Interpersonal Motor Synchronization Through Smartwatches

Yuval Roth, Gal Cohen, Olga Oznovich, Tamir Avisar

-ARD-

Contents

Chapter 1

Introduction

1.1 The Problem Domain

Interpersonal Motor Synchronization (IMS) is a phenomenon where individuals synchronize their movements, often fostering a sense of connection, cooperation, and social bonding. While traditionally observed in co-located settings such as dancing, rowing, or therapy, this project explores how IMS can be enabled in remote interactions through smartwatch applications. The wearable nature of smartwatches, coupled with their ability to provide haptic, visual, and auditory feedback, makes them an ideal medium for experimenting with IMS in real-time remote communication scenarios.

1.2 Context

The proposed system operates as a smartwatch-based application that enables real-time IMS between two users. The system leverages the communication capabilities of WearOS and (potentially) WatchOS smartwatches and a game server to establish connections and manage sessions.

The application integrates sensors available in smartwatches to measure and synchronize inputs such as tapping or swiping. Visual, auditory, and haptic feedback mechanisms are used to provide immediate feedback on synchronization. A data collection server is integrated into the system to store user interaction data, enabling future research and optimization.

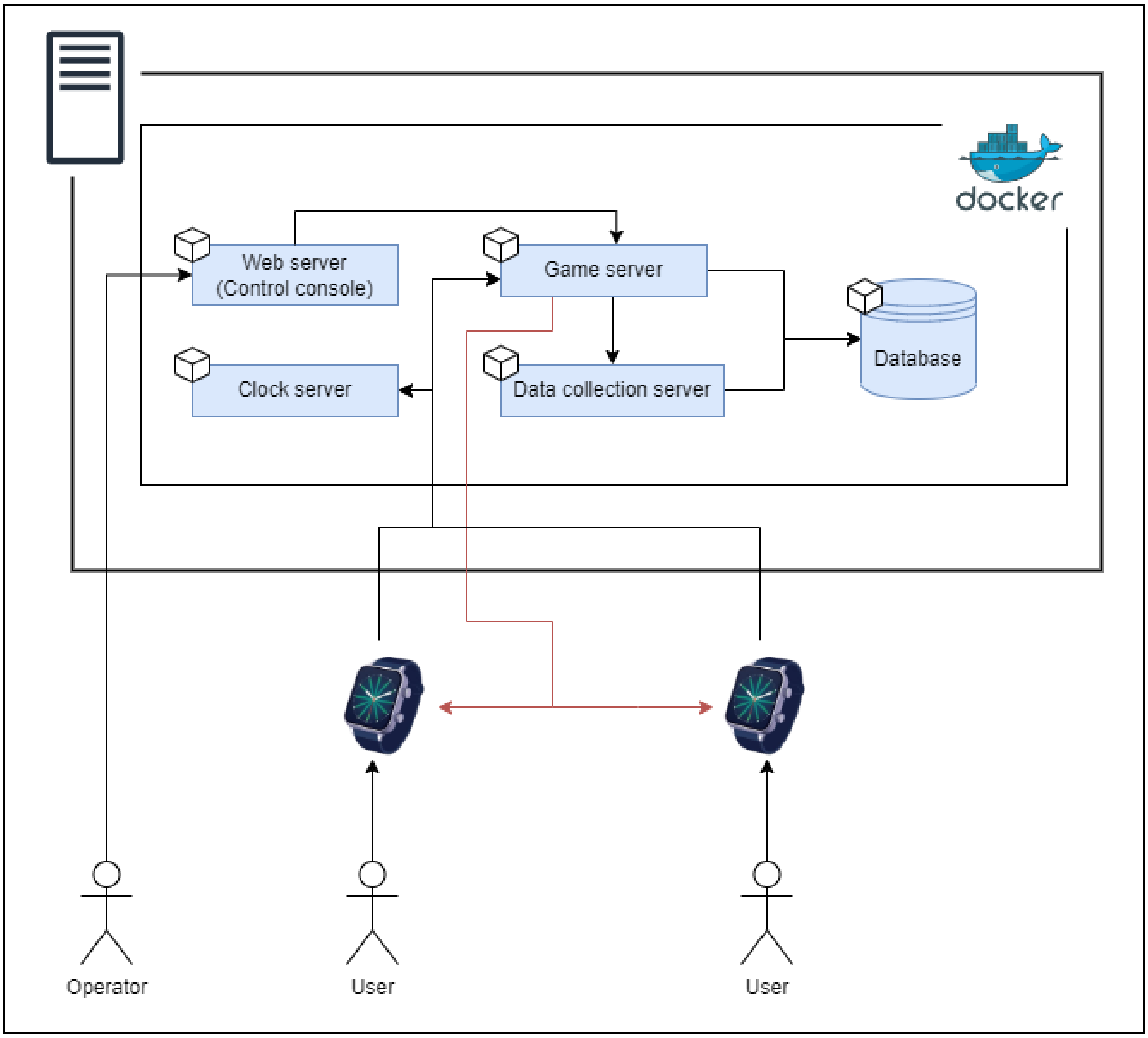
Additionally, a web interface is included to act as a manager's front end, facilitating session control and participant management. The web interface communicates with the game server, allowing the manager to insert and edit participant details, monitor active participants and their paired watches, pair two participants in a lobby while managing their readiness status, and start the IMS session only when both participants in the lobby are marked as ready.

This project is built upon similar concepts from our client's previous research of IMS through technological means, but in this project we use smartwatches instead of smartphones to conduct the research, which gives the researchers much more information about the participants.

The main components of the system include:

- **Smartwatches**: Responsible for input/output interactions, communication, and data collection.
- **Communication Infrastructure**: Uses UDP-based protocols for real-time interaction, facilitated by a game server.

- **Game Server**: Manages device connections, synchronization, and communication between participants.
- **Web Interface:** A tool exclusively for the manager, enabling participant and session management and monitoring.
- **Data Collection Server**: Stores and organizes data such as session duration, synchronization metrics, and physiological readings for further analysis.



1.3 Vision

The primary objective of this project is to explore how **remote IMS can enhance interpersonal interactions using wearable technology.**

The application aims to:

- Facilitate real-time IMS between remote users.
- Utilize smartwatches unique capabilities to provide multisensory feedback.

- Collect and analyze data including physiological measures to improve synchronization methods and understand its effects on social bonding.
- Offer an intuitive and engaging user interface, with meditative and incentivized designs, to maximize user **experience.**

Ultimately, the project seeks to create a groundbreaking system that bridges physical distances, fostering deeper connections through synchronized movements.

1.4 Stakeholders

- **Researchers:** Michal Rinott and Noam Tractinsky, who aim to study IMS in remote settings using wearable technology.
- **End Users:** Individuals participating in the pilot study, potentially including students, colleagues, or other remote collaborators.
- **Developers:** Responsible for implementing the smartwatch app, backend systems, and communication protocols.
- **Data Analysts:** Researchers analyzing the collected data to derive insights about IMS and its implications.

1.5 Software Context

The system is a smartwatch-based application with the following key components:

- **Inputs**:
  - User interactions (e.g., tapping, circular swiping), physiological data (heart rate, stability).
  - Manager input via the web interface (e.g., adding/editing participants, pairing users, starting sessions).

- **Processing**:
  - Synchronization metrics, latency handling, and feedback generation.
  - Management commands processed via the game server.

- **Outputs**:
  - Multisensory feedback, including haptic vibrations, visual changes, and auditory tones, delivered in real-time.
  - Manager view of active participants, paired devices, and session status via the web interface.

- **Communication**:
  - Device-to-device connections over a game server.
  - Web interface communication with the game server for session and participant management.

- **Data Collection**: Server-side storage of interaction and physiological data for further analysis.

Use cases include:

- **Tapping in Sync**: Two users tap on their smartwatches, receiving feedback on their synchronization through ripples or growing plants.
- **Circular Swiping**: Users swipe in circles, with haptic feedback representing the partner's actions and visual/audio cues for synchronous movements.
- **Data Collection for Analysis**: Interaction data is sent to a server to study the impact of IMS in remote communication scenarios.

Chapter 2

Usage Scenarios

2.0 Definitions

Lobby - An admin will create a connection session for two specific participants in the control console. That "connection session" is called a lobby.
IMS session – When the lobby is created and both participants signal readiness, the game session of the study is taking place. That "game session" is called an active IMS session.

2.1 User Profiles — The Actors

User – a human with a smart watch device.

Sensors – various sensors included in smartwatches, such as heart rate and temperature. Data is collected for further analysis
Admin – the study operator.

2.2 Use-cases

2.2.1 <u>Login with user id</u>

Actor: User

Description: A user which is not in a lobby, or an active IMS session will enter its' user id.

Preconditions:

- The user id is not registered in any watch.
- The user id is registered in the system.
- The user's device has a network connection.

Postconditions:

- The user will be associated with the watch.

Basic flow:

a. The user opens the IMS app on their smartwatch.

b. A prompt to enter user id is shown by the system.

c. The user enters their id.

d. The user is redirected to the main screen of the app.

Alternative flows:

- The user entered the wrong user id – the system will notify the user of the error and will allow the user to try again.

2.2.2 Connecting to an existing lobby

Actor: User

Description: A user which is not in a lobby, or an active IMS session will join an existing lobby.

Preconditions:

- The user is logged in to the watch.
- The user is not in a lobby.
- The user is not in an active IMS session.
- The user's device has a network connection.
- The user had been assigned to a session by an admin.

Postconditions:

- The user will be part of a lobby.

Basic flow:

a. From the main screen of the app, the user chooses to join a lobby.
b. The system will automatically direct the user to the lobby it was assigned to ahead of time.
c. The option to signal readiness is displayed.

Alternative flows:

- Lobby joining failed due to network issues – the system will notify the user of the error and will redirect to the main screen of the application.
- Lobby joining failed due to session assignment issues – the system will notify the user that it should contact the study operator and will redirect it to the main screen of the application.
- The user stopped the lobby joining in the middle – the system will redirect the user to the main screen of the application.

2.2.3 Sensor data collection

Actor: Sensors

Description: Sensor's information will be logged by the system every set time for further analysis.

Preconditions:

- An active IMS session is taking place.
- Sensor usage is enabled in the app's permissions
- A sensor detected information.

Postconditions:

- The new data will be saved.

Basic flow:

a. The application is running on the watch.
b. The system logs the data from the sensors every set amount of time.
c. The system saves the data locally until the end of the session when it sends it to the server.

Alternative flows:

- Data storage failure – the system will notify of the error and will attempt to log the data again. If still failed, data is not deleted until sent.

2.2.4 Performing IMS actions in session

Actor: User

Description: A user in an active IMS session will be able to perform the selected IMS action (e.g. swiping or tapping).

Preconditions:

- The user is part of an active IMS session.
- The action is the selected action for the session.

Postconditions:

- The performed action will be logged by the system.
- The other participant will receive feedback of the action.

Basic flow:

a. The application is running, and an active IMS session is taking place.
b. The user performs the IMS action.
c. The system logs the action.

d. The system notifies the rest of the party of the action.

Alternative flows:

- User's action not registered due to lag or malfunction – nothing special will happen, the system will continue waiting for the user's next action.
- No user actions detected for a set time – the system will pause the session and prompt the user with instructions and an option to resume.

2.2.5 Receiving IMS actions in session

Actor: IMS Session Process (contextual actor)

Description: During an active IMS session, chosen feedback (e.g. haptic vibration, sound indication or color changes) will be invoked due to an action performed by the other participant.

Preconditions:

- An active IMS session is taking place.
- The feedback type is enabled by the application.
- The other participant has performed the chosen IMS action.

Postconditions:

- The feedback has been performed in real-time.

Basic flow:

a. The application is running, and an active IMS session is taking place.
b. The system receives an IMS action from the other participant.
c. The system logs the received data.
d. The system performs the chosen feedback type.

Alternative flows:

- Sent action is delayed or not received – nothing special will happen, the system will continue as before.
- No actions received for a set time – the system will check the connection between the parties and pause the session.

2.2.6 Create and configure a new lobby

Actor: Admin

Description: The admin will be able to create a session and set all of the relevant settings via the control console.

Preconditions:

- The admin is logged in the system.
- Relevant participants are registered in the system.
- The computer running the console is connected to the network.

Postconditions:

- The session will be created

Basic Flow:

a. The control console is running on the admin's computer.
b. The admin chooses to create a lobby.
c. The admin assigns participants to the session.
d. The admin configures the session (length, iterations, action types, interpolation level)
e. The admin submits the configuration.

Alternative Flows:

- Invalid configuration settings – The system will perform input validation and alert the admin of invalid inputs.

2.2.7 Registering new participant

Actor: Admin

Description: The admin registers a new study participant to the system.

Preconditions:

- The admin is logged in the system.
- The computer running the console is connected to the network.

Postconditions:

- The new participant information will be saved in the database.

Basic Flow:

a. The control console is running on the admin's computer.
b. The admin chooses to register a participant.

c. The admin inserts relevant information such as name, id, age, gender, etc.

d. The admin submits the registration form.

Alternative Flows:

- Duplicate participant registration – An error should be returned from the database noting the participant is already registered.

2.2.8 Login to console

Actor: Admin

Description: A study operator will login to the control console with admin credentials.

Preconditions:

- Admin credentials exist in the system.
- The computer running the console is connected to the network.

Postconditions:

- Admin will be logged into the system.

Basic Flow:

a. The control console is running on the study operator's computer.

b. The study operator chooses the login option.

c. The study operator inserts login credentials.

d. The system redirects to the main menu window.

Alternative Flows:

- Admin already logged in another device – logging from two devices should not be able, so an error should be returned.

2.2.9 View system information

Actor: Admin

Description: The admin will be able to view information about registered participants, existing lobbies and running sessions.

Preconditions:

- The admin is logged in.
- The computer running the console is connected to the network.

Postconditions:

- Relevant information will be displayed to the admin.

Basic Flow:

a. The control console is running on the admin's computer.

b. The admin chooses to view information.

c. The admin chooses which information exactly to view.

d. The system displays the results.

Alternative Flows:

- Error in fetching the data from the server – The system will retry fetching the data. If the issue remains, an error popup will be shown.

2.2.10 Start a session

Actor: Admin

Description: The admin will start a specific experiment session.

Preconditions:

- The admin is logged in
- The users have signaled readiness to begin.

Postconditions:

- The session will be active on the user's watches and the console.

Basic Flow:

a. The control console is running on the admin's computer.

b. The admin chooses a session to start.

c. The admin starts the session.

d. The system notifies the participant's watches.

e. The system on the watches runs the session.

Alternative Flows:

- Session didn't start on at least one watch – There should be confirmation from the watches that they have received the start notification.

2.2.11 End a session

Actor: Admin

Description: The admin will end a specific experiment session.

Preconditions:

- The chosen session is currently running.

Postconditions:

- The users watches do not take part in any session.

Basic Flow:

a. The control console is running on the admin's computer.

b. The admin chooses a session to end.

c. The admin ends the session.

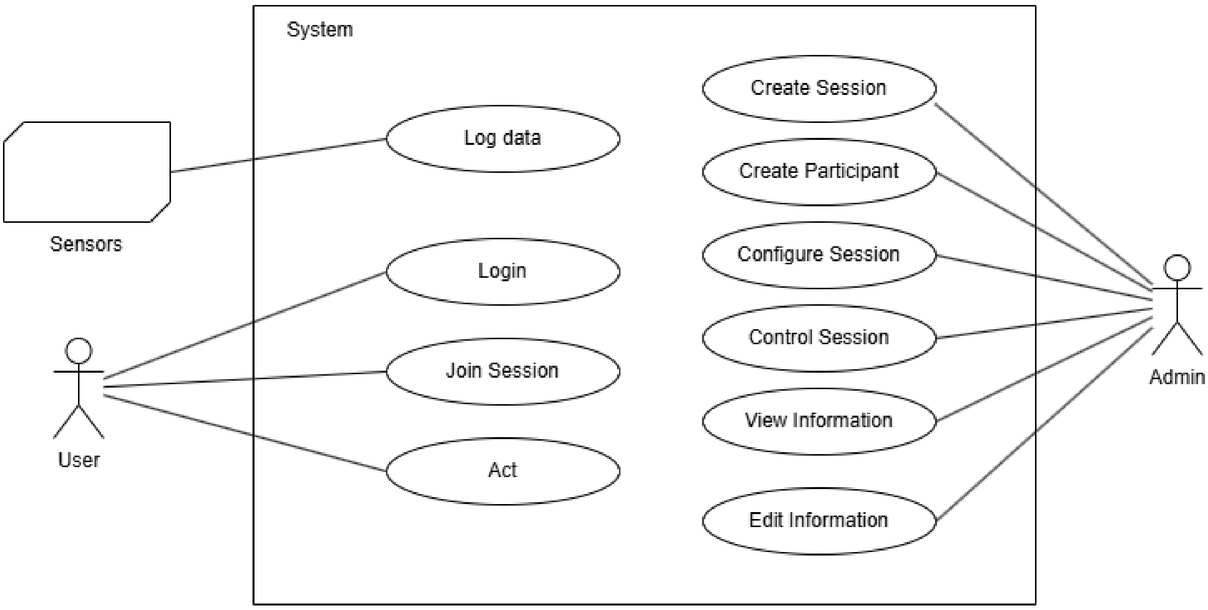
d. The system notifies the participant's watches.

e. The system on the watches terminates the session.

Alternative Flows:

- Session already ended – The system will notify the admin and will remove it from the control panel.

2.3 Use Case Diagram



2.4 Special usage considerations

N/A

Chapter 3

Functional Requirements

Overview

The smartwatch application enables Interpersonal Motor Synchronization (IMS) between two users through tapping and swiping interactions. Visual feedback is provided continuously during user actions, while haptic feedback is triggered based on server responses, indicating the synchronization status. A light haptic feedback is provided when users are not synchronized, and a stronger feedback is delivered when synchronization is achieved. The system also collects interaction and physiological data for analysis.

1. User Interaction and Feedback

1.1 <u>Input Handling</u>

- Detect tap inputs from users on the smartwatch screen.
- Detect finger position while swiping on the screen.
- Allow managers to input commands via the web interface to manage participants and sessions.

1.2 <u>Feedback Mechanisms</u>

1.2.1 Visual Feedback:

- Always provided during user actions (tapping or swiping).
- Includes effects such as ripples, glowing trails, or plant growth animations.

1.2.2 Haptic Feedback:

- Provided only when the server returns a response.
- Feedback strength indicates synchronization status

1.2.3 Audio Feedback:

- Harmonious tones are played when synchronization is achieved.

1.3 <u>User polling after the session ends:</u>

- In the smartwatch app, the user will be asked to rate the session between 1 to 7 immediately after the session ends.

2. Communication and Synchronization

2.1 <u>Real-Time Communication</u>

- Establish connections (via a game server).
- Support web interface communication with the game server for session control.

2.2 Synchronization Detection

- Detect synchronization based on the alignment of actions (timing and positions).
- Trigger appropriate feedback based on synchronization results.

2.3 Data Collection and Storage

2.3.1 Sensor Data Acquisition

- Collect interaction data, including tap and swipe timings and positions.
- Record physiological data such as heart rate and hand stability.

2.3.2 Data Storage and Analysis

- Transmit collected data to a remote data collection server.
- Store session details, including duration, synchronization attempts, and physiological metrics.

2.4 Connectivity and Stability

2.4.1 Device Discovery and Pairing

- Devices pair via the game server.
- Users can pair their smartwatch with their assigned user ID. Managers can pair two participants into a lobby.

2.4.2 Connection Management

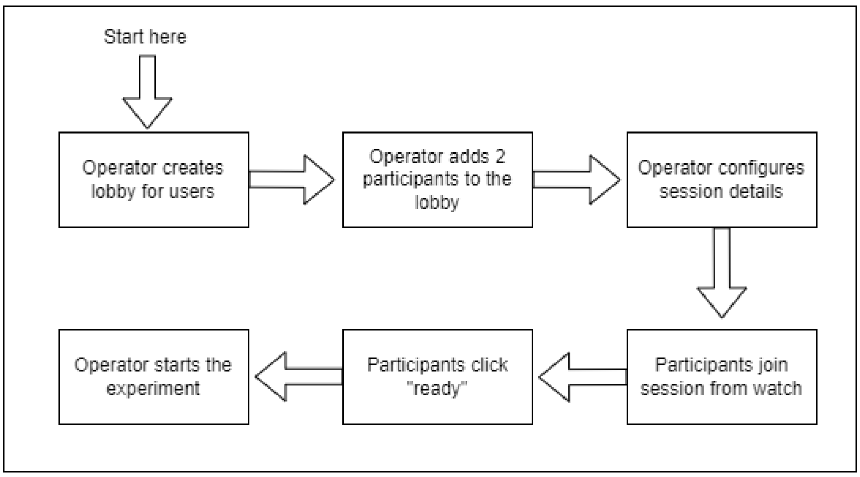
- Manage reconnection automatically if the connection is disrupted.
- Optimize feedback timing to mitigate potential latency effects.

3. Web Interface

3.1 Management Features

- The web interface is designed exclusively for the manager.
- Enable the manager to insert and edit participant details.
- There will be a list showing active participants* and their paired smartwatches
- Pair two participants into a lobby and be able to start the game. Start the IMS session only when both participants in the lobby are marked as ready.

*active participants – participants that are currently in session



Chapter 4

Non-functional requirements

4.1 Implementation constraints

Performance (Speed, Capacity, Throughput, etc.)

- The latency time for data transfer between 2 users will be minimal and less than 20ms, so the synchronization be optimal.
- At least 99% of the data samples will be recorded and saved on the server.
- The system will meet the capacity of 4 users at the same time.

Reliability & Stability

- The session will not be able to continue when the other user is not available.
- In the case of a short communication failure, the system will continue to operate with an error-corrector, but in the case of a long failure, the session will stop.

Safety & Security

- The data collected will be stored in an encrypted database with limited access.
- Only the researchers and those on their behalf will be able to access the data collected on the subjects.

Portability

- The app will work for Android operating system for smart watches (WearOS).

Usability

- Basic instructions will be displayed on the screen during use.
- A user will be able to operate the application without prior experience or training.
- The users will not be able to see any measures (like blood pressure for example) in real time so as not to be influenced by them.
- The screen won't turn off as long as the session is active.

Availability

- The application should be available at least while performing the experiments.

User polling

- At the end of the session, the user will fill a form with questions for the research

4.2 Platform constraints

- The watch application must be for wearOS 4

4.2.1 SE Project constraints

- The real data will be collected during experiments on different users.
- A remote cloud server is required that will allow a connection to be made from anywhere.
- It is required to collect the data in such a way that it can be exported in a readable and understandable way.

4.3 Special restrictions & limitations

N/A

Chapter 5

Risk assessment & Plan for the proof of concept

The proof-of-concept prototype is divided into two parts:

1. Developing a rudimentary server and smartwatch app to serve as the foundation for implementing the communication infrastructure for interaction between the project's components.
2. Starting the GUI design to create a user-friendly visual framework for the app, without incorporating any game logic at this stage. Out of four potential GUI options, we will be starting with the simplest one to evaluate its effectiveness

<u>How will this help us, the developers?</u>

In interpersonal motor synchronization, low latency and smooth real-time communication are very important.

By building a basic server and smartwatch app, we will be able to focus on the fundamentals and better understand the technical needs for communication between the devices. This includes figuring out the right protocols, data formats, and performance measures (latency and connection stability) needed for smooth interaction between the server and the smartwatches.

By starting with the simplest GUI option, we will learn the basics of the android GUI framework and be able to apply the knowledge we gained when working on the more complicated options

Furthermore, by having something to show for the GUI, our client will also be able to weigh in on the design and further adjust the requirements if he sees any flaws that will come up in the early stages.

<u>Risk assessment</u>

None of us started this project with any knowledge about Kotlin nor the android system and its GUI framework. As such, developing this prototype will be a challenging task.

However, it is an opportunity for us to learn Kotlin and the android operating system which will help us further down the road when implementing the more complicated parts of the project.

Thanks to the prototype being small scale and having relatively basic requirements, we believe will be able to learn quickly enough and accomplish the task.