Aalto University 24.3.2025

Test Plan for Hiking Tour Assistant 1.0

for Embedded Systems Development – E8408 project: Hiking Tour Assistant

Document version: 0.1

Group B authors:
Markus Mattsson 102659867
Teemu Rauha 902988
Vilma Väisänen 1025958

Version history:

Revision	Submission date	Reason for changes / comments
0.1	24.3.2025	Final submission

Table of Contents

1. Introduction

1.1 Purpose

This Test Plan outlines the strategies, objectives, and scope for testing the Hiking Tour Assistant 1.0 system, which consists of a smartwatch and an Arduino-based unit. The system's functionality includes hiking session tracking, step counting, data synchronization, and user settings configuration. The purpose of this document is to verify that the system meets the functional and non-functional requirements, as well as to identify any defects or issues.

1.2 Scope

The testing will cover unit testing, integration testing, and system testing for the smartwatch and Arduino components. The components will be verified individually (unit testing) and in combination (system integration). The goal is to ensure the system functions correctly, provides accurate results, and meets all specified requirements, including those related to synchronization, user interaction, and data tracking.

1.3 Overview of the System

The Hiking Tour Assistant consists of:

- **Smartwatch**: Tracks steps, distance, time, and hiking session status. It communicates with the Arduino via Bluetooth for synchronization.
- **Arduino**: Receives data from the smartwatch, processes it, and stores it for further calculations, such as calorie estimation, which is displayed on an LED screen.

2. Test Strategy

2.1 Types of Testing

The testing approach will consist of the following phases:

1. **Unit Testing**: Each component (watch and Arduino) will be tested individually for correct behavior of each function.
2. **Integration Testing**: After unit testing, both the smartwatch and Arduino will be tested together to ensure proper synchronization and data flow.
3. **System Testing**: Complete system testing will validate that the integrated system meets the requirements outlined in the Software Requirements Specification (SRS).

2.2 Testing Objectives

- To validate that the watch turns on, enters standby mode, starts a hiking session, counts steps, calculates distance, and synchronizes data with the Arduino.
- To ensure that the Arduino receives data, processes it, and displays session statistics correctly.
- To verify synchronization between the watch and Arduino via Bluetooth and Wi-Fi.
- To test the system's reliability, including edge cases such as failed synchronization and incorrect data entries.

3. Test Approach

3.1 Requirements Verification

The verification process consists of multiple stages:

- **Functional Requirements**: Verifying the behavior of the system as per the specified requirements.
- **Non-Functional Requirements**: Evaluating performance, usability, and reliability of the system.
- **Test Case Mapping**: For each functional requirement, test cases are defined and mapped.

Requirements for the Hiking Tour Assistant:

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 for 1 s 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 for 1 s.
FR-1-09: The watch shall save the hiking data to XXXX 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 BLE. If the synchronization is unsuccessful, the watch will attempt 3 more times with 10 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 BLE synchronization.
FR-1-15: When turned on, the watch shall turn off after user presses the power button for 3 seconds.

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.

Arduino Requirements (FR-3-XX)
FR-3-01: Once turned on by attaching the power cable, the Arduino shall begin to poll the BLE sensor XXXX 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.

For each requirement mentioned above, 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 1 below.

Table 1. 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 BLE 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.

Each test case covers functional requirements. The figure 1 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 1 Test case mapping table

3.2 Test Environment

Testing will occur in the following environment:

Hardware: Hiking Assistant Smartwatch Arduino with Bluetooth devices for synchronization

Software: Custom smartwatch firmware for step counting, session tracking, and synchronization, Arduino firmware to handle data reception, calorie calculation, and LED display control

Test Tools: Bluetooth sniffer for monitoring communication, debugging tools for tracking internal states and data flow.

3.3 Risk Assessment

Risk 1: Failure of Synchronization Between the Smartwatch and Arduino Due to Bluetooth Interference
Synchronization between the smartwatch and Arduino may fail due to interference or environmental factors affecting Bluetooth communication.
Mitigation: To address this, multiple test runs will be performed in different environments, simulating potential interference scenarios. The system will be tested to ensure that synchronization retries, and notifications are correctly implemented, and the connection is stable under various conditions.

Risk 2: Incorrect Stride Length Settings Affecting Distance Calculation
If the stride length is set incorrectly by the user, the distance calculation may be inaccurate, leading to incorrect session statistics.

Mitigation: The stride length setting will be verified through user configuration tests. The system will be tested under various stride length scenarios to ensure accurate distance calculation. A user-friendly calibration process will be available to adjust stride length for better accuracy.

Risk 3: Data Loss or Corruption During Synchronization
During synchronization between the smartwatch and Arduino, there is a possibility of data loss or corruption due to incomplete transmission or system failure.
Mitigation: Data integrity checks will be implemented to verify successful transmission. The system will be tested for edge cases, such as partial data transfers, and recovery mechanisms will be put in place. If synchronization fails, the system will retry and notify the user of the issue.

Risk 4: Insufficient Battery Life During Long Hiking Sessions
Prolonged use of the smartwatch in hiking sessions may lead to excessive battery drain, especially when Bluetooth function is active.
Mitigation: The system will be tested for battery life under heavy usage, including extended hiking sessions. Power-saving features will be validated to ensure they activate automatically when necessary. Battery consumption will be monitored to ensure the system can meet the demands of long hikes.

Risk 5: Inaccurate Calorie Calculation Due to Missing or Incorrect Elevation Data
If the Arduino lacks accurate elevation data, it may fail to adjust the calorie calculation properly, resulting in inaccurate session statistics.
Mitigation: The system will be tested under varying elevation conditions to ensure that the Arduino correctly adjusts the calorie calculation when data is available. If elevation data is missing or incorrect, the system will fallback to default calculations, and users will be notified of discrepancies.

3.4 Test Schedule

Testing will be conducted over 4 weeks, divided into the following phases:

- Week 1: Unit testing for both the smartwatch and Arduino.
- Week 2: Integration testing of the Bluetooth communication.
- Week 3: System testing with the full functionality, including synchronization and data processing.
- Week 4: Final regression testing and defect fixing.

Documentation will be done during each week.

4. Test Cases

Here are listed all the test cases with test steps and expected results.

TC-1: Watch Power on and Start Initial Program

- **Objective**: Verify that the watch turns on when the power button is pressed for 3 seconds and starts the initial program.
- **Test Steps**:
    1. Press and hold the power button for 3 seconds.
    2. Ensure the watch powers on and starts the initial program.
- **Expected Results**: Watch should power on and begin the initialization process.

TC-2: Display Standby View

- **Objective**: Confirm that the watch displays the program standby view with the application name and access to settings.
- **Test Steps**:
    1. Power on the watch.
    2. Verify the standby view displays the app name and the settings button.
- **Expected Results**: The standby view shows the app name “Hiking Assistant” and a touchscreen button for settings.

TC-3: Start Hiking Session

- **Objective**: Verify that the hiking session starts by pressing the power button for 1 second.
- **Test Steps**:
    1. In standby mode, press and hold the power button for 1 second.
    2. Verify that the hiking session starts.
- **Expected Results**: Hiking session should begin.

TC-4: Step Counting and Distance Calculation

- **Objective**: Ensure the watch counts steps and calculates distance correctly.
- **Test Steps**:
    1. Start the hiking session.
    2. Walk around and ensure the watch counts steps.
    3. Verify that the distance is calculated correctly based on stride length and steps.
- **Expected Results**: Steps and distance should be displayed and updated correctly.

TC-5: Display Hiking Session Data

- **Objective**: Verify that the watch displays updated session data (steps, distance, and time).
- **Test Steps**:
    1. Start the hiking session.
    2. Ensure that the display updates with steps, distance, and time.
- **Expected Results**: Watch should show updated statistics after each interrupt.

TC-6: Synchronization with Arduino

- **Objective**: Verify that the watch synchronizes with Arduino and transmits data successfully.
- **Test Steps**:

1. Stop the hiking session.
2. Attempt synchronization with Arduino.

- **Expected Results**: Data should be transmitted successfully, and the user should be notified of success or failure.

TC-7: Arduino Data Processing

- **Objective**: Verify that Arduino processes received data and calculates calories.
- **Test Steps**:
  1. Arduino receives session data from the watch.
  2. Verify that the data is parsed and stored.
  3. Confirm that calories are calculated and displayed.
- **Expected Results**: Arduino processes and displays the correct session statistics.

TC-8: Synchronization and Acknowledgment from Arduino

- **Objective**: Verify that the Arduino acknowledges the received data and the watch is notified about the synchronization status (successful or failed).
- **Test Steps**:
  1. Stop the hiking session on the watch.
  2. Initiate synchronization with the Arduino.
  3. Verify that Arduino acknowledges the reception of the data.
  4. Ensure that the watch receives feedback about the synchronization status (success or failure).
- **Expected Results**:
  - Arduino acknowledges data receipt and the synchronization status is communicated to the watch.
  - If successful, the watch should show confirmation; if unsuccessful, it should display an error message or icon.

TC-9: Automatic Synchronization and Range

- **Objective**: Verify that the watch attempts automatic synchronization when it is within range of the Arduino.
- **Test Steps**:
  1. Ensure the automatic synchronization setting on the watch is enabled.
  2. Move the watch within Bluetooth range of the Arduino.
  3. Verify that the watch attempts synchronization automatically.
  4. Check that the Arduino begins polling the BLE sensor for incoming transmissions.
- **Expected Results**:
  - The watch should automatically initiate synchronization when in range of the Arduino.
  - Arduino should begin polling the BLE sensor and process the data.

TC-10: Bluetooth Synchronization

- **Objective**: Verify that the watch can establish a Bluetooth connection with the Arduino if the Bluetooth synchronization option is enabled.

- **Test Steps**:
    1. Enable the Bluetooth synchronization setting on the watch.
    2. Attempt to synchronize the watch with the Arduino over Bluetooth
    3. Verify that the Arduino adjusts the calorie calculation formula if elevation data is available.
    4. Confirm that the Arduino processes the session statistics, including the steps, distance, duration, and calories, and displays them on the LED screen.
- **Expected Results**:
    - The watch should establish a Bluetooth connection with the Arduino and attempt synchronization.
    - Arduino should adjust calorie calculations based on the available elevation data and correctly display the session statistics.

Post-test procedure: Record of results, cleaning up test data, and resetting systems. Documentation of the results.

5. Conclusion

This Test Plan outlines the comprehensive approach for testing the Hiking Tour Assistant system. It covers all functional and non-functional requirements, ensuring that the system meets the specified objectives. Testing will be performed in stages, starting with unit tests, followed by integration and system tests. By conducting thorough testing, we aim to ensure that the system functions reliably and meets the needs of users for hiking sessions, data synchronization, and display of statistics.