the user

		Touchscreen used as an input			
Source of input	User hand	User hand	User hand	User hand	N/A
Destination of output	N/A	Smartwatch screen	N/A	N/A	RPi HDMI output
Units of measure	seconds	Hz	ms	ms	Hz
Timing	The button shall react to inputs in less than 1 s.	The screen shall update at a frequency of 1 Hz or higher	The RPi should respond to inputs in 500 ms or less	The RPi should respond to inputs in 500 ms or less	The screen shall update at a frequency of 30 Hz or higher
Relationships to other inputs/outputs	Session statistics are shown on the watch screen	Sessions are started with the smartwatch button	Controls the RPi together with the keyboard	Controls the RPi together with the mouse	Responses to the inputs from the mouse and keyboard are seen on the screen
Screen formats/organization	N/A	240x240 pixels	N/A	N/A	1920x1080 pixels
Window formats/organization	N/A	LED matrix	N/A	N/A	Data is displayed in a list
Data formats	N/A	N/A	Serial data	Serial data	N/A
Command formats	Boolean input	N/A	N/A	N/A	N/A
End messages	N/A	Session end message	N/A	N/A	N/A

3.1.3 Software interfaces

The system shall work together with the smartwatch firmware and RPi operating system.

Name of item	TWatch LilyGo firmware	RPi operating system	Web browser	Python & required libraries
Description of purpose	Interface between the HTA software and hardware	Interface between RPi hardware and software running on the RPi	Running the web server on the RPi	Interface for implementing the web server and data synchronization on the RPi
Source of input	HTA buttons	Mouse, keyboard	RPi operating system	RPi operating system

Destination of output	HTA screen	Monitor	RPi operating system	RPi operating system
Timing	The firmware should react to user inputs in less than 5 seconds	The operating system should react to user inputs in less than 5 seconds.	The web browser should load a page in less than 5 seconds.	N/A
Relationships to other interfaces	The user interfaces of the HTA are directly related to the firmware	The web server and synchronization application run on top of the OS	Runs on top of the RPi OS	Runs on top of the RPi OS
Window formats/organization	LED matrix	Fullscreen/window	Fullscreen/window	Command line
Data formats	Various (.cpp, .h, etc.)	Various	Various (HTML, .txt, etc.)	Various (.py, .txt, etc.)
Command formats	Various	Various	Various	Various

3.1.4 Communications interfaces

Name of item	Bluetooth interface
Description of purpose	Synchronize the data between the HTA and RPi
Source of input	Input data from the HTA
Timing	Synchronization attempt shall take at most 10 s, otherwise it shall time out
Relationships to other inputs/outputs	The data synchronization command starts the synchronization process
Data formats	Floating point numbers
Command formats	N/A
End messages	Synchronization result on the HTA and RPi screens

3.2 Functional requirements

The functional requirements for the system are described through use cases in the table below.

Purpose	The user wants to start a new session	A session is in progress	The user wants to end the session	The user wants to synchronize session data	The user wants to save their weight	The user wants to save their step length	The user wants to see data from the previous session

Input	Side button on the watch	Steps taken by the user	Side button on the watch	Synchronization application started on the RPi	Keyboard connected to the RPi	Touchscreen on the watch	Opening the session data web page
Output	Session started	Recorded data (shown on the HTA screen)	Session ended, session data saved	Session data on the RPi	User weight saved into the RPi	User step size saved into the HTA	Session data displayed
System state	HTA: from idle to recording	HTA: recording	HTA: from recording to stopped and then idle	HTA & RPi: from idle to synchronizing and then idle	RPi: from idle to saving data to idle	HTA: idle	RPi: from idle to displaying
Pre-conditions	A session is not ongoing, the watch is turned on	N/A	A session is in progress	The HTA is on, a session is not ongoing	The web server is on, the user data page is open	A session is not ongoing	The web server is open, the index page is open
Post-conditions	The HTA is recording a session	N/A	A session is not in progress	The session data is saved on the RPi	The user weight is saved into a text file	The user step size is saved into the HTA. It is used in the following sessions.	Session data, including steps, distance and burned calories is displayed on the web page
Basic flow	The user presses a button to start the session and the watch starts recording steps.	Steps taken by the user are tracked and used to calculate the traveled distance. The traveled distance	When the session is ended, the recorded session data is saved into non-volatile memory. The HTA screen	When the synchronization command is given from the RPi, it will try to form a connection with the HTA. If the connection is formed,	The user inputs their weight. The system checks if the input is a number in the correct range. If it is, the	The user inputs their step length on the HTA screen. The watch displays the current step length and touching either the "-" or "+" sign decreases or increases	When the session data web page is opened, the data is retrieved from the RPi memory and the data is displayed in a

		and step count are displayed on the screen as an LED matrix.	shall show a session ended message and then turn off.	it will attempt to synchronize the data. The synchronization attempt can be interrupted with CTRL+C.	input is accepted and saved into the memory. Otherwise, it is rejected and asked again.	the step length by 0.05 meters. The step length range is [0.1m, 2.0m] and touches are ignored if these limits are reached.	readable format. If no data is found, the web server shall show a "No session data" message.
Error handling	No additional error handling	No additional error handling	No additional error handling	The synchronization application on the RPi closes following an error	If the input reading fails, the system asks for input again	No additional error handling	If retrieving data fails, the web server shows a message on the session data page

3.3 Performance requirements

Requirement/ System component	Traffic	Timing	Security
HTA	The watch shall store one recorded hiking trip at a time.	The watch shall update the recorded trip data on LCD every 1000 ms. The synchronization process shall take no more than 5 seconds.	No additional security features implemented.
RPi synchronization program	The RPi shall store one recorded hiking trip at a time, along with the user's weight.	The RPi shall process the recorded hiking data from the watch in 5 seconds or faster.	No additional security features implemented.
RPi web server	The web server is accessed by only one user at a time.	The web page shall be loaded in 5 seconds or faster.	The web server shall be accessed only from the RPi. No additional security measures are necessary.

3.4 Design constraints

1. The RPi and the smartwatch communicate over Bluetooth connection, so if we want to expand support for other devices, they must be Bluetooth compatible.
2. The smartwatch uses an ESP32 processor, so the used libraries must be compatible with ESP32.
3. The LilyGo watch has 520 kB of SRAM, 8 MB of PSRAM and 16 MB of non-volatile memory (flash storage). The program shall not exceed these limitations.
4. The TTGO_TWatch_Library shall be used for developing the smartwatch program. Additionally, the LittleFS_esp32 library is used to implement the saving the session statistics into text files.
5. The smartwatch program shall be designed as a state machine, where the watch switches in between states based on user inputs.
6. The source code for the smartwatch is written in C++ and should follow the best practices for C++ programming.
7. The RPi has 1 GB RAM and 32 GB non-volatile memory (flash drive). These limits shall not be exceeded by the program.
8. The source code for the RPi is written in Python and should follow the best practices for Python programming.
9. The required Python modules for the RPi program are: PyBluez for the Bluetooth synchronization, Flask for web application development, and Pillow for image processing.

3.4.1 Standards compliance

The following standards shall apply to the development of the HTA:

- IEEE-830 std for Software requirements specification
- Bluetooth v4.2 standard

3.5 Software system attributes

3.5.1 Reliability

The HTA shall perform its intended function consistently without errors. If a system error is unavoidable, the system shall be able to recover from it and continue functioning as well as possible.

3.5.2 Availability

The smartwatch should save the last session statistics into non-volatile memory so that it can be viewed in the web UI. The statistics of the previous session can be deleted as a new session is started. In this case, the watch should ask the user to confirm that the data from the earlier session can be deleted.

3.5.3 Security

Only the latest session is saved on the device. The communication between the smartwatch and RPi using Bluetooth is only established when necessary, otherwise Bluetooth should be turned off to minimize security risks. The RPi is not connected to the internet; thus, a potential attacker would need physical access to the RPi to perform any malicious activity, for example steal user data. Therefore, no data encryption is needed.

3.5.4 Maintainability

The code should be modular and commented on as necessary to make it easily readable to increase maintainability. The design report shall describe the system in sufficient detail to make understanding and maintaining the system easy and straightforward.

3.5.5 Portability

Using a commonly used language such as Python and C++ increases the portability of the system. For instance, the web server should also work on a Windows-based PC without any modifications to the code.