Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

An architecture for seamless communication between wearables and connected vehicles

Yuye Tong

Course of Study:	Informatik
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Christian Hackenbeck, M.Sc., Mercedes-Benz
Commenced:	February 15, 2023
Completed:	August 15, 2023

Abstract

This thesis explores the potential and complexities that arise from the integration of wearable technology with connected vehicles. With the rise of hyper-connected lifestyles, wearable devices like smartwatches, smart bands, and fitness trackers are not only ubiquitous but are also collecting valuable biophysical data that can potentially enhance driver and passenger safety and comfort. Likewise, vehicles have evolved beyond mere transportation modes, transforming into sophisticated, sensor-laden, communication-enabled systems. Despite their distinct advancements, little research has engaged with the interconnectivity of heterogeneous wearable devices with third-party systems, particularly connected vehicles.

To bridge this research void, this work presents a novel wearable-vehicle integration platform. Through a systematic three-phased methodology encompassing exploration, conceptualization, and experimentation, an innovative architecture emerges. Designed with components such as the device layer, device manager, service proxy, and backend connector, it not only addresses the fragmentation problem identified through technical market research in the exploration phase but also aspires to foster a cohesive ecosystem for varied wearable manufacturers.

By promoting a unified ecosystem for wearables and connected vehicles, the proposed architecture not only promises enhanced user experiences and safety measures but also paves the way for manufacturers to exploit a broader spectrum of wearable data. Consequently, industries and end-users are expected to benefit from more personalized, efficient, and interconnected vehicular systems.

Insights from this study underline that the challenge of wearable-vehicle integration is multidimensional. It is not just about bridging technological divides; it is also about navigating the dynamic business landscapes, vendor challenges, and overarching safety and security considerations. As technological frontiers expand, the findings of this research stand foundational, laying a groundwork for future innovations and studies in wearable technology and vehicular interconnectivity.

Kurzfassung

Diese Arbeit untersucht das Potenzial und die Komplexitäten, die sich aus der Integration von tragbarer Technologie in vernetzte Fahrzeuge ergeben. Mit dem Aufkommen von hyper-vernetzten Lebensweisen sind tragbare Geräte wie Smartwatches, Smartbands und Fitness-Tracker nicht nur allgegenwärtig, sondern sammeln auch wertvolle biophysische Daten, die die Sicherheit und den Komfort von Fahrern und Passagieren potenziell erhöhen können. Ebenso haben sich Fahrzeuge über bloße Transportmittel hinaus entwickelt und sich in hochentwickelte, mit Sensoren ausgestattete, kommunikationsfähige Systeme verwandelt. Trotz ihrer unterschiedlichen Fortschritte wurde die Verknüpfung von heterogenen tragbaren Geräten mit Drittsystemen, insbesondere vernetzten Fahrzeugen, nur wenig erforscht.

Um diese Forschungslücke zu schließen, stellt diese Arbeit eine neuartige Plattform zur Integration von Wearables und Fahrzeugen vor. Durch eine systematische, dreiphasige Methodik, die Erkundung, Konzeptualisierung und Experimentierung umfasst, entsteht eine innovative Architektur. Entwickelt mit Komponenten wie der Geräteebene, dem Gerätemanager, dem Service-Proxy und dem Backend-Connector, adressiert sie nicht nur das durch technische Marktstudien in der Erkundungsphase identifizierte Fragmentierungsproblem, sondern strebt auch danach, ein kohärentes Ökosystem für verschiedene Wearable-Hersteller zu schaffen.

Durch die Förderung eines einheitlichen Ökosystems für Wearables und vernetzte Fahrzeuge verspricht die vorgeschlagene Architektur nicht nur verbesserte Benutzererfahrungen und Sicherheitsmaßnahmen, sondern ebnet auch den Weg für Hersteller, ein breiteres Spektrum an Wearable-Daten zu nutzen. Folglich wird erwartet, dass Industrien und Endverbraucher von personalisierten, effizienten und vernetzten Fahrzeugsystemen profitieren.

Erkenntnisse aus dieser Studie unterstreichen, dass die Herausforderung der Integration von Wearables in Fahrzeuge vielschichtig ist. Es geht nicht nur darum, technologische Kluften zu überbrücken; es geht auch darum, sich in dynamischen Geschäftslandschaften, Herausforderungen von Anbietern und übergreifenden Sicherheits- und Sicherheitserwägungen zurechtzufinden. Da technologische Grenzen erweitert werden, bilden die Ergebnisse dieser Forschung eine solide Basis und legen ein Fundament für zukünftige Innovationen und Studien in der tragbaren Technologie und der Vernetzung von Fahrzeugen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In today's age of technological convergence, the overlap between domains like wearable technology and vehicular communication is inevitable.

Wearable devices, such as smartbands and smartwatches, have seen widespread adoption worldwide, with the market predicted to experience significant growth [OSK+21]. These devices offer a variaty of services, from health monitoring to navigation assistance, and have become pervasive in daily routines.

On the other hand, vehicles have transitioned from being merely a mode of transport to sophisticated, connected systems with onboard sensors, advanced infortainment systems, and communication capabilities. These vehicles, often termed as "connected vehicles", have the potential to transform the way we commute, ensuring safety, efficiency, and personalization.

The convergence of wearables and connected vehicles holds the potential to introduce diverse applications, from personalized vehicle settings based on wearable data to emergency health responses.

1.2 Problem Statement

While both wearables and connected vehicles have made remarkable strides individually, there exists a gap in research focused on their combined potential. The communication landscape is challenged by diverse protocols and standards, especially when considering the variety of approaches introduced by different wearable manufacturers. Key issues that emerge include: How can we establish uniform communication? How can wearable data optimize vehicular experiences? And, most importantly, what architecture can best support this integration?

1.3 Objective

This thesis aims to holistically analyze and explore the possiblity of wearable-vehicle communication. By understanding the challenges, requirements, and potential applications, we seek to develop an extensible solution that not only integrates the two domains but also paves the way for future innovations.

1.4 Research Approach

Our methodology is structured across three distinct phases: exploration, conceptualization, and experimentation.

- **Exploratory Phase:** We begin with a thorough technical market research to identify the distinguishing characteristics of wearables, with the intention of abstracting wearables types and communication technologies for the architecture design.
- **Conceptualization Phase:** In this phase, the focus is on the development of a software architecture blueprint. It addresses the essential components that are vital for wearable-vehicle interoperability.
- **Experimental Phase:** Here, the proposed architecture is validated through a proof of concept. Challenges encountered during implementation are addressed, with insights provided for future research directions.

1.5 Thesis Structure

Chapter 2 will discuss related work, highlighting the current state-of-the-art and identifying gaps in existing research. In Chapter 3, we will investigate wearables from different aspects to examine and gain a deeper understanding of these devices. Subsequent chapters will analyze extensively the proposed architecture, its components, functionalities, and potential applications. Towards the end, we will conduct experiments and evaluate a proof of concept of the proposed architecture, and conclude with future recommendations and implications.

2 Related Work

The integration and interoperability of wearables and other IoT devices with various platforms have become a central theme in recent research. A spectrum of approaches has emerged, each addressing different challenges and application domains. In this section, we discuss various studies that contribute to our understanding of wearable integration and how they relate to our work.

Mavrogiorgou et al. [MKK22] proposes a four-layered IoT data integration middleware that automatically connects and integrates IoT heterogeneous wearable medical devices with various healthcare platforms, while interacting only with the reliable and relevant ones. The middleware is limited to a Bluetooth interface for the connection of available devices, and the Devices Type Recognition layer uses image techniques to recognize the type of the connected unknown devices. The proposed middleware addresses the challenge of gathering heterogeneous IoT wearable medical devices' data and integrating only the data that is of high-reliability. The paper offers valuable insights into the management of IoT wearable devices, which is a highly relevant aspects for us.

Another exploration into the realm of wearables is by de Arriba-Pérez et al. [ACS16], who addresses the interoperability challenges related to the collection and processing of data from wearable devices for the purpose of exploiting the data in the educational context. The authors propose guidelines to solve key interoperability issues and identify the need for standardized data models and data transfer solutions in the wearable domain. The guidelines offer a multi-layer approach to developing interoperability solutions, which includes connectivity, data, authentication, and user levels. The paper highlights the potential of wearable data, specifically sleep and stress indicators, to support educational scenarios and inspire us with analogous potential applications in the domain of connected vehicles.

Shifting the focus to embedded devices, Guinard et al. [GTK+10] introduced an integration architecture that promotes efficient service search on these devices and deploys missing functionalities upon request. The paper leverages Service-Oriented Architecture (SOA) approaches, traditionally used for for heavyweight IT systems and found in corporate IT systems, and applies them to real-world devices. This dynamic architecture enables the query, selection, and use of real-world services in complex business applications. Their paper, though focusing on embedded devices providing atomic operations, provides significant inspiration for our work due to its layered architecture and the abstraction within its device layer. Wearables, being multi-functional devices, pose greater integration challenges, pushing us to conceive an expansive support for integration to ensure their smooth inclusion within an IoT network.

To understand the broader landscape, Ometov et al.'s survey [OSK+21] offers a holistic view of wearable technology. Their comprehensive review discusses the evolution and current state of wearable technology, offering foundational knowledge of various wearables. While it deals with wearables at a macroscopic level, it forms the foundational knowledge base for understanding the diverse spectrum of wearable devices.

Our work addresses a significant research gap that exists. To our knowledge, there is no accessible research that presents an extensible integration solution for various wearables with vehicles. As an exploration into this uncharted territory, our work aims to contribute to the body of knowledge by proposing a comprehensive high-level architecture for wearable-vehicle integration. By building upon the insights provided by existing studies and taking inspiration from their methodologies, we seek to offer a novel solution that not only bridges the gap but also lays the foundation for further advancements in this emerging field.

3 Wearables and the Fragmentation Problem

In this chapter, we will review the state of the art with respect to the current wearable device market and specifically focus on the available types of application programming interfaces (APIs), functionalities and communication technologies supported by different market vendors.

3.1 Types of Wearables

Wearable devices, also known as wearables or wearable technology, are electronic devices that can be worn as near-body accessories or clothes or that can be tattooed on the skin or implanted on the body. In this work, the terms “wearable(s)” and “wearable device(s)” are interchangeably used to denote such devices. These wearables can be categorized based on their product types. Our focus in this research is on wrist- and hand-based wearable devices, including activity trackers, smart bands, smartwatches, and smart rings, as outlined in Table 3.1.

The selection of activity trackers, smart bands, smartwatches, and smart rings is driven by their widespread adoption for daily life. Their diverse capabilities also make them ideal representatives for examining integration challenges and potential solutions in the wearable-vehicle integration context.

Activity trackers	Simple and relatively cheap devices mainly focused on everyday activity monitoring, including the number of steps, basic heart rate, and/or body temperature data collection. The main goal is to increase the overall physical activity participation of an average user.
Smart bands	Carrying the functionality of modern activity trackers but sometimes also providing gesture recognition, stress/mood detection, or ECG monitoring functionality.
Smart watches	The most widely adopted wearables after the activity tracker. Generally, they provide almost the same functionality as smartphones.
Smart rings	Similar functionality as activity trackers but in a smaller form-factor and without displays. Some smart rings also have a notification device functionality but are kept in a fashionable accessorized form.

Table 3.1: Snippet of table *Classification based on the wearable device type* from [OSK+21]

3.2 Functionalities of Wearables

Wearable devices provide a wide range of functionalities that can be tailored to meet the needs and preferences of individual users. Some of the most common functionalities offered by wearable devices include:

Fitness and Health Tracking Wearables equipped with sensors can monitor and track various health and fitness metrics. They can provide real-time data on heart rate, steps taken, calories burned, sleep patterns, and more. This functionality helps users maintain an active and healthy lifestyle by providing insights into their physical activity and overall well-being.

Notifications and Messaging Wearables can receive and display notifications, messages, and calls from a paired smartphone or as standalone devices with built-in SIM cards. This means that users can stay connected and receive important updates directly on their wearable devices, eliminating the need to constantly check their phones. Whether paired with a smartphone or functioning independently, wearables provide a convenient way to stay informed and connected throughout the day.

Media Control Many wearables offer media control capabilities, enabling users to manage media playback on their paired smartphones. Users can play or pause music, adjust volume, and skip tracks directly from their wearable device, providing convenient access to their media content.

Navigation and Location Services Some wearables incorporate navigation and location services, providing users with turn-by-turn directions [TBT] and GPS tracking. This functionality is particularly useful for activities such as running, hiking, or exploring new places, where users can receive guidance and track their location directly from their wearable device.

Payment and Ticketing Certain wearables support contactless payment and ticketing functionalities, allowing users to make purchases or access events by simply tapping their device. This feature eliminates the need to carry physical wallets or tickets, offering a convenient and seamless payment experience.

Voice Assistants Wearables integrate voice assistant technology, such as Siri, Alexa, or Google Assistant, enabling users to interact with their wearable device using voice commands. Users can ask questions, request information, set reminders, and perform various tasks, all through natural language interactions with the voice assistant.

Smart Home Control Some wearables provide the ability to control smart home devices, allowing users to manage their connected home appliances, lights, thermostats, and security systems directly from their wearable device. This functionality offers convenience and remote control capabilities, enhancing the smart home experience.

The functionalities offered by wearables can vary depending on the manufacturer, the operating system it runs on, and the specific model. This diversity contributes to the fragmentation of wearables, which will be further explored and closely examined in the following section.

3.3 The Fragmentation Problem

The proliferation of wearable technology has led to the emergence of a diverse range of functionalities that cater to the needs of distinct customer groups, vendors, hardware, and software platforms. While this has enabled the provision of personalized services and experiences, it has also engendered the fragmentation problem, characterized by the lack of interoperability and standardization among different wearable devices and the need for specialized applications to operate them. This phenomenon poses significant challenges to the development of a unified wearable ecosystem or integration platform, which are critical for the seamless exchange of data and interoperability among devices.

In this section, we analyze the fragmentation problem from multiple viewpoints, including the market, hardware, and software. By examining the root causes of fragmentation in wearables and exploring its implications for the industry, we aim to propose potential solutions that promote greater standardization and convergence in this burgeoning market.

3.3.1 Market Fragmentation

The wearables market experienced a slowdown [IDC1] in growth in 2022 due to the economic challenges at a global level, despite significant growth witnessed during the initial two years of the pandemic. However, the wearables category is expected to revive in 2023 with an anticipated decrease in the economic slowdown and increasing demand from emerging markets [IDC1]. The wearable device market is also projected to experience substantial growth over the next five years, as predicted by industry experts [MI4][LPI5].

According to industry research, the smart wearable market is characterized by a moderate level of concentration, with a handful of major players holding a significant market share, but with a large number of other players also competing for market share [MI4]. As indicate in Figure 3.1, the market is neither fully consolidated nor highly fragmented, but rather falls somewhere in between. Despite the presence of established players, the market remains highly competitive, with companies seeking to differentiate themselves through innovation, branding, and other strategies in order to gain a competitive advantage and capture a larger share of the market.

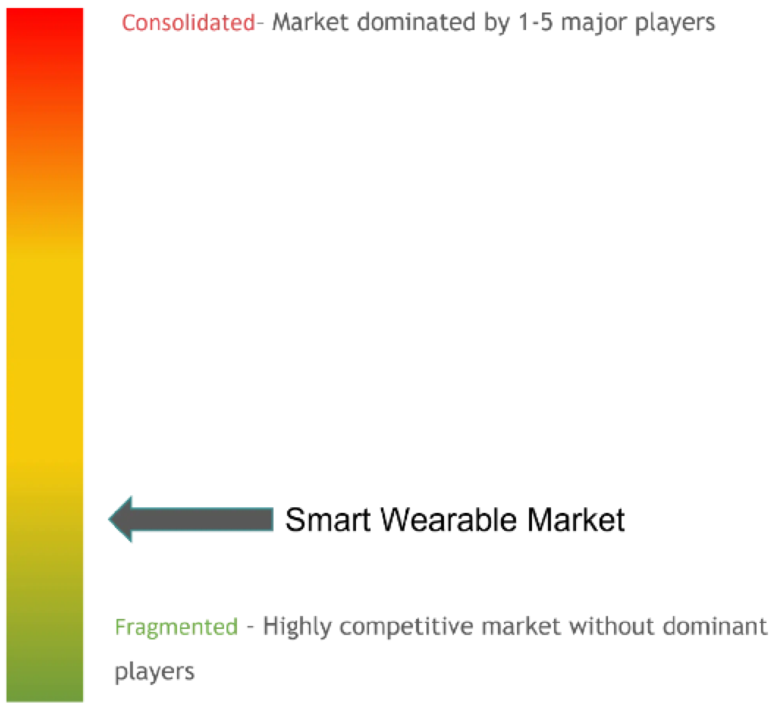

Consolidated- Market dominated by 1-5 major players
Smart Wearable Market
Fragmented - Highly competitive market without dominant players

Figure 3.1: Smart wearable market concentration from [MI4]

Market research firms categorize the wearable devices market into various segments based on product types and geography, with each segment having different players and market dynamics. Consequently, the market share statistics can vary depending on the source and the specific market segment being analyzed. Furthermore, the numbers can change over time as new products are introduced and consumer preferences evolve.

Owing to the market segmentation and dynamics, we have identified a list of major market players, based on multiple sources, including [MI4]. However, this list is unordered, and the positioning of the players may fluctuate depending on the market segment and timeframe being considered.

- Apple Inc.
- Google LLC
- Xiaomi Corporation
- Samsung Electronics Co., Ltd.
- Huawei Technologies Co., Ltd.
- Garmin Ltd.
- Fitbit Inc. (acquired by Google)
- Amazfit
- Imoo (watch phone for kids)

- Smart ring brands: Motive Ring, Oura Ring, Nimb Smart Ring, K Ring

Apple Apple Inc. currently owns the leading place according to the global smartwatch market share based on shipments by quarter [CTM]. Apple's wearable devices, such as the Apple Watch and AirPods, have become increasingly popular since their release. These devices have not only revolutionized the way we interact with technology but also how we manage our daily lives. The Apple Watch [APE], for example, offers a variety of features, such as fitness tracking, heart rate monitoring, sleep tracking, and GPS functionality. Users can also receive notifications, make phone calls, send texts, and access a wide range of apps directly from their wrist. All Apple Watch models feature a touch-sensitive screen that can recognize different types of touch inputs, such as taps and swipes.

The watch's sensors are one of its key features. These include a heart rate sensor that uses photoplethysmography to measure the wearer's heart rate, and a blood oxygen sensor that measures the level of oxygen in the wearer's blood. There's also a gyroscope, accelerometer, and barometric altimeter, which enable the watch to track the wearer's movements and activities. Additional sensors, e.g., electrical heart sensor, might be availbale, depending on the model.

The Apple Watch also features a variety of wireless connectivity options, including Wi-Fi and Bluetooth. Cellular connectivity is available on the cellular models of the watch, which allow users to make and receive phone calls and texts even when they're not near their iPhone.

Apple differentiates itself with its ecosystem integration and seamless user experience. The tight integration between Apple's wearable devices, such as the Apple Watch and AirPods, and its other products like iPhones and Macs, provides a cohesive user experience.

Samsung Samsung is another major player in the wearable device market. Samsung's wearable devices include the Galaxy Watch series and the Galaxy Buds earbuds [SAG]. The Galaxy Watch series is Samsung's line of smartwatches that offers a range of health and fitness tracking features, while also providing smartwatch functionalities and compatibility with Android and iOS devices. The latest model is the Galaxy Watch 5, which runs on the new Wear OS platform co-developed by Samsung and Google since the Galaxy Watch 4 [TIN]. Samsung's partnership with Google for the development of the Wear OS platform further enhances its smartwatch offerings.

Garmin Garmin is a well-known brand in the wearable device market, offering a wide range of devices that cater to different user needs and preferences [GAN]. Some of Garmin's most popular wearable devices include the Garmin Forerunner series and the Garmin Venu series.

Garmin's specialization in fitness and sports tracking sets it apart from other market players. The Garmin Forerunner series is designed specially for runners and other athletes, and offers a range of features that help users track their workouts and performance. These watches include features such as GPS, heart rate monitoring, and a variety of training modes and plans that can help users improve their fitness and reach their goals.

The Garmin Venu series, on the other hand, is designed for users who want a more general-purpose smartwatch that can be used for a variety of activities. The latest model in this series is the Garmin Venu 2, which offers a range of features such as GPS, heart rate monitoring, and a variety of workout tracking modes. It also includes features such as music storage and streaming, contactless payments, and smart notifications.

Fitbit Fitbit, another wearable fitness giant, is famous for it fitness tracker on the global market. Fitbit products excel in battery life and fitness features. Fitbit was purchased by Google in January 2021. So far, Fitbit and Google Fit remain alongside as two seperate platforms.

Google's purchase of Fitbit has had a significant influence on the company and its products. Since the acquisition, Fitbit has been integrated more closely with Google, and has started to adopt some of Google's technologies and services.

One of the most notable changes since the acquisition has been the integration of Google's voice assistant, Google Assistant, into the latest Fitbit smartwatches. This allows users to control their smartwatch and access other Google services with voice commands, making it easier and more convenient to use [FIT].

Another significant change has been the integration of Fitbit with Google's cloud services, which enables the company to leverage Google's expertise in machine learning and data analysis to develop new and more advanced health and fitness tracking features. This could potentially lead to more accurate tracking and analysis of users' health data, as well as new insights into how to improve overall health and wellness.

In addition, the acquisition has also provided Fitbit with more resources and capabilities to invest in research and development. This has led to the development of new and more advanced wearable devices, for example, more Fitbit wearables can now detect AFib, receiving FDA approval recently to use heart rate sensors to capture data.

Overall, the purchase by Google has had a positive influence on Fitbit, helping the company to develop new and more advanced wearable devices that offer users a range of advanced features and capabilities. However, as with any major acquisition, there are also concerns about privacy and data security, and it will be important for both companies to ensure that users' personal data is protected and handled responsibly.

On the other hand, Google also benefit from Fitbit's expertise in health and fitness tracking capabilities. The Pixel Watch by Google fully integrates with the Fitbit ecosystem — including a Fitbit app on the Pixel Watch and the ability to sync that data with the Fitbit smartphone app.

Smart Rings' Market Despite the relatively small market share compared to other wearable devices such as smartwatches, fitness trackers, and smart bands, smart rings are gaining popularity as they offer a more discreet, compact or fashionable alternative to traditional wearable devices. They offer a variety of functionalities such as fitness and health tracking, NFC-payment function, and emergency calling capabilities.

Smart rings can be categorized based on the types of in-phone apps that are available to use with them. The first type of smart rings, such as the Motiv Ring [MOV], allows users to connect with Apple Health or Google Fit to save their fitness data. These rings sync with the user's smartphone and provide detailed metrics related to activity, heart rate, and sleep.

The second type of smart rings, such as the Oura Ring and Nimb Smart Ring, have their own dedicated apps that are often available on the Apple App Store or Google Play Store. The Oura Ring app requires a monthly membership subscription and tracks vital signs, such as body temperature, heart rate, and respiratory rate, in addition to sleep and activity monitoring. The Nimb Smart Ring comes with a panic button that sends an emergency alert to first responders and emergency contacts through its dedicated app.

Additionally, smart rings such as the K Ring [KRG] have the NFC-payment function integrated within the ring. The K Ring is powered by Mastercard and benefits from all of Mastercard's security features. It can be locked and unlocked through the app and does not require a pairing smartphone to make payments.

3.3.2 Hardware Fragmentation

Sensors

In the field of wearable technology, one of the key aspects that contribute to the diversity and functionality of wearables is the wide range of sensors they incorporate. Wearable devices come with a set of sensors to capture and measure different aspects of the user's environment and physiology.

A notable research study [ACS16] from 2016 examined over 140 wrist wearables to analyze the prevalence of different sensors. The bar chart Figure 3.2 from the study showcases the distribution of sensor types found in these wearables, shedding light on the most commonly deployed sensors.

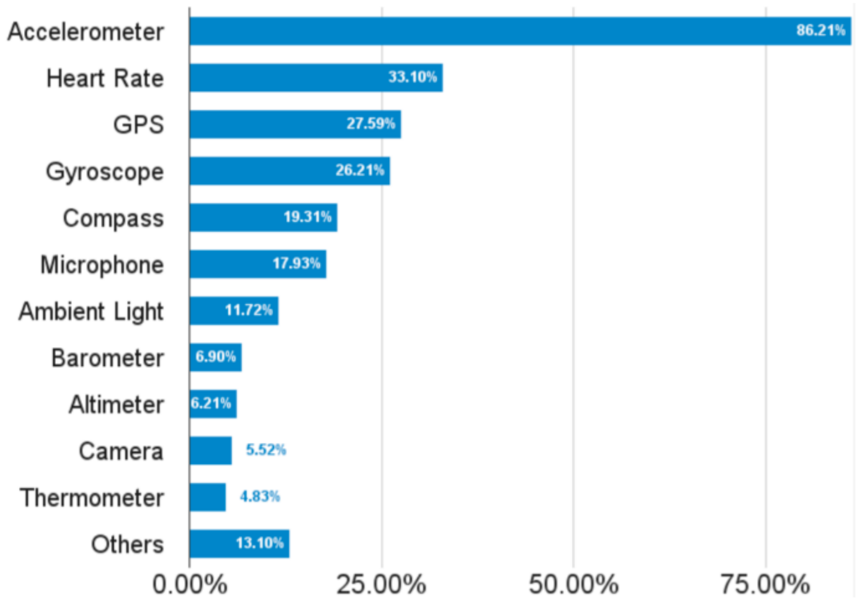


Figure 3.2: Sensors available in wearables from [ACS16]

As depicted by the chart, the most prevalent sensors in wrist wearables were the accelerometer and heart rate sensor. The accelerometer provides insights into movement and orientation, enabling activity tracking and gesture recognition. The heart rate sensor offers real-time monitoring of the user's heart rate, which is valuable for fitness tracking and health monitoring applications.

Other commonly found sensors include GPS for location tracking, gyroscope for rotational movement detection, compass for directional information, microphone for audio input, and ambient light sensor for environmental light measurements. Additionally, wearables may incorporate sensors such as barometers, altimeters, cameras, thermometers, and various other sensors to capture additional contextual data.

It is important to note that sensor precision may vary among different wearable devices, and the accuracy of the collected data can be influenced by factors such as the positioning of the device on the user's body [ACS16]. However, for non-medical purposes, the precision of these sensors is generally considered sufficient [ACS16], making wearables suitable for self-quantification, fitness tracking, and various other consumer-oriented applications.

The diversity of sensors in wearables opens up new possibilities for data collection and analysis, enabling innovative applications and services. The integration of these sensor capabilities within a wearable-vehicle integration platforms would offer opportunities for enhancing the driving experience, personalized services, and advanced data-driven functionalities.

Battery Life

Wearable devices, including smartwatches, smart bands, activity trackers, and smart rings, predominantly rely on rechargeable lithium-based batteries, such as lithium-ion and lithium-polymer variants, due to their advantageous properties. These batteries exhibit high energy density, extended cycle life, minimal self-discharge rates, adaptability in form factor, and comprehensive safety features, which make them ideally suited for the stringent compactness and weight requirements of wearable technologies.

The battery life of wearable devices varies significantly based on their specific features, design elements, and use cases. According to manufacturers' estimates, the Apple Watch Series 7 has a battery life of 18 hours, the Samsung Galaxy Watch4 lasts 24-48 hours, and the Garmin Venu 2 endures up to 11 days. Smart bands, such as the Fitbit Charge 5, Xiaomi Mi Band 6, and Garmin Vivosmart 4, offer battery life ranges of up to 7 days, 14 days, and 7 days, respectively. Activity trackers like the Fitbit Inspire 2, Garmin Vivofit 4, and Withings Move display battery life spans of up to 10 days, 1 year, and 18 months, respectively, with the latter two utilizing non-rechargeable coin cell batteries. Finally, smart rings, including the Oura Ring and Motiv Ring, have battery life estimates of 5-7 days and 3 days, respectively, while the NFC Ring operates without a battery through NFC-enabled devices.

It is important to recognize that these estimates may vary, with actual battery life dependent on factors such as usage patterns, device settings, and environmental conditions. The battery life of wearable devices is influenced by numerous aspects, encompassing device type, operating system, display technology, embedded sensors and features, power management strategies, battery capacity, and alternative charging methods like solar charging. These determinants collectively contribute to the overall battery performance, shaping the user experience and practicality of wearable devices across various applications and settings.

The remaining battery life of a wearable device considerably influences its management and communication with external systems. As the battery capacity depletes, a variety of power-saving strategies may be employed to maintain device functionality. As a result, these strategies can affect the communication frequency and quality between the wearable device and associated integration platforms, as well as the availability of certain features and services offered by these platforms. In the context of automotive applications, safety-related functions reliant on wearable data might necessitate premature disconnection or deactivation to guarantee driving safety and comfort.

Connectivity Options

Wearable devices, including smartwatches, activity trackers, smart rings, and smart bands, incorporate a variety of communication technologies to augment their functionality and connectivity. Notably, wearables exhibit distinctive features in terms of connectivity options as compared to other low-battery IoT devices. While 6LoWPAN, Z-Wave and Zigbee are commonly used for home automation applications, they are typically not employed in wearables due to their limited throughput [ZVB][COW]. Herein, we provide a concise overview of the diverse communication technologies integrated into wearable devices.

Bluetooth Bluetooth is a short-range wireless communication technology that operates on the license-free 2.4 GHz ISM band [Blu06]. Wearable devices often employ Bluetooth to connect to smartphones, tablets, and laptops, allowing them to receive notifications, control music playback, and send data to paired devices. Bluetooth Low Energy (BLE) is a variant of the Bluetooth protocol that consumes less power, making it an ideal choice for wearables that require long battery life.

Bluetooth technology enables communication between wearables and connected cars over short distances. Wearables can be paired with the car's infotainment system or other onboard devices to provide access to media and messaging services, and allow hands-free calling and texting.

Wi-Fi Wi-Fi is a wireless communication technology that operates in the 2.4-2.48 GHz or 4.9-5.8 GHz frequency band, providing high-speed data transfer rates over a wider range compared to Bluetooth [OSK+21]. Wearables that support Wi-Fi connectivity can access the internet directly, stream music, make phone calls, and utilize web-based services.

Wi-Fi can be used to create a local network between wearables and the car's infotainment system or other onboard devices, enabling seamless communication between the two. This can be used to provide access to media and messaging services, and also allow for remote monitoring of the car's diagnostics.

NFC (Near Field Communication) NFC is a short-range wireless communication technology that operates at a frequency of 13.56 MHz [OSK+21]. It enables two devices to exchange data over a distance of a few centimeters. Wearables that support NFC can perform contactless payments and exchange small amounts of data, such as contact information.

NFC technology can be used to provide a simple and secure way for wearables to interact with connected cars, such as unlocking the car's doors or starting the engine.

Cellular Wearables with built-in cellular connectivity operate as standalone devices, allowing them to make phone calls and send text messages without the need for a paired smartphone. Cellular wearables use a mobile network to establish a connection and operate over a broad range, making them ideal for outdoor activities or situations where smartphone connectivity is unavailable.

Cellular connectivity can be used to provide seamless connectivity between wearables and connected cars, even when outside the range of local Wi-Fi networks. This allows for features such as remote starting and unlocking of the car, as well as monitoring of car diagnostics.

GPS (Global Positioning System) GPS is a navigation system that utilizes a network of satellites to determine the location and track movement. Wearables that incorporate GPS functionality can monitor distance traveled, track workouts, and provide location-based services.

GPS technology can be used to provide location-based services and enable wearables to provide real-time updates on the car's location and other important data.

ANT+ ANT+ is a wireless communication protocol that establishes personal area networks (PANs) and operates within the 2.4 GHz frequency band. Its application is primarily found in the realm of fitness trackers, and although conceptually similar to Bluetooth low energy, it is specifically designed for sensor usage [ANT]. Wearable devices equipped with ANT+ technology can effectively communicate with other ANT+-enabled devices, including but not limited to heart rate monitors and pressure monitors.

Wearables with ANT+ technology can transmit data to a car equipped with an ANT+ receiver, allowing for easy and convenient monitoring of the wearer's health and fitness data while driving.

In conclusion, the most common network technologies for wearable devices are Bluetooth, WiFi and cellular, followed by NFC, GPS and ANT+. The choice of communication technology employed by a wearable device depends on its intended use, features, and user requirements. Utilizing a combination of these communication technologies can enable the development of a comprehensive and seamless software architecture for communication between wearables and connected cars.

3.3.3 Software Fragmentation

Platforms and Operating Systems

Nowadays, it is rare for a wearable device to not have an embedded operating system, as an operating system is necessary to manage the hardware and software of the device.

However, there are some simple wearable devices that may not have a full-fledged operating system, but instead use a simple firmware or microcontroller to manage basic functionality.

An example of a wearable device that doesn't have an embedded operating system is the Polar H10 heart rate sensor. The Polar H10 is a chest strap heart rate monitor that is designed to be used during physical activity to measure and record heart rate data. Polar H10 connects and transfers data via Bluetooth and ANT+. The device uses a simple firmware to manage its basic functionality and to transmit heart rate data to other devices, such as smartphones or smart watches. It does not have a full operating system or display, but instead relies on other devices to provide data analysis and visualization. Also, it is compatible with third party apps and devices that support BLE or ANT+ [COP].

The field of wearable computing is dominated by a few operating systems, which include FreeRTOS, LiteOS, Tizen OS, WatchOS, and Wear OS [WEC].

FreeRTOS is a real-time operating system kernel that is widely used in embedded devices and is typically found in devices that emphasize battery life, rapid responsiveness and provide basic functionalities. Manufacturers, including Huawei/Honor, Lenovo, realme, TCL, and Xiaomi, have integrated FreeRTOS into their wearable devices [WEC]. And platforms like Garmin OS, LiteOS, Zepp OS and Amazfit OS utilize RTOS foundations.

LiteOS is the proprietary operating system of Huawei smartwatches. As with most operating systems from Chinese smartwatch manufacturers, LiteOS is a closed operating system and has almost no third-party applications. Even the most popular apps are missing, although a few individual apps, e.g. Spotify, can be installed on some Huawei smartwatches. For the majority of smartwatches using LiteOS, only a few proprietary, specialized apps are available.

Tizen OS is an open-surce operating system jointly developed by Intel and Samsung, for Samsung's smartwatches. When it comes to compatibility with Android smartphones, Tizen OS Watches work seamlessly with Samsung smartphones, but may require a more complex setup with other Android smartphones. Additionally, Samsung has recently moved to Wear OS 3 for its latest smartwatches, including the Samsung Galaxy Watch 4 and Samsung Galaxy Watch 4 Classic. Samsung has stated that it will maintain its efforts on Tizen OS, and as a result, previous models of Samsung watches such as the Galaxy Watch, Galaxy Watch 3, and Active 2 will not receive the Wear OS update [LOA].

WatchOS is Apple's proprietary operating system, specifically designed to run on the Apple Watch. WatchOS integrates seamlessly with the wider Apple ecosystem, especially iOS. The deep hardware-software symbiosis, paired with an extensive application repository, distinguishes WatchOS in the realm of wearable technology platforms.

Wear OS previously known as Android Wear, is a smartwatch operating system maintained by Google, which is also the parent company of the Android Operating System [LOA]. One advantage of the Wear OS operating system is its compatibility with both Android and iPhone smartphones, making smartwatches equipped with Wear OS compatible with both types of devices.

APIs

The assortment of Application Programming Interfaces (APIs) employed in wearable devices offers developers an extensive range of alternatives for devising applications that engage with these devices.

1. **Minimalist Proprietary OS:** A minimalist proprietary operating system (OS) with confined exposed device capabilities merely permits communication between the wearable and the smartphone. This type of API constrains the functionalities that a developer can incorporate within their application.
2. **Minimalist OS with Language-Specific API:** A minimalist OS with restricted exposed endpoints, supporting mini-programs and utilizing language-specific APIs, such as Fetch API [FAI] supported by Zepp OS, for requesting web resources with support for a wide range of request types and options. Although this API offers a degree of adaptability, it remains constrained in terms of functionality and necessitates a companion app to function alongside a device app.
3. **REST API:** A Representational State Transfer (REST) API is employed in Tizen OS, Fitbit OS, and Garmin devices. This type of API furnishes a more comprehensive set of functionalities, enabling developers to devise applications that engage with the device.
4. **SDKs for Wearable and Smartphone Apps:** Software Development Kits (SDKs) for the development of apps on the wearable and its associated smartphone. This permits developers to create bespoke applications for both the wearable device and its companion smartphone app.

5. **Integration with Health Platforms:** APIs that facilitate users connecting with Apple Health or Google Fit to store their fitness data. This integration fosters improved data sharing between third-party apps and these platforms.

Regarding real-time sensor data access, the third scenario utilizing REST API and the fifth scenario employing an alternative health platform frequently do not offer this capability due to several reasons.

- **Privacy Concerns:** These platforms often handle sensitive data, and granting access to real-time data may engender privacy concerns.
- **Preprocessing:** The sensor data undergoes preprocessing by the companion device, possessing greater computing power and enhanced battery capacity before being rendered accessible to a third-party system.

Consequently, real-time sensor data access is typically achievable only in the fourth scenario utilizing SDKs. The employment of SDKs enables developers to devise custom applications that can access real-time sensor data directly from the wearable device. This is feasible because the SDKs offer a more exhaustive set of functionalities, permitting developers to interact with the sensors directly. Nonetheless, the utilization of SDKs also demands increased development efforts and a more profound comprehension of the device and its APIs.

3.4 Summary

The fragmentation problem in the wearable technology industry, as summerized in Table 3.2, poses significant challenges and implications, particularly concerning the development of a unified ecosystem. The lack of interoperability and standardization among different wearable devices, platforms, and software applications hinders the seamless exchange of data and reduces the potential for wearables to integrate with other technologies, limiting their usefulness. This creates significant challenges for vendors and developers, as they must contend with the proliferation of specialized apps and hardware components, resulting in increased complexity, development costs, and maintenance challenges.

Moreover, the fragmentation problem can also lead to concerns regarding data privacy and security, particularly for wearable health devices that collect sensitive health-related data. This underlines the necessity for common standards and protocols for wearables to ensure consistent data protection and security measures.

In the following chapter, we propose an integration platform's software architecture as a potential solution to the fragmentation problem. We explore the technical aspects of the integration platform and how it can facilitate interoperability among various wearable devices and software applications, promoting standardization and convergence.

Aspect	Sub-Aspect	Main Information
Market Fragmentation	Wearable Types (Our research Focus)	Activity trackers: monitor fitness metrics and physical activity. Smart bands: combine fitness features with smartphone notifications. Smartwatches: offer advanced features like apps and communication. Smart rings: provide compact wearable tech with various functions.
	Players	Apple Inc., Google LLC, Xiaomi Corporation, Samsung Electronics Co., Ltd., and more. Smart ring brands: Motive Ring, Oura Ring, Nimb Smart Ring, K Ring.
Hardware Fragmentation	Sensors	Accelerometer, heart rate, GPS, gyroscope, compass, microphone, ambient light, barometer, altimeter, camera, thermometer, and others.
	Battery Life	Varies by device: - Apple Watch 7: 18 hours - Samsung Galaxy Watch4: 24-48 hours - Garmin Venu 2: up to 11 days - Fitbit Charge 5: up to 7 days - Xiaomi Mi Band 6: up to 14 days - Oura Ring: 5-7 days
	Connectivity Options	Bluetooth for short-range communication. Wi-Fi for wireless internet access. NFC for contactless interactions. Cellular for mobile network connectivity. GPS for accurate positioning.
Software Fragmentation	Platform and Operating Systems	Operating systems include FreeRTOS, LiteOS, Tizen OS, WatchOS, Wear OS. Each OS offers different features and capabilities.
	APIs	Minimalist Proprietary OS: Strong limitations on exposed functionality. Minimalist OS with Language-Specific API: Restricted exposed endpoints; uncommon. REST API: Comprehensive functionalities. SDKs for Wearable and Smartphone Apps: Custom app development. Integration with Health Platforms: Enhanced data sharing.

Table 3.2: Summary of the fragmentation problem of wearables

4 An Architecture for Wearables-vehicles Communication

In this chapter, we propose a high-level software architecture aimed at enhancing the communication between wearables and connected vehicles. Our proposed architecture is designed to alleviate the problem of fragmentation, which was discussed in detail in chapter 3, and promote compatibility across diverse devices and platforms. The proposed solution facilitates efficient data exchange between wearables and vehicles and also provides a scalable, adaptable framework that can be easily extended to support additional manufacturers and devices in the future. The aim of this chapter is to present a comprehensive and robust solution for wearables-vehicles communication, which can lay the foundation for a more connected, intelligent, and personalized driving experience.

4.1 A Solution to the Fragmentation Problem

This section outlines an approach aimed at integration by dealing with each aspects of the fragmentation problem in Chapter 3.

The proposed approach involves the use of a middleware that provides a unified interface for interacting with the connected vehicle's ecosystem. To illustrate this approach, we present a UML component diagram that treats both the wearables and the connected vehicle's ecosystem as black boxes. The expected interaction between these systems is depicted in figure 4.1.

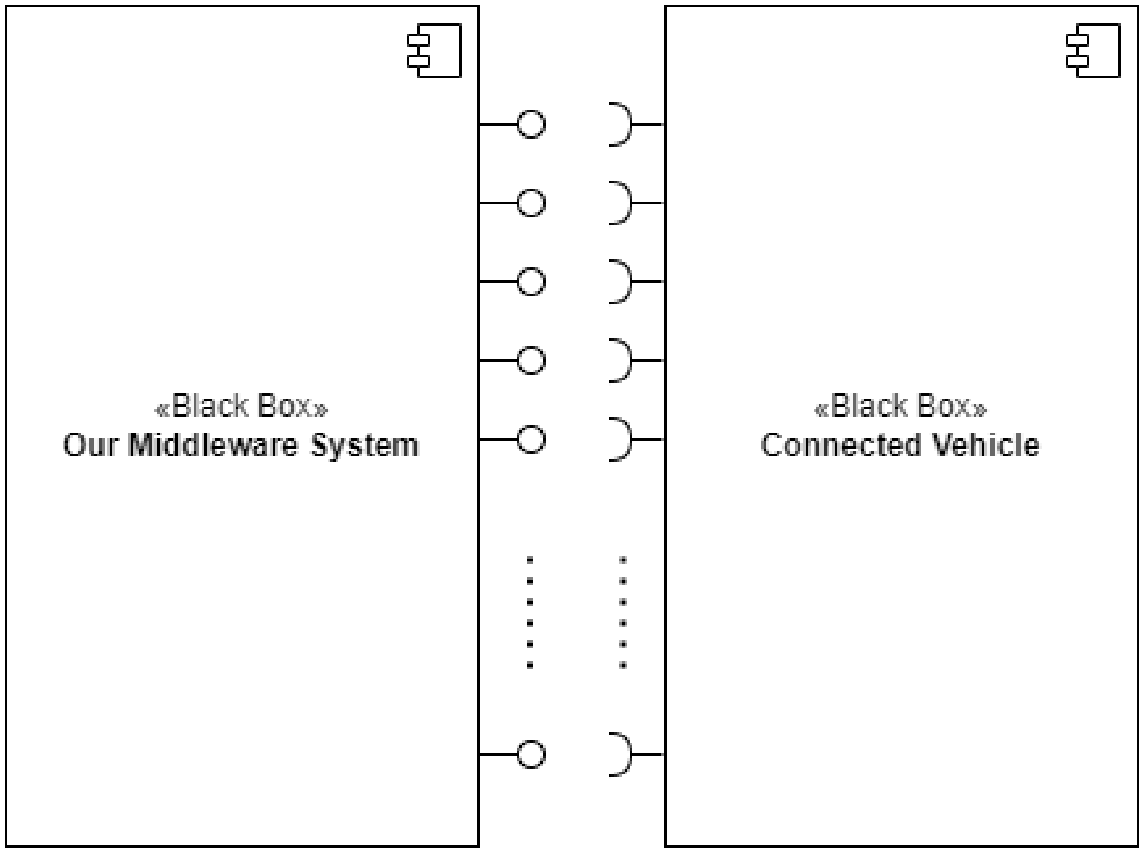


Figure 4.1: Interfaces for interaction between our and the vehicle's system

In order to address the fragmentation problem of wearables, as discussed in Chapter 3, it is necessary to develop a software system that can handle the various aspects of this problem. This section presents the key elements of the proposed software system, which work together to resolve the fragmentation issue. A comprehensive overview of these elements and their respective mappings to the sub-problems they aim to solve is presented in Table 4.1.

4.1.1 Essential Elements

A concise description of each element presented in Table 4.1 is outlined below:

Abstract Interface The common abstract interface for wearables encompasses the necessary functionality, such as connecting, disconnecting, transmitting data, and issuing device-specific commands. This interface serves as the basis for all device interactions.

Device Plug-ins Device plug-ins for each supported wearable implement the common abstract interface. These plug-ins facilitate communication with the devices, abstracting the details of individual manufacturers and device functions.

Element	The Fragmentation Problem					
	Market	Hardware			Software	
		Sensor	Battery	Connectivity	OS	API
Abstract Interface	*					
Device Plug-ins	*				*	
Updater	*				*	
Service Mapper		*				*
Device Management			*			
Communication Layer				*		
Device Registry	Usability					

Table 4.1: Proposed essential elements as a solution to address the fragmentation problem

Device Manager The device manager component is responsible for managing the lifecycle of devices, including discovering, connecting, and disconnecting devices. The device manager cooperates with the device plug-ins to interact with the devices in a uniform manner.

Communication Layer The communication layer abstracts the specific communication methods used between the vehicle and the devices (e.g., Bluetooth, Wi-Fi, or cellular). This layer allows the system to support multiple communication methods and adapt to new methods in the future without necessitating significant modifications.

Device Registry The device registry component stores the supported devices and meta data needed for their corresponding device plug-ins. This registry enables simple and rapid support for additional manufacturers and devices by allowing the system to dynamically load device plug-ins at runtime.

Service Mapper Drawing on the comparison presented in Table 4.2, and without loss of generality, we assume that the services supported by the connected vehicle follow a hierarchical structure, as shown in 4.2. In order to facilitate the integration of wearable services into this structure, a service mapper component is employed. The service mapper analyzes the hierarchical structure of the services supported by the connected vehicle's ecosystem and identifies the appropriate levels and categories within this structure where wearable services can be mapped. Through the creation of a mapping between the wearable devices and their corresponding services, the service mapper ensures that the wearable services are mapped to the appropriate locations within the hierarchical structure.

Updater The updater modifies the common abstract interface and device plug-ins to account for the hierarchical structure of the supported services. This involves updating method signatures, adding new methods, or adjusting data structures to facilitate proper mapping and communication with the connected vehicle's ecosystem.

	Hierarchical Structure	Flat Structure
Pros	Presents a clear and consistent structure for mapping services based on functionality and dependencies [HDVL03]	Offers enhanced flexibility and adaptability to accommodate new and rapidly-evolving services [RIM07]
	Facilitates comprehension of relationships between services and ensures uniformity in naming and design	Provides a more fluid approach to mapping services based on functional resemblances and usage patterns
	Naturally organizes services according to required levels of control and complexity for automated driving	Facilitates cross-functional services by enabling access and sharing of services across diverse functional domains
	Supports automated tools for service mapping	Allows for manual mapping of services, offering greater flexibility and customization for specific use cases
	Aids in establishing a coherent and systematic structure for mapping services based on security requirements	
	Simplifies the implementation of security policies for various services or service groups	
Cons	Might exhibit inflexibility when incorporating new or rapidly-evolving services that do not align with the existing structure	May result in challenges maintaining consistency in the naming and design of services [RIM07]
	Could lead to functional isolation due to separated development efforts, limiting support for cross-functional services	Could impede understanding of relationships between services and their interdependencies [SRPA07]
	May hinder the identification and accessibility of specific services deeply nested within the structure	Could give rise to a large number of services that are cumbersome to manage [SRPA07] and access
	Could pose challenges in managing and updating security policies for specific services or service groups	Could result in an abundance of security policies that are demanding to manage and update

Table 4.2: Comparison of hierarchical and flat structures for services: Advantages and disadvantages

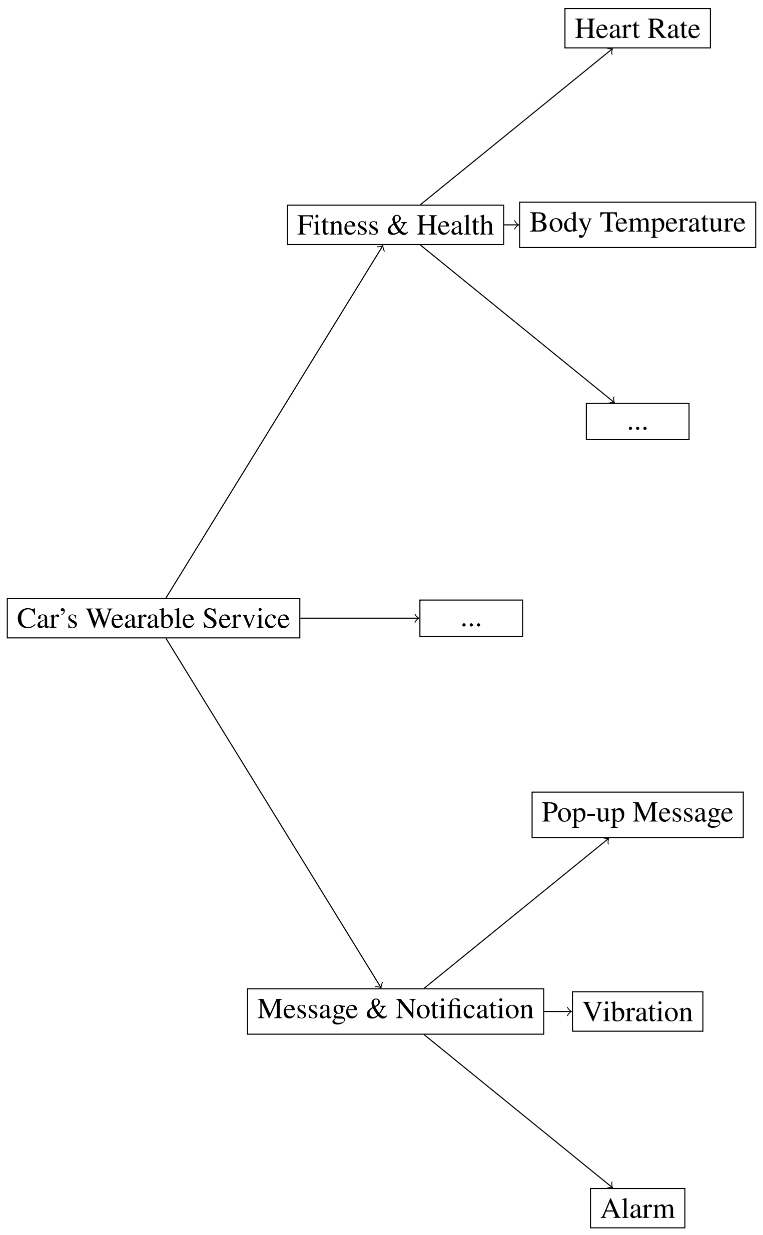


Figure 4.2: A snippet of an example service ontology adopted by the connected vehicle system

4.2 The Overall Architecture

In the preceding section, we outlined the key elements of the software system, collaboratively addressing the challenge of fragmentation. We now introduce the component diagram depicted in Figure 4.3, illustrating the overall architecture. Within this diagram, several key elements have been consolidated into single components, driven by the following rationales:

- Emphasis on Cohesion: By presenting these elements as unified components, we underscore their shared purpose and collaborative function towards achieving a specific objective.
- Focus on High-Level Concepts: Given that intricate internal interactions may not be imperative at this stage, combining several elements into a single component allows us to center our attention on the higher-level concepts and interactions.
- Simplified Representation and Visual Clarity: Consolidating elements enhances both the simplicity of representation and the clarity of visualization.

Subsequent sections will delve into the particulars of each individual component, providing an in-depth exploration of their functionalities and interactions.

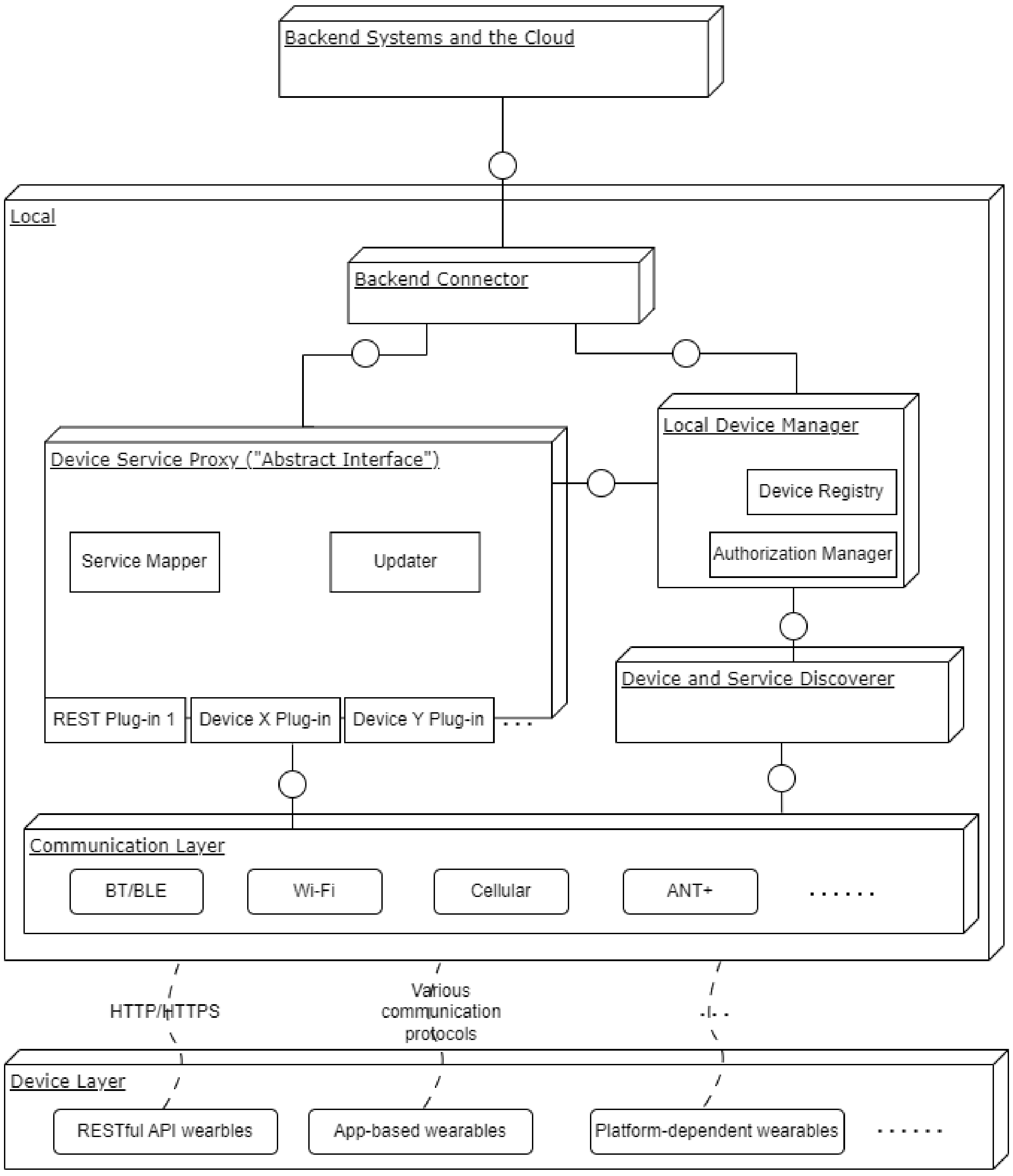


Figure 4.3: The overall architecture with essential components

4.3 The Device Layer

The device layer operates as an abstraction for wearable devices, with this abstraction being established based on the categorization of diverse APIs in the wearable fragmentation problem, as discussed earlier. This abstraction revolves around the last three API types, as the former two exhibit either considerable limitations in terms of exposed functionalities, which are not directly relevant to wearable device integration, or rely on relatively uncommon language-specific Web APIs.

Hence, we categorize devices into three distinct groups, each corresponding to a specific API type:

- Type 1: RESTful API wearables,
- Type 2: App-based wearables, and
- Type 3: Platform-dependent wearables.

For App-based devices, it is also important to consider the available communication protocols. In the subsequent sections, we undertake a examination of communication protocols tailored to Android Wearables as an initial exploration into wearable communication protocols. The Android ecosystem's reputation for openness often translates to a wider range of communication options and protocols for developers. This diversity in options makes Android Wearables an compelling subject of investigation. Additionally, Android Wearables hold a significant portion of the market share. Thus, a dive into their communication protocols can provide insights into a substantial segment of wearable technology.

Analysis of Android Wearables Communication Protocols The Android SDK supports a number of communication protocols that can be used for building applications.

One such protocol is HTTP/HTTPS, which is widely used for communication over the web. This protocol can be used to send and receive data between wearable devices and the connected vehicle, making it a useful choice for implementing a bi-directional communication channel.

Another protocol that can be used for this purpose is Bluetooth, which allows devices to communicate wirelessly over short distances. Bluetooth can be used for a variety of applications, including transferring files and connecting to wearable devices.

NFC is another protocol that can be used for communication between wearable devices and the connected vehicle. NFC allows devices to communicate over short distances by touching or bringing them close together, and is commonly used for mobile payments and contactless ticketing.

In addition to these protocols, there are also third-party libraries available for implementing communication over protocols such as XMPP and WebSocket. XMPP is a messaging protocol used for real-time communication and messaging over the internet, while WebSocket provides a full-duplex communication channel over a single TCP connection.

MQTT is a lightweight messaging protocol that is commonly used in Internet of Things (IoT) applications. The Android SDK does not have built-in support for MQTT communication, but there are third-party libraries that can be used to implement MQTT communication in Android apps. One popular library is Eclipse Paho.

gRPC is a high-performance remote procedure call (RPC) framework developed by Google. gRPC is designed to be language- and platform-agnostic, and it supports a variety of programming languages, including Java and Kotlin. The gRPC Java library can be used in Android apps to communicate with gRPC-based services.

Table 4.3 includes links to the official Android documentation for the above discussed protocols that are supported by the Android SDK, and links to the third-party libraries for the protocols that are not natively supported. This table may not be exhaustive and there may be other communication protocols available for use in Android (wearable/smartphone) applications.

Table 4.3: Communication protocols for Android applications

Protocol	Description	SDK Support	Third-Party Libraries
HTTP/HTTPS	Used for communication over the web	Yes	N/A
Bluetooth	Used for wireless communication over short distances	Yes	N/A
NFC	Used for communication over short distances	Yes	N/A
SIP	Used for voice and video communication over the internet	Yes	N/A
XMPP	Used for real-time communication and messaging over the internet	No	Smack
WebSocket	Provides a full-duplex communication channel	No	OkHttp, Java-WebSocket
MQTT	Lightweight messaging protocol commonly used in IoT applications	No	Eclipse Paho
gRPC	High-performance remote procedure call (RPC) framework	No	gRPC Java library

4.4 Comprehensive Device and Service Discovery Mechanism

Considering the diversity of wearables, their capabilities, and their interoperability with platforms such as Apple Health and Google Fit, a comprehensive and multifaceted approach for device and service discovery will be required.

4.4.1 Wearable Device Types and Interoperability

Firstly, a understanding of the wearable types in Section 4.3 are essential to design the appropriate mechanism for each:

Type 1: RESTful API wearables

Type 1 wearables provide RESTful APIs, which are stateless, cacheable, and have a uniform interface. Discovery of these types of wearables can be achieved through various mechanisms such as network scanning, or Bluetooth/Wi-Fi discovery protocols.

For example, in the case of Bluetooth-enabled wearables, the integration platform can use Bluetooth scanning to detect nearby devices and retrieve their unique identifiers. This information can then be stored in a device registry or directory, allowing the integration platform to maintain a record of the discovered devices.

Similarly, for Wi-Fi-enabled wearables, network scanning techniques can be employed to identify devices connected to the local network. The devices' IP addresses or other network identifiers can be stored in the registry for subsequent communication and interaction.

Type 2: App-based wearables

Type 2 wearables require the implementation of an additional app on the wearable device or/and the paired smartphone. The discovery process for these wearables can be facilitated through specially designed app-to-app communication protocols or customized interprocess communication (IPC) mechanisms. These protocols and mechanisms need to be tailored to the unique characteristics and functionalities of the wearables, ensuring seamless and secure communication between the wearable app and the integration platform. By developing dedicated communication solutions, the discovery process can be optimized for efficient device detection, service identification, and data exchange between the wearables and the platform. This approach enables effective integration and enhances the interoperability of the wearables within the connected ecosystem.

Type 3: Platform-dependent wearables

Type 3 wearables establish connections with prominent health platforms such as Apple Health or Google Fit. To seamlessly integrate these wearables, the connected vehicle platform can leverage the APIs provided by these health platforms. Accessing health data from these platforms necessitates user authorization, in compliance with stringent privacy regulations

4.4.2 Discovery Mechanisms

In the era of rapidly advancing technology, the importance of effective discovery mechanisms cannot be overstated. Connecting wearables to the vehicle's automotive infotainment system necessitates a efficient process for both identification and integration. To cater to this varied landscape of wearables, discovery mechanisms can be broadly grouped based on their triggering factors, their nature (manual or automatic), their underlying protocols, the types of services they cater to, and the mechanisms they use to maintain their registry and ensure continued security.

In the following, we explore each of these facets in detail.

Triggering Initial Discovery

The initiation of the discovery process could be triggered by a range of events. For instance:

- Vehicle Start: When the vehicle's infotainment system starts up (e.g., when the car is turned on), it could initiate a device discovery process.
- User Prompt: The system could also provide an option for the user to manually trigger a device discovery process. This could be useful in scenarios where a new device has been brought into the vehicle after the initial startup.
- Periodic Intervals: The system could be designed to automatically initiate a discovery process at regular intervals, ensuring that any new devices brought into the vehicle are detected promptly.

User-Initiated VS Automatic Device Discovery

- User-Initiated Discovery: This mechanism is suitable for all types of wearables. In this scenario, the user manually connects the wearable to the infotainment system, either by entering the device's IP, its vendor-specific user ID, or by selecting it from a list of nearby Bluetooth devices. This method offers high security, as it requires explicit user consent.
- Automatic Discovery: Discovery can be automated using protocols like Bluetooth Low Energy (BLE) and Wi-Fi Direct. With BLE, the wearable can periodically advertise its presence for the infotainment system to detect. In the case of Wi-Fi Direct, the wearable and infotainment system can negotiate a connection without the need for a central Wi-Fi access point.

Multi-Protocol Support

The inherent diversity in communication protocols among wearable devices, extending but not limited to support for Bluetooth, Wi-Fi, Cellular, and ANT+, necessitates the adoption of a multi-protocol approach in the discovery process. The discovery service ought to be founded on an abstract communication layer capable of handling multiple protocols. Using this layer, it should be possible to launch discovery processes on various interfaces (Bluetooth, Wi-Fi, Cellular, ANT+) either simultaneously or sequentially.

Below, we offer a concise overview of some protocol-specific discovery mechanisms:

- BT/BLE: For Bluetooth or Bluetooth Low Energy devices, it is convenient to utilize the standard BT/BLE device discovery process. The system broadcasts a discovery request, to which the wearable responds if set in the discoverable mode. Services can be discovered via the Bluetooth SDP or the GATT services for BLE devices.
- Wi-Fi: Devices supporting Wi-Fi can be discovered using mechanisms like mDNS or SSDP for local network discovery, depending on the device capabilities and the network configuration. Service discovery can be facilitated through standards like Web Services Dynamic Discovery (WS-Discovery) or DNS-SD, or via proprietary APIs provided by the device.
- Cellular: Cellular-connected wearables can be more challenging to discover directly since they often link to broader networks rather than local ones. These wearables might offer APIs for service discovery, potentially employing RESTful interfaces over HTTP/HTTPS. In scenarios like this, a registry of recognized devices might be beneficial, where each device reports its status periodically.
- ANT+: ANT+ is a wireless protocol specifically designed for monitoring sensor data and is commonly used in health and fitness wearables. Device discovery can be done by broadcasting a request on a shared ANT channel. For service discovery, ANT+ defines device profiles that specify the data format and communication requirements for different types of devices, such as heart rate monitors, foot pods, and power meters.

Type-based Service Discovery

Once device discovery is accomplished, the next step is to discover the services offered by the wearable. The method of discovery is influenced by the type of wearable:

- Type 1 (RESTful API wearables): For wearables equipped with a RESTful API, services can be identified by sending queries to this API. Wearables might offer a machine-readable depiction of their services using standardized formats such as the Web Application Description Language (WADL) or Swagger/OpenAPI. However, there are instances where vendors' APIs do not align with such conventions. In these cases, it becomes crucial to maintain a repository of these APIs to facilitate service discovery and subsequent interactions. Thus, for service discovery, one would either query the wearable's API and refer to its machine-readable service description (if available) or consult the maintained repository for the relevant API information.
- Type 2 (App-based wearables): Inter-Process Communication (IPC) mechanisms or app-to-app communication protocols can be utilized to communicate with the companion application for these wearables. The services can be discovered by querying this app, and perhaps by utilizing a specific service discovery protocol within the app itself.
- Type 3 (Platform-dependent wearables): Service discovery for these devices mainly relies on the APIs provided by the health platforms to which they are connected. These APIs typically require authentication and might offer limited service discovery, based on the platform's privacy policies. For service discovery, the infotainment system needs to authenticate itself

with the platform’s API. Subsequently, the user would need to explicitly allow the system to access each category of health data. This authorization could be granular, necessitating distinct permissions for different data types such as heart rate, step count, and sleep details.

Service Registration and Cataloging

Once services are discovered, they should be cataloged for future reference.

A service registry should keep a list of services from each wearable device and update it whenever new services emerge or existing ones are removed. This registry can be stored locally on the infotainment system. Given that it may store sensitive information, such as user credentials for REST API calls, it’s imperative to secure it to safeguard user data.

Continuous Discovery and Re-authentication

Moreover, the system should periodically re-scan for devices and services, updating its information and requiring re-authentication as necessary:

- Periodic Rescanning Regularly repeat the discovery process to find new devices or services. This is especially important in dynamic environments where devices may frequently enter or exit the range of the infotainment system.
- Re-authentication Periodically prompt the user for re-authentication. This maintains security by ensuring continued user consent and protects against situations where a paired device may have been compromised.

In summary, this section has explored a range of methods and aspects for device and service discovery, highlighting that no single method is universally applicable. We’ve presented a holistic solution that embraces the variety of wearable devices, prioritizes user authentication, and remains adaptive to ever-changing environments. Implementing this approach would be complex and would necessitate careful management of factors such as user privacy, secure communication, and efficient resource allocation. Moreover, it would need to stay flexible to adapt to evolving wearable technologies and shifts in user preferences.

4.5 Device Manager

After the discovery process concludes, the information of the discovered devices is transmitted to a core component named the Device Manager. The Device Manager’s primary role is to manage the lifecycle of wearable devices. This lifecycle management includes initiating the discovery process, establishing connections with detected devices, and overseeing the disconnection process.

4.5.1 Device Registry

The Device Manager incorporates a sub-component called the Device Registry. This registry is responsible for storing details of both known and unknown devices, serving as an record of device specifications [MKK22]. At its core, it contains a specification storage of known devices, a repository that securely holds metadata about the devices, the responsible plug-ins, and their services.

As illustrated in Figure 4.4, the specification storage forms an essential data source for the Device Service Proxy. This proxy utilizes data from the Device Registry to carry out tasks such as facilitating the invocation of device services, handling service-related errors, and conducting data format translations. This ensures compatibility between various wearable devices and the platform.

Given the wide-ranging nature of wearable devices and their services, both the Device Manager and Device Registry are crucial in effectively handling this diversity. By segregating the discovery process from service invocation — overseen by the Device Service Proxy — the architecture provides a flexible mechanism for device and service management. It also enhances scalability, priming the platform for future advancements and a growing number of devices.

4.5.2 Authorization Manager

Another sub-component of the Device Manager is the Authorization Manager. This component manages the authorization-related details of each wearable device and can automatically refresh authorizations.

For instance, to oversee the OAuth 2.0 process employed by some wearables, the Authorization Manager takes care of tasks associated with OAuth 2.0. This includes generating authorization requests, redirecting to the relavent authorization server, managing responses, storing access tokens, and refreshing tokens as needed [AUN].

Certain user data, such as the scopes for which an access token was issued, can be stored to avoid making unnecessary calls to the authorization server for data a user has consented to share. Moreover, a refresh token can be saved to obtain a new access token without requiring user intervention.

If a token expires, the Authorization Manager can use the stored refresh token (if available) to seamlessly fetch a new access token. If necessary, it can also prompt the user to re-authorize the application. This provides a balance between user convenience and the necessary security precautions.

Nevertheless, when storing sensitive information such as refresh tokens, it’s vital to adopt additional security measures to ensure safe handling and protection. These measures are essential for safeguarding sensitive information and maintaining the integrity of the system.

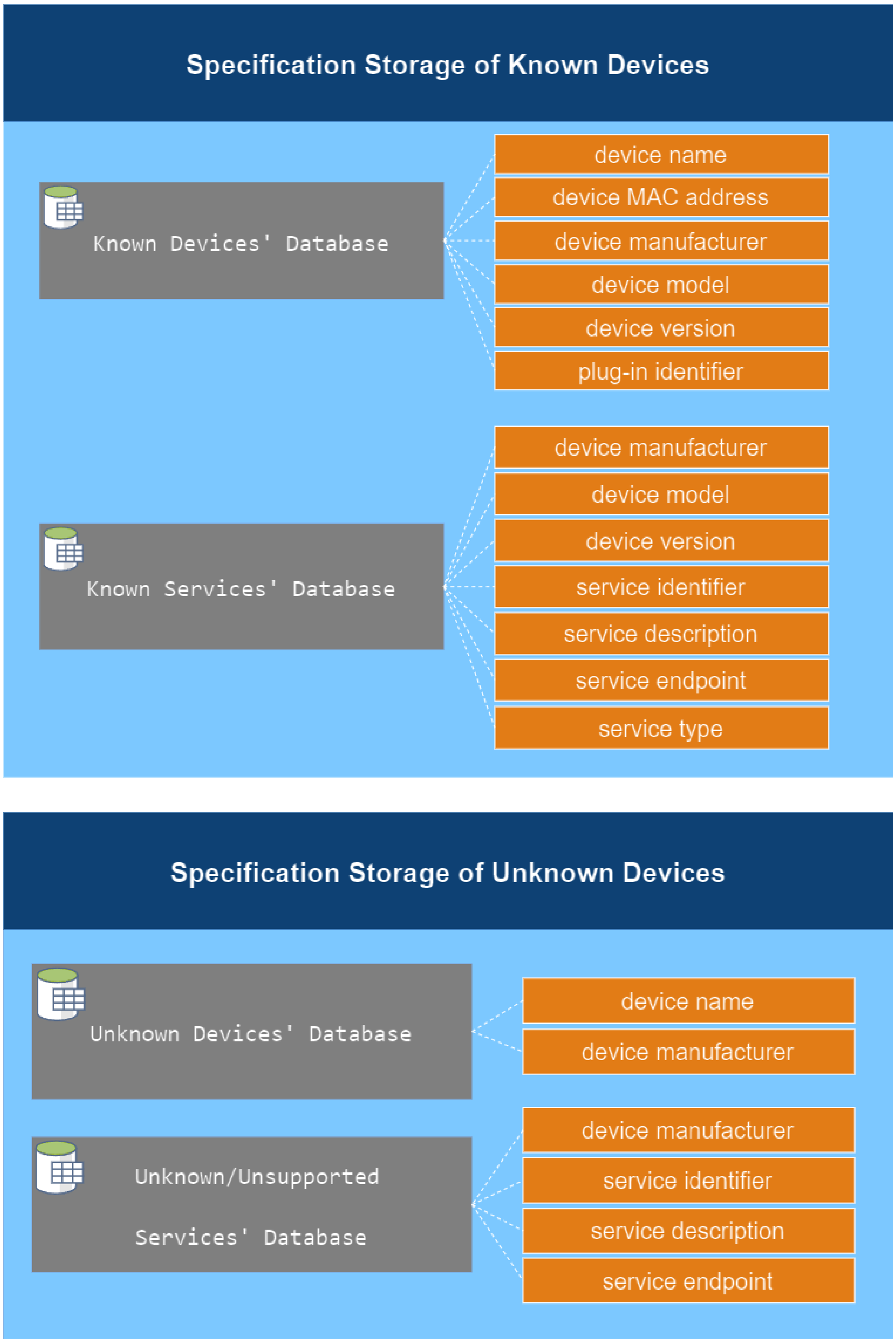


Figure 4.4: Specification storage for known/unknown devices (based on [MKK22])

4.6 Device Service Proxy

Device Service Proxy is a central part of the proposed architecture, for which the concept of common protocol and the adapter pattern is adopted.

A common protocol is crucial, to integrate wearables with different protocols into a unified integration platform. The common protocol acts as a bridge between the diverse protocols used by wearables and our middleware system, enabling seamless communication and data exchange. This section outlines some major design and implementation considerations for the common protocol, leveraging the adapter pattern and protocol-specific adapters.

4.6.1 The Common Protocol

Designing a common protocol involves defining a standardized message format and communication conventions that can accommodate the variations and requirements of different wearables. Aspects such as message structure, data representaion and metadata should be considered:

Attribute	**Contribution**
Message ID	Provides a unique identifier for each message for tracking
Time Stamp	Records the time when the data was generated or transmitted
Command and Control	Enables sending commands and states to devices and controlling their behavior
Device ID	Uniquely identifies a device for identification and routing
Device Type ID	Identifies the type of device for categorization and compatibility
Service ID	Identifies the specific service or functionality being accessed
Data Type	Specifies the type of data being transmitted or exchanged
Payload	Contains the actual data being transmitted or exchanged
Metadata	Provides additional information about the data, such as its source, format, or version
Security Credentials	Includes authentication tokens or cryptographic keys to ensure secure communication and access control
Quality of Service	Defines the level of reliability, delivery guarantees, or priority assigned to the data
Error Handling Code	Facilitates error detection, reporting, and appropriate actions

Table 4.4: Critical data structure attributes for integration

Message Structure The structure of the messages exchanged between the wearables and the middleware is to be defined. This structure should capture the necessary data fields, metadata, and any additional information required for seamless integration, among which the device ID, device type ID, service ID are most important. In Table 4.4 we list some of the critical attributes and their contributions for seamless integration.

Data Representation After decision on a common data structure, we need to specify the format and encoding of data within the messages. Considering the most common infotainment systems in connected vehicle are based on QNX, Linux, or Android, we could chosen among JSON, XML, YAML, or Protobuf, for the most suitable data representation formats based on factors like simplicity, extensibility, and interoperability. We conducted a short comparison between four possible data representation formats as shown in Table 4.5.

	XML	**JSON**	**YAML**	**Protobuf [PRB]**
Format	Tags and attributes	Key-value pairs	Indentation-based data serialization	Binary data serialization
Readability	Human-readable	Human-readable	Human-readable	Binary representation
Complexity	Can represent complex structures	Supports nested objects/arrays	Supports nested objects/arrays	Supports nested objects/arrays
Extensibility	Supports namespaces, schema	No inherent schema support	No inherent schema support	Supports schema definition
Compatibility	Widely supported	Widely supported	Widely supported	Language-specific libraries; platform-neutral
Performance	Parsing overhead, larger file size	Parsing efficiency, compact size	Parsing efficiency, compact size	Efficient binary serialization

Table 4.5: Comparison of XML, JSON, YAML, and Protobuf for data representation

Metadata and Context Considering the special context of connected cars, we may need additional attention when incorporating metadata and contextual information within the messages for data being transmitted. This may include information about the wearable device, service, data, time stamps, as already covered in Table 4.4.

Furthermore, contextual information such as the in-car position/roll of wearable wearers could be relevant and benefit complex interaction not only between the wearable and the connected vehicle, but also among wearables based on specified and/or customized rules.

In the following, we take REST API for example, which illustrates how the position information could be included in the endpoints to allow access control based on the roll of the wearer in the vehicle.

```
GET /devices
GET /devices/{position_id}/{device_id}
GET /devices/{position_id}/{device_id}/services
GET /devices/{position_id}/{device_id}/services/{service_id}/data
POST /devices/{position_id}/{device_id}/services/{service_id}/control
```

Possible implementation for *position_id* could be the combination of row and column of seat, with 11 being the id for the driver, 00 meaning unspecified and 01 meaning for all.

Example Code: Common Protocol Message Structure

The following example code illustrates a common protocol message structure for wearable-vehicle integration. It includes attributes such as the wearable ID, timestamp, and other relevant fields, as discussed in Table 4.4. This structure ensures a standardized representation of wearable data, facilitating seamless interpretation and processing by the middleware. Some fields may not have values specified in the example and can be filled in during the subsequent data interpretation process.

```
{
        "message_id": 0023152,
        "timestamp": "2023-06-01T21:15:00",
        "wearable_id": "xyz123",
        "wearable_type_id": "abc_v1",
        "command": "start_collection",
        "service_id": "dataFromWearable",
        "payload": {
                "type": "body_temperature",
                "value": 36.5,
                "unit": "C"
        },
        "data_type": ,
        "credentials": ,
        "QoS": ,
        "error_code": ,
        "metadata":
}
```

The message structure captures key attributes necessary for the integration of wearables with the vehicle. The “wearable_id” field uniquely identifies the wearable device, while the “wearable_type_id” field specifies the type or model of the wearable. The “timestamp” field indicates the time at which the data is recorded, enabling temporal analysis and synchronization.

The “command” field represents a specific command or action to be performed by the system, such as starting data collection from the wearable. The “service_id” field identifies the specific service or functionality associated with the data being transmitted.

The “payload” field contains the actual data payload from the wearable, in this case representing body temperature. The “type” field specifies the type of data, the “value” field contains the measured value, and the “unit” field indicates the unit of measurement.

Additional fields such as “data_type”, “credentials”, “QoS” (Quality of Service), “error_code”, and “metadata” can be included based on specific requirements and considerations. These fields provide flexibility and extensibility to accommodate different use cases and provide necessary context or information related to the data.

4.6.2 Adapter Pattern for Protocol Conversion

The adapter pattern plays a crucial role in the implementation of the common protocol. Protocol-specific adapters act as intermediaries between the wearables and the middleware, facilitating the application-level conversion [CL90] of different protocols to the common protocol. Steps in the process for protocol conversion are outlined in Figure 4.5.

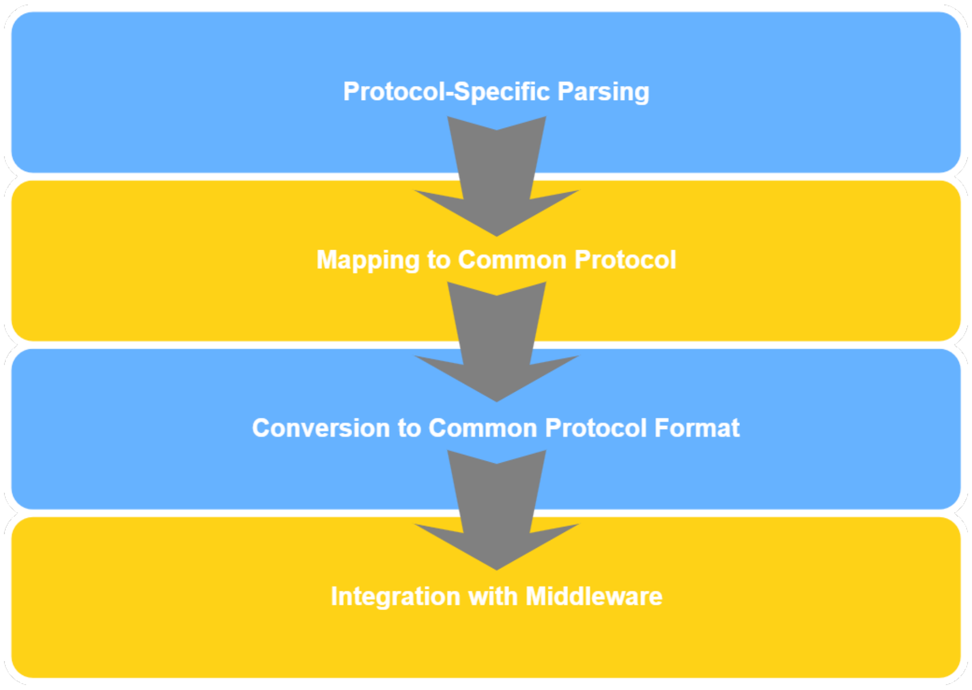

Protocol-Specific Parsing
Mapping to Common Protocol
Conversion to Common Protocol Format
Integration with Middleware

Figure 4.5: Process for protocol conversion

Protocol-Specific Parsing Within each adapter, the incoming messages from the wearables are parsed according to their specific protocols. This involves extracting the relevant data fields, such as sensor readings, device information, and any other protocol-specific details. For example, if a wearable uses a proprietary binary protocol, the adapter would need to parse the binary data and extract the required information. This approach offers a significant benefit as wearable vendors can provide their own adapters that conform to the common protocol of the integration platform, while still maintaining the confidentiality of their proprietary protocols and business secrets.

Mapping to Common Protocol After parsing the protocol-specific messages, the extracted data fields are mapped to the corresponding fields in the common protocol message structure. This step ensures that the data fields are aligned and can be seamlessly integrated with other components of the system. For instance, if the wearable provides heart rate data, the adapter maps the heart rate value to the corresponding field in the common protocol message structure.

Conversion to Common Protocol Format Once the mapping is complete, the adapter converts the mapped message into the agreed-upon format of the common protocol. This format, such as JSON or Protobuf, ensures consistency and interoperability across different components of the integration platform. The conversion process transforms the data from the specific protocol format to the standardized format defined by the common protocol.

Integration with Middleware The converted messages are then forwarded to the middleware for further processing and integration with other services. The middleware is designed to understand and handle the common protocol, enabling seamless integration of data from different wearables. For example, the middleware may aggregate data from multiple wearables, perform data analysis, and trigger appropriate actions based on predefined rules and logic.

By following these steps, the adapter bridges the gap between the specific protocols used by wearables and the common protocol supported by the middleware. This allows for efficient and consistent data communication and integration within the wearable-vehicle integration system. The adapter's role is crucial in ensuring that the diverse data from wearables can be seamlessly processed and utilized by the middleware and other components of the system.

Beyond the fundamental function of bridging different protocols, this middleware framework brings forth a phenomenon referred to as the *Emergence Effect*.

The Emergence Effect At its core, the emergence effect refers to the middleware's ability to unlock and integrate additional functionalities, drawing on the combined capabilities of all connected wearables. Some of the potential advantages include:

- Intelligent Environment Adaptation: The middleware, acting as a central hub, gathers data and user requests from various drivers and passengers. Using its advanced data analysis and processing capabilities, it intelligently tunes the car's environment settings to cater to individual preferences. For instance, it can dynamically adjust air-conditioning and airing modes based on the collected body temperatures of each occupant, ensuring optimal comfort for each individual in the vehicle.
- Contextual Notifications: The middleware can generating contextually relevant notifications and alerts based on real-time data sourced from wearables. As an illustration, consider a scenario where a caretaker sets a specific health threshold for someone left in the car. Should the wearable data indicate any abnormalities, such as a spike in heart rate or drastic temperature changes, the middleware instantly notifies the caretaker's wearable, prompting them to check immediately.

- Adaptive Assistance: The middleware also aids drivers by offering dynamic support. If a driver's wearable data signals fatigue or drowsiness, the middleware can trigger safety alerts, suggest breaks, or adjust vehicle systems to mitigate potential risks

Following the emergence effect and its role in amplifying the in-car experience, it's pertinent to dive deeper into the backbone of these functionalities: data models and their interpretation. The richness of the middleware's capabilities is deeply rooted in how data is structured, stored, and interpreted. Proper data modeling is critical, as it ensures that the vast amounts of information flowing from wearables are not just aggregated, but also understood. This comprehension is essential to translate raw data into actionable insights and responsive interventions. Let's delve into the intricacies of data models and their pivotal role in data interpretation.

While the middleware's emergence effect showcases the application of integrated data in real-world scenarios, the foundation of these capabilities lies not just in collecting the data, but in understanding, interpreting, and homogenizing it for consistent application.

4.6.3 Data Models and Data Interpretation

Due to the inherent differences in data generated by wearables from various vendors, disparities arise in data models, identification methods, vocabularies, and temporal availability. These disparities can result in discrepancies in granularity and units of measurement and necessitate a homogenization process to bring the data into a unified format [ACS16].

Moreover, intelligent algorithms, tailored to the characteristics of the collected data, may impose specific requirements on the data model. This is where an interpreter or mapping mechanism becomes indispensable, bridging the gap between the diverse collected data and the algorithmic requirements. The interpreter facilitates the conversion, matching, and transformation of data, ensuring compatibility and enabling the algorithms to operate seamlessly on the collected data. By harmonizing the data representation and making it compatible with the algorithms' specific expectations, the interpreter enhances the effectiveness and accuracy of the subsequent processing and analysis.

The homogenization process, in conjunction with the interpreter, is foundational to standardizing the wearable data and making it consistent across different devices and vendors. It ensures that the data conforms to a standardized convention, allowing for seamless integration, interpretation, and analysis within the wearable-vehicle integration platform. Furthermore, it facilitates the effective deployment of intelligent algorithms, enabling them to leverage the collected data effectively and generate meaningful insights.

Pipes and Filters

Combining the "Adapter Pattern" with the "Pipes and Filters" architectural style can be a beneficial approach to handle the complexities and variations in data processing.

The "Adapter Pattern" adapts various wearable protocols to a unified protocol employed by the integration system. It focuses on the communication aspect, ensuring that data can be transmitted and understood between the wearables and the middleware.

On the other hand, the “Pipes and Filters” architectural style is concerned with the processing and transformation of data. It involves the use of independent filters that perform specific data processing tasks and are connected through pipes, forming a data flow network. Each filter takes input data, applies specific operations, and produces output data.

Integrating the “Pipes and Filters” approach alongside the “Adapter Pattern” can bring several advantages:

Flexibility and Extensibility The use of filters allows for the modularization of data processing logic. New filters can be added or existing ones modified without affecting the overall system, providing flexibility to accommodate changes in data processing requirements.

Separation of Concerns The “Pipes and Filters” architecture separates the processing logic into individual filters, each responsible for a specific task. This separation allows for better code organization, maintainability, and understanding of the data processing flow.

Reusability Filters can be reused across different integration scenarios or even in other projects. Once developed and tested, filters can be easily plugged into the data flow network, promoting code reuse and reducing development effort.

Scalability The modular nature of the “Pipes and Filters” architecture enables scalability. Filters can be distributed across multiple nodes or systems, allowing for parallel processing and accommodating high volumes of data.

For example, in the wearable-vehicle integration scenario, different filters can be designed to handle specific data transformations, such as data normalization, aggregation, or anomaly detection. The adapter components can serve as the entry point to the data flow network, receiving data from wearables and feeding it into the filters for processing. The processed data can then be passed through subsequent filters or forwarded to other components within the integration system.

4.6.4 Advantages of Device Service Proxy

The Device Service Proxy, which utilizes a common protocol, the adapter pattern, and pipes and filter, offers several advantages for wearable-vehicle integration, enhancing interoperability, data processing, flexibility, and fostering ecosystem growth. By establishing a unified framework for communication and data exchange, the integration platform can seamlessly integrate wearables with different protocols, ensuring seamless interoperability and easy scalability.

Interoperability

The common protocol enables seamless interoperability among wearables with different protocols. It eliminates the need for the middleware to have prior knowledge of the specific protocols used by each wearable, allowing for flexible integration and easy scalability. With the common protocol, wearables from various vendors can communicate and exchange data effectively within the integration platform, regardless of the underlying communication protocols.

Unified Data Processing

By converting diverse protocols to a common format, such as JSON, the middleware can perform unified data processing through matching and mapping. The common protocol message structure can be enriched with additional metadata, including the device ID, device type ID, and specific data type ID of the wearable device. This metadata enhances the semantic understanding of the data and facilitates consistent data transformation, data schema matching, and analysis techniques on the received data.

The metadata provides crucial context and information about the source and nature of the data, allowing the middleware to apply appropriate data processing algorithms and logic. By unifying the data processing pipeline and providing a consistent data format, the integration platform can deliver standardized and cohesive services to applications and users, regardless of the underlying protocols used by the wearables.

Additionally, the separation of plug-ins (adapters) and pipes-and-filters (data processing) can be implemented to handle the variations in data generated by different wearables. The plug-ins focus on the communication aspect, translating the common protocol messages to the specific protocols of the wearables. On the other hand, the pipes-and-filters components handle the processing of the received data, performing necessary transformations, filtering, or aggregation based on the specific requirements of the integration platform.

This separation allows for modularity and extensibility, enabling the integration platform to accommodate new wearables with minimal changes to the existing components. The common protocol serves as the foundation for communication, while the pipes-and-filters components provide the necessary flexibility for handling variations in data across different wearables.

Flexibility and Future-Proofing

The common protocol promotes flexibility and future-proofing of the integration platform. As new wearables with different protocols emerge, new protocol-specific adapters can be developed and seamlessly integrated into the middleware. The modular architecture of the integration platform allows for the easy addition or replacement of adapters, ensuring compatibility with the evolving landscape of wearables.

The decoupling of the communication (adapters) from the data processing (pipes-and-filters) enables the integration platform to adapt to changing protocols and technologies without disrupting the overall system. This flexibility allows the platform to support a wide range of wearables and accommodate future advancements in the wearable industry, providing a future-proof solution for wearable-vehicle integration.

Ecosystem and Innovation

By establishing a common protocol, an ecosystem is fostered, encouraging wearable device vendors and developers to align with the common standard. This promotes interoperability, and simplifies the development of new wearable applications and services that can leverage the integration platform. The common protocol creates a unified framework for collaboration, enabling developers to build upon existing functionality and integrate their applications seamlessly with the integration platform.

The common protocol also opens up opportunities for third-party developers to create plug-ins, filters, and services that enhance the capabilities of the integration platform. This vibrant ecosystem of developers and vendors drives innovation, expands the range of available services, and ultimately benefits end-users by providing a rich and diverse ecosystem of wearable-vehicle integration solutions.

4.7 Backend Connector and the Cloud

The connection of the integration platform to backend systems, including cloud infrastructure, facilitated by the backend connector, ensures that data from the wearables and the platform can be effectively used, processed, and stored. Leveraging the power of cloud computing offers scalability and flexibility for the overall integration platform. This section discusses the significance of the backend connector in bridging the wearable-vehicle integration platform with backend systems, including the cloud, and the benefits of the application of the cloud.

4.7.1 Backend Connector

The backend connector serves as a link that connects the wearable-vehicle integration platform with various backend systems, including databases, cloud services, and existing systems within the connected vehicle. Its primary function is to facilitate efficient and reliable data exchange, retrieval, and storage operations between the integration platform and these backend systems.

By abstracting the complexities associated with different backend technologies and cloud services, the backend connector streamlines the integration process and enables seamless communication and data synchronization between the wearable-vehicle integration platform and the backend services. It acts as a mediator, ensuring the smooth flow of information and enabling the integration platform to leverage the capabilities of existing proprietary systems and cloud-based services within the connected vehicle.

In addition to handling data exchange, the backend connector plays a vital role in data transformation and formatting. It ensures that data received from the wearables is translated into a format compatible with the backend systems, enabling seamless interoperability and data synchronization. Conversely, it also converts data retrieved from the backend systems into a format suitable for consumption by the wearables and the integration platform.

The backend connector incorporates various mechanisms for authentication, authorization, and secure data transmission to safeguard the integrity and confidentiality of the exchanged information. It establishes secure connections with the backend systems, implements access controls, and enforces security protocols to protect sensitive data.

Furthermore, the backend connector demonstrates a high degree of scalability and extensibility. It can seamlessly accommodate the integration of new wearables and adapt to evolving backend systems. This flexibility enables the integration platform to evolve alongside technological advancements, ensuring future-proofness and facilitating the integration of emerging wearable devices.

By acting as a link between the wearable-vehicle integration platform and the backend systems, the backend connector enhances the overall functionality, interoperability, and efficiency of the integration platform. It empowers the platform to harness the capabilities of existing backend services, leverage data from connected vehicles, and provide a unified and seamless user experience for wearable users within the vehicle environment.

4.7.2 Cloud

Cloud computing can be utilized to host and deploy the wearable-vehicle integration platform or its associated components. The cloud provides a scalable and flexible environment for hosting the middleware, adapters, filters, and other components of the integration platform. It offers on-demand resources, such as computing power, storage, and networking capabilities, which can be dynamically allocated and scaled based on the needs of the system.

By leveraging cloud infrastructure, the integration platform can benefit from the inherent advantages of cloud computing, including high availability, scalability, fault tolerance, and cost-effectiveness. The cloud enables the seamless deployment and management of the integration platform, allowing for efficient resource utilization and minimizing the need for upfront infrastructure investments.

For example, the cloud can provide additional services, such as data analytics, machine learning, and real-time processing, which can enhance the capabilities of the wearable-vehicle integration platform. By leveraging cloud-based services, the platform can perform advanced data analysis, generate meaningful insights, and enable intelligent decision-making based on the integrated wearable data.

Furthermore, introducing a cloud device manager along with the local device manager enables managing wearables in both local and remote environments and exploits benefits of cloud computing for enhanced functionality and resource utilization.

Local Device Manager The local device manager is a component within the wearable-vehicle integration platform that is responsible for managing and interacting with the wearable devices that are physically or frequently present within the local environment, i.e. the vehicle's infotainment system.

The local device manager handles tasks related to device discovery, data synchronization, and device-specific configurations.

By having a dedicated local device manager, the integration platform can efficiently manage the local wearables without connection to the cloud, and provide real-time services that require low-latency communication with the devices.

Cloud Device Manager The cloud device manager is a component or service that operates in the cloud infrastructure and is responsible for managing and coordinating the wearable devices that are connected to the integration platform remotely, typically over the internet.

The cloud device manager acts as a centralized hub for managing a large number of wearables across different locations. It handles tasks such as device registration, authentication, access control, firmware updates, and remote configuration management. The cloud device manager provides a unified interface for managing and monitoring the wearables, regardless of their physical location.

By utilizing a cloud device manager, the integration platform can benefit from centralized device management, scalability, and ease of maintenance. It allows for efficient management of a distributed network of wearables, provides global visibility and control, and enables seamless integration with cloud-based services and analytics platforms.

Offloading in the Cloud Offloading certain tasks or services, such as offloading computationally intensive tasks, data storage and processing, to the cloud, enables the integration platform to leverage the scalability, computational power, and storage capacity of the cloud infrastructure. This allows for efficient utilization of resources, cost optimization, and flexibility in scaling up or down based on the workload requirements.

4.8 Summary

Following a detailed technical exploration in the preceding chapter that identified "the fragmentation problems", encompassing market, hardware, and software challenges for wearables, this chapter focuses on constructing an architectural solution for the integration of wearables with connected vehicles.

To address each aspect of the fragmentation problem, the chapter introduces essential elements tailored to counter specific challenges. These elements, although devised as standalone solutions, are cohesively grouped into components to present a holistic architectural framework. This framework breaks down into:

- **The Device Layer:** Abstracts wearables into three primary types, based on the findings from the fragmentation problem.

- **Comprehensive Device and Service Discovery Mechanism:** Discusses all vital aspects for the efficient identification and interfacing of the device layer.
- **Device Manager:** Manages the lifecycle of wearable devices and stores meta-information for interactions in the platform.
- **Device Service Proxy:** Acts as an intermediary, facilitating communication with wearables by adopting a common protocol and the adapter pattern.
- **Backend Connector:** Serves as the link to backend systems and the cloud, ensuring efficient and reliable data exchange, retrieval, and storage operations between the integration platform and these backend systems.
- **The Cloud:** Provides scalability and flexibility for the overall integration platform.

Each component is described in detail, with technical considerations and operational nuances discussed in depth.

The subsequent chapter will focus on constructing a proof-of-concept based on the outlined architecture and conducting experiments to validate its feasibility.

5 Experiments and Evaluation

In this chapter, the experiments and findings of a small prototype for wearable-vehicle integration are presented. The specific objectives, evaluation metrics, and experimental setup are outlined. Illustrations of the user interfaces showcase its capabilities and potential for seamless integration and data exchange. An evaluation of the prototype based on predefined metrics is conducted. Challenges faced during the experiments are discussed, providing valuable insights for future development. The chapter concludes with a summary of key findings.

5.1 Overall Goal

The objective of the experiments is threefold:

1. to assess the feasibility and establish the proof-of-concept of the proposed high-level architecture for wearable-vehicle integration;
2. to identify potential challenges for future implementation and refinement of the integration platform;
3. to serve as an illustrative demonstration of the previously proposed architecture.

Due to the constraints of a limited time frame for prototyping, our focus lies on prioritizing the essential functionalities and key components that effectively showcase the core aspects of the wearable-vehicle integration platform. Such an approach is essential when dealing with large and complex systems, as it allows for targeted evaluations of specific components, which may not be feasible to assess in their entirety at once.

The conducted experiments aim to address the critical aspects of the integration platform, providing insights into the following key aspects:

1. Device Discovery and Connection: Explore the device discovery and connection mechanism by conducting experiment for a specific type of wearable. We explore the widely used wearable technology and Bluetooth Low Energy (BLE) and Wi-Fi and develop the necessary code to discover and establish connections with the wearables.
2. Data Exchange: Develop a simplified data exchange module that showcases the reception of data from the connected wearables.
3. Adapter Pattern and Backend Connector: Create prototypes for the adapter component and the backend connector. The adapter prototype should focus on dealing the translating the common protocol messages to the specific protocol used by the selected wearable technology. The backend connector prototype should demonstrate the integration with a simple backend system, such as a local database, to showcase data synchronization.

4. User Interface: Design and develop a basic user interface that allows users to interact with the wearables and access the integrated functionalities. Prioritize the core functionalities and a simple user interface design to showcase the basic interactions and data visualization.

By focusing on these core components and functionalities, the aim is to create a functional prototype within the given timeframe. The prototype may not encompass all the advanced features and complexities of the final system, but it should serve as a proof-of-concept to validate the feasibility and integration capabilities of the wearable-vehicle integration platform.

5.2 Experiment

In this section, the metrics for evaluating the experiments are established. Subsequently, the decision-making process concerning the development environment is discussed, followed by a list of the chosen experimental materials, which were readily available for conducting the experiments. Furthermore, insights into the experiment's setup are provided, involving the integration platform in conjunction with the essential registration and configuration procedures at the market vendor's site.

5.2.1 Metrics

To evaluate the experiments and measure the success of the conducted prototypes, the following metrics are defined:

1. **Coverage of Functionality** This metric assesses the extent to which the essential components of the proposed high-level architecture have been implemented. It is quantified by counting the number of core functionalities that have been implemented to some extent within the prototype. The evaluation is based on the implementation of the following core functionalities: 1. Device Discovery and Connection, 2. Data Exchange, 3. Adapter Pattern and Backend Connector, 4. User Interface.
2. **Successful Data Transmission and Reception** This metric focuses on the reliability of data transmission and reception between wearables and the middleware. It involves measuring the rate of successful data transmission and reception for the experiment devices, ensuring that the data flows seamlessly between devices and the integration platform.
3. **Device-specific Functionality Support** This metric measures the middleware's ability to support device-specific functionalities and unique capabilities provided by each wearable device. It quantifies the percentage of experimental devices that the middleware can effectively handle, demonstrating the system's adaptability and compatibility with different wearables.

5.2.2 Development Environment Selection

In this study, the choice of developing the project using Python and Flask on a virtual Ubuntu machine while having the option to directly implement it on a Windows computer using Visual Studio Code was evaluated. The decision to use a virtual Ubuntu machine for development purposes was considered based on the following factors:

1. **Environment Consistency**: Developing on a virtual Ubuntu machine ensured that the development environment closely resembled the target production environment, reducing compatibility issues and ensuring consistent behavior across platforms.
2. **Dependency Management**: The virtual Ubuntu machine facilitated effective management of project dependencies, ensuring consistent package versions across development and production environments.
3. **Linux-Specific Features**: If the project required Linux-specific features or services, developing on a virtual Ubuntu machine allowed for their utilization without complications, particularly when interacting with Linux-based system resources or APIs.
4. **Testing and Debugging**: Running and testing the Flask application within an environment that closely resembled the target production environment facilitated the identification and resolution of Linux-specific bugs or dependencies at an early stage of development.

5.2.3 Experimental Materials

The materials used for the experiments conducted in this study include the following:

- Garmin quatix 7: A smartwatch designed for water sports, offering features for both everyday life and an active lifestyle.
- Fitbit Inspire 2: An energy-efficient fitness tracker that measures vital data, such as heart rate.
- Virtual Ubuntu Machine: A virtual machine running the Ubuntu operating system.
- Visual Studio Code: An integrated development environment capable of supporting both Linux and Windows platforms for coding and project management.

5.2.4 Experimental Setup

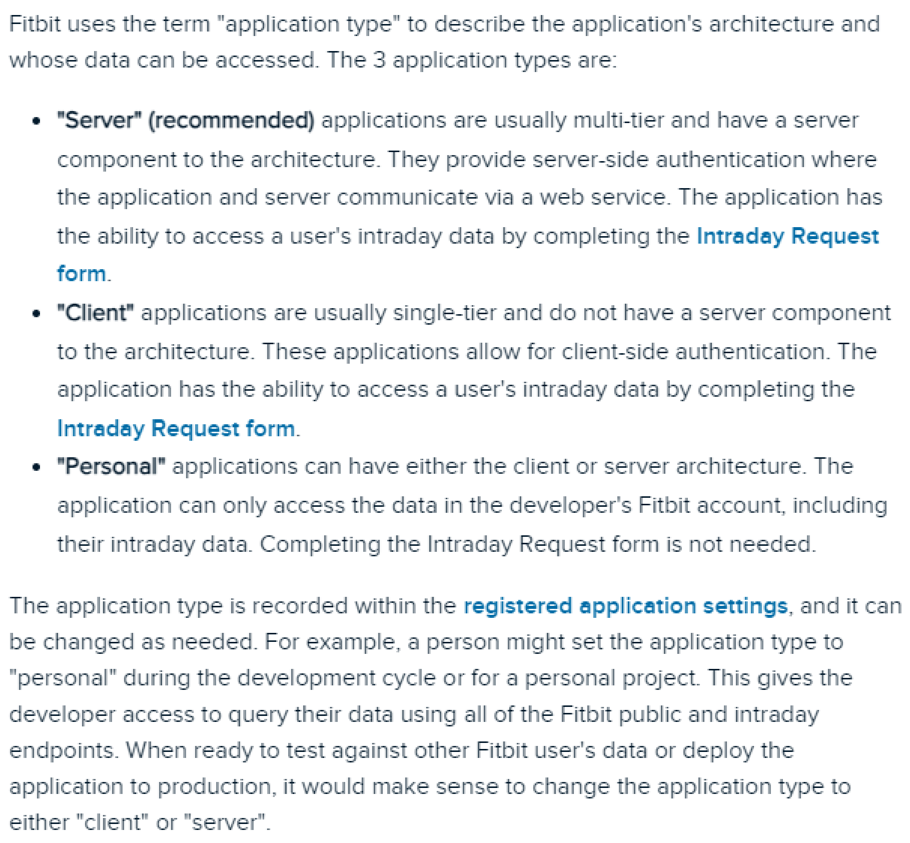
This subsection describes the experimental setup, which encompasses three essential parts, each focusing on a specific aspect of the experimental preparation and implementation of the experiments.

Fitbit Registration

To use the Fitbit Web APIs, an application should be registered to integrate with the Fitbit services. As long as the application complies with the Fitbit Platform Terms of Service, the Fitbit User Data and Developer Policy, and the Fitbit user consents to share their data with the developer's application, integration is permissible [AUN][APD].

Among the following three types of application types listed in Figure 5.1, the personal type is adopted. This choice suits the need for exploring and experimenting.

The development of a fitness tracker application involves generating two key legal documents: the Privacy Policy and the Terms of Service. In Table 5.1 and Table 5.2, we summarize the key components of the Privacy Policy and the Terms of Service. The Privacy Policy outlines how the


Fitbit uses the term "application type" to describe the application's architecture and whose data can be accessed. The 3 application types are:

- **"Server" (recommended)** applications are usually multi-tier and have a server component to the architecture. They provide server-side authentication where the application and server communicate via a web service. The application has the ability to access a user's intraday data by completing the **Intraday Request form**.
- **"Client"** applications are usually single-tier and do not have a server component to the architecture. These applications allow for client-side authentication. The application has the ability to access a user's intraday data by completing the **Intraday Request form**.
- **"Personal"** applications can have either the client or server architecture. The application can only access the data in the developer's Fitbit account, including their intraday data. Completing the Intraday Request form is not needed.

The application type is recorded within the **registered application settings**, and it can be changed as needed. For example, a person might set the application type to "personal" during the development cycle or for a personal project. This gives the developer access to query their data using all of the Fitbit public and intraday endpoints. When ready to test against other Fitbit user's data or deploy the application to production, it would make sense to change the application type to either "client" or "server".

Figure 5.1: Application types supported by Fitbit [APD]

application collects, uses, stores, shares, and protects user data. Meanwhile, the Terms of Service delineate the rules users must abide by when using the application. These documents are essential for defining the business's relationship with its users. While generic templates are available online, customized policies are crucial due to the unique requirements of each business. Several online platforms, including Termly and Iubenda, offer services to generate these documents. Although these services provide document generation tools, a comprehensive review by a legal professional is indispensable for ensuring legal compliance. In our demo project, we only provide a mock link for these legal documents.

Fitbit employs OAuth 2.0 for authorization, offering support for several authorization models including Authorization Code Grant Flow, Authorization Code Grant Flow with PKCE, and Implicit Grant Flow [AUN].

When developing integrations with Fitbit's Web APIs to utilize data services, a specific redirect URL is crucial. This URL serves as a route within the integration application, enabling us to capture and handle the response from Fitbit's authorization server.

In our integration application, we utilize the Authorization Code Grant Flow. After a user has granted authorization for our integration app via the Fitbit website, the user is redirected back to our specified endpoint, accompanied by an authorization code. This code is then leveraged to exchange for an access token. This exchange is achieved by making a POST request to Fitbit's token endpoint, https://api.fitbit.com/oauth2/token.

Component	**Description**
Data Collection	Clarification on the types of data collected, the methods of collection, and the reasons for its necessity.
Data Usage	Explanation of how the user's data will be used.
Third-Party Sharing	Details on whether data is shared with third parties, and if so, the reasoning and involved parties.
Data Protection	Measures taken to protect user data.
User Rights	The rights of users, including the right to access, correct, and delete data.
Policy Changes	How users will be informed of changes to the policy.

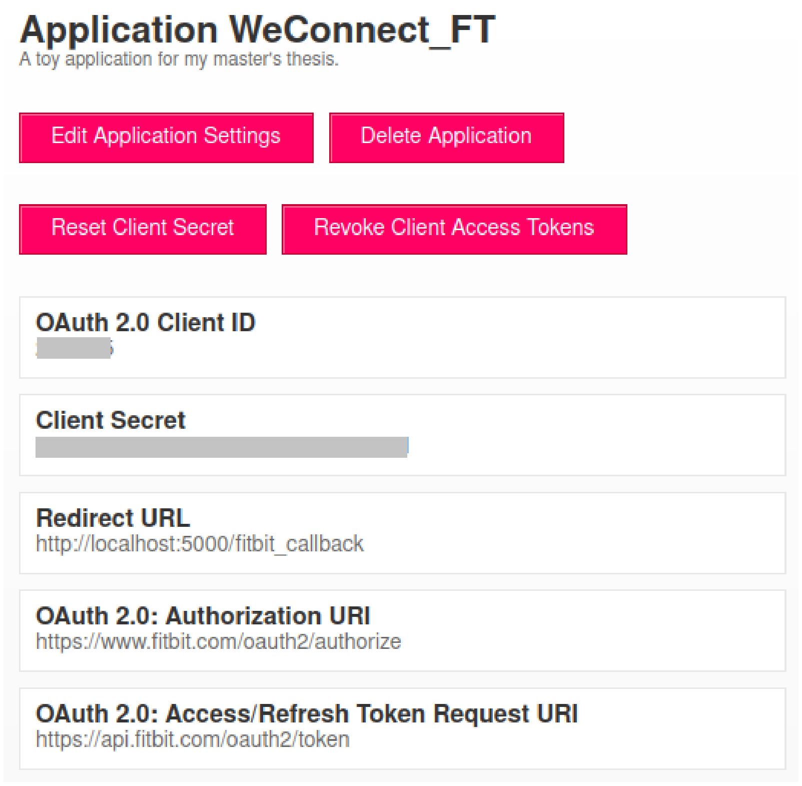
Table 5.1: Key contents of Privacy Policy (derived from [LA02][MT06])

Component	**Description**
User Obligations	Expected behaviors and unacceptable activities from users.
Intellectual Property	Ownership of content and logos within the application.
Termination Conditions	Terms and conditions for account termination.
Limitations of Liability and Disclaimers	Outline of the company's liabilities and disclaimers.
Governing Law	The jurisdiction that governs the terms.
Dispute Resolution	Method by which disputes will be handled.
Changes to Terms	How users will be informed if the terms change.

Table 5.2: Key components of Terms of Service (derived from [VLM+16])

The exchange process necessitates two key credentials, as depicted in Figure 5.2. The first is the 'client id', which should be provided by the user to the integration application. The second key credential is the 'client secret', assigned to our integration application (the client) by Fitbit's authorization server. The purpose of the 'client secret' is to authenticate our application to Fitbit's authorization server.

If the access token request is successful, the token is subsequently stored within a local SQLite database. This stored access token allows the application to make authorized API requests on behalf of the user, enabling the extraction of valuable user data from the Fitbit platform.



Application WeConnect_FT

A toy application for my master's thesis.

OAuth 2.0 Client ID
[redacted]

Client Secret
[redacted]

Redirect URL
http://localhost:5000/fitbit_callback

OAuth 2.0: Authorization URI
https://www.fitbit.com/oauth2/authorize

OAuth 2.0: Access/Refresh Token Request URI
https://api.fitbit.com/oauth2/token

Figure 5.2: Fitbit application settings

Limitations in Garmin Development Integration

The integration of Garmin wearable devices into the proposed research project faced limitations due to strict constraints imposed by the Garmin Connect Developer Program. The request for access to Garmin's data exchange solutions underwent a rigorous review process and was not approved for this academic research project. The reasons for non-approval are primarily related to the specific requirements set forth by Garmin for enterprise customers:

1. **Valid Website and Company Association:** Garmin requires the requester to have a valid website associated with the company for which access is being requested. In this academic research project, the absence of a dedicated website linked to the requesting organization may have contributed to the non-approval.

2. **Privacy Policy Link Accessibility:** Garmin stipulates that the requester's homepage should have an externally accessible privacy policy link, available to all visitors. The absence of such a link might have impacted the access request.

3. **Legal Authority for Representation:** The requester is mandated to possess the legal authority to represent the company or organization for which access is sought. As an academic research project, the representation of a research entity might not meet Garmin's enterprise customer requirements.

4. **Non-Personal Use:** Garmin's data exchange solutions are designed primarily for non-personal use cases, and access is typically granted to enterprises or businesses rather than academic research endeavors.

Given the constraints and requirements set forth by Garmin, it is evident that the integration of Garmin wearable devices into the experimental setup for this academic research project is not feasible. Conducting experiments involving Garmin devices would require collaboration with enterprises or organizations that meet Garmin's specific criteria.

Although this research project currently excludes experiments with Garmin devices, future studies by entities meeting Garmin's criteria may offer valuable insights into integrating these wearables with the proposed platform.

The Prototype Application

As mentioned in Section 5.2.2, the prototype application is a Flask-based integration platform that facilitates wearable-vehicle integration. In this section, we discuss the key aspects of the prototype application, focusing on device discovery and connection, data exchange, as well as the implementation of the adapter pattern and backend connector.

Device Discovery and Connection In the context of wearable-vehicle integration, device discovery is a critical process that allows the integration platform to identify and connect with wearable devices. We have explored both user-initiated and automatic device discovery methods in Section 4.4.2. For the Fitbit Inspire 2 Fitness Tracker, we encountered some challenges with Bluetooth and Wi-Fi scanning, as the device's Bluetooth mode is set to invisible when connected to a paired device, and it relies on its paired device to join Wi-Fi networks.

As a result, we opted for a user-initiated device discovery method. When a user initiates the discovery process, the corresponding authentication flow for the Fitbit Inspire 2 Fitness Tracker manages the connection between the device and the integration platform. This approach proved to be effective in establishing connections with the Fitbit device.

Data Exchange Once the Fitbit Inspire 2 Fitness Tracker is successfully connected, the prototype application receives a response from the Fitbit authentication server containing an access token, expiration time, refresh token, scope, token type, and user ID.

To protect sensitive information, we have replaced them with "HIDDEN" in our response example:

```
{
        "access_token": "HIDDEN",
        "expires_in": 28800,
        "refresh_token": "HIDDEN",
        "scope": "respiratory_rate location profile sleep oxygen_saturation nutrition
                social settings activity weight temperature heartrate",
        "token_type": "Bearer",
        "user_id": "HIDDEN"
}
```

With the access token obtained, the prototype application can successfully retrieve various data through Fitbit's Web APIs [WEA] . This data might include fitness and health-related information, depending on the user's authorization scope.

However, the actual process of data transformation could benefit from more detail in future development.

Adapter Pattern and Backend Connector Our prototype operates on a monolithic architecture and integrates both the service proxy and adapter design patterns. HTTP requests, structured in a standard format (e.g., POST `/{position_id}/{device_type_id}/services/{service_id}`), along with the user interface, function as the service proxy. In contrast, the adapter is embodied by specific Python methods, HTML templates, and redirection logic. These manage the connection and data retrieval processes tailored for each device type.

In terms of backend connectivity, we integrated a local SQLAlchemy database to store information about successfully connected wearable devices and the wearer's position in the vehicle. Here is a snippet of the database models:

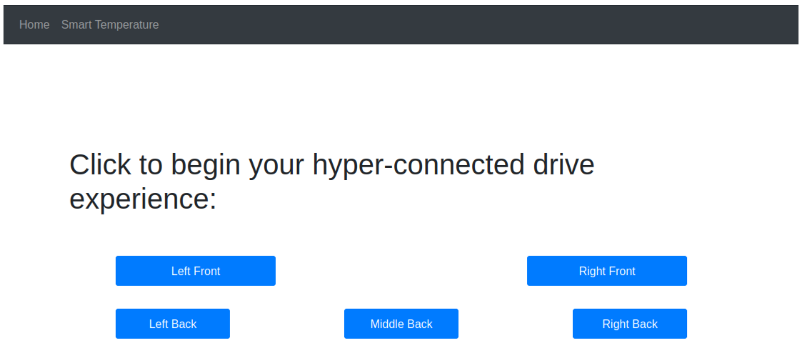
```
class CarPosition(db.Model):
id = db.Column(db.String, primary_key=True)
device_id = db.Column(db.String, db.ForeignKey('device.id'))


class Device(db.Model):
id = db.Column(db.String, primary_key=True)
manufacturer = db.Column(db.String)
model = db.Column(db.String)
adapter = db.Column(db.String)
client_id = db.Column(db.String)
client_secret = db.Column(db.String)
access_token = db.Column(db.String)
refresh_token = db.Column(db.String)
car_position = db.relationship('CarPosition', backref='device', uselist=False)
```

By utilizing the adapter pattern and the backend connector, the prototype application demonstrates its adaptability to various wearables and its capability to successfully store relevant data in the local database.

5.2.5 Demonstration of the Prototype

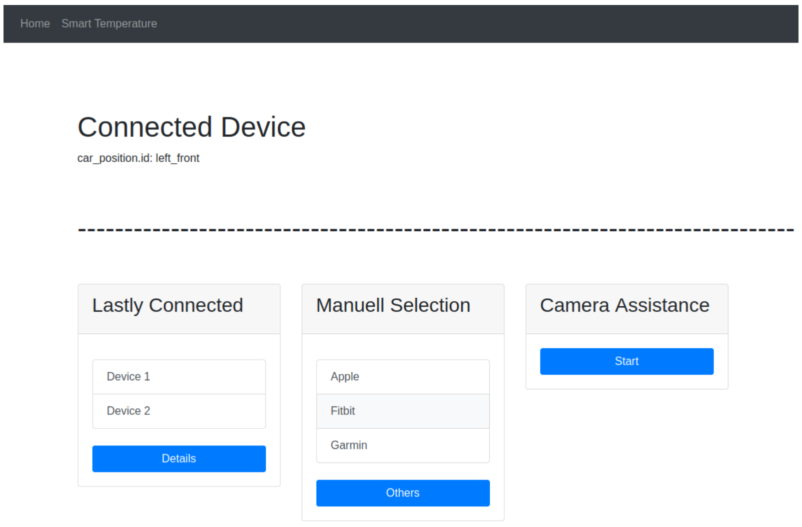
We demonstrate the prototype by walking through a successful process flow for connecting and retrieving data from the Fitbit Inspire 2 Fitness Tracker. The following screenshots illustrate the step-by-step process:

Screenshot 1: Main Page of the Prototype Application



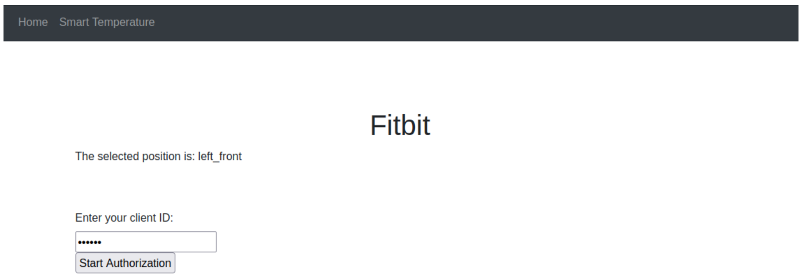
The main page of the prototype application shows the user interface with options for device discovery and connection for each position. The user chooses a position and initiates the device discovery process by clicking on the specific position button.

Screenshot 2: Discovery Page



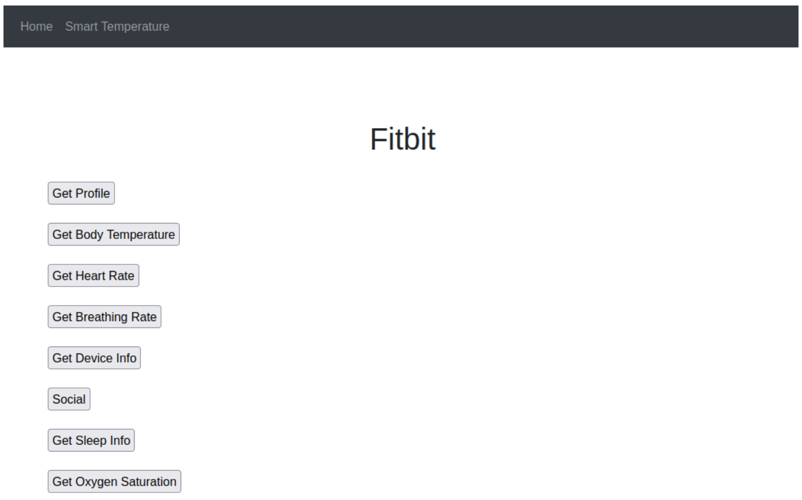
The application displays the already connected devices for this position, and provides several options for the user to start a new connection.

Screenshot 3: Fitbit Device Connection



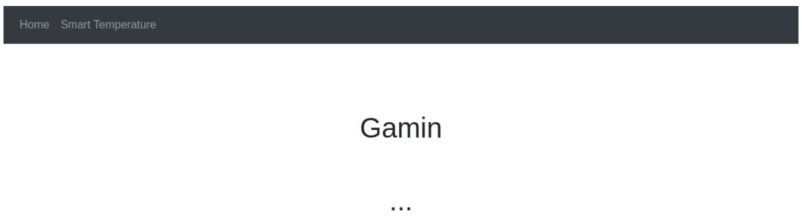
If the use chooses Fitbit for a new connection, the user should provide his/her client id for the authentification.

Screenshot 4: Data Exchange and Retrieval



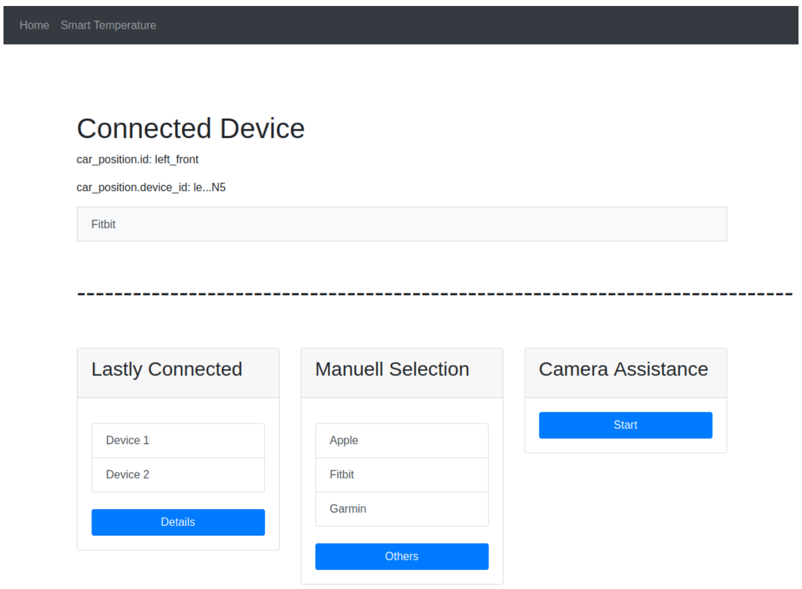
After successful device connection, the Fitbit authentication server returns the access token required for data retrieval. The prototype application shows the page for accessing various types of data (e.g., profile, heart rate) from Fitbit.

Screenshot 5: Service Proxy and Adapter Selection



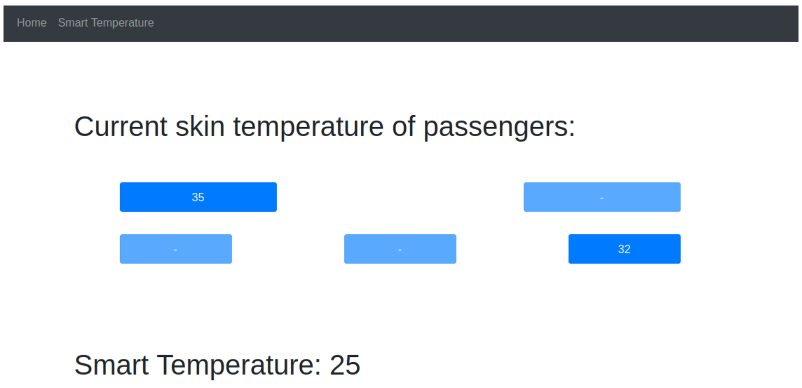
The service proxy and adapter components dynamically load the appropriate Python methods and HTML templates based on the device type selected by the user. If Garmin is chosen in the connection page, the mock HTML page for Garmin is rendered.

Screenshot 6: Data Storage in Backend Database



The prototype application stores information about connected wearable devices and their positions in the vehicle in the local SQLAlchemy database. Available services can be accessed without new user authorization.

Screenshot 7: Smart Temperature Management



The system demonstrates the emergence effect of smart temperature management, where the air-conditioner intelligently adjusts based on the collected passenger data. This ensures a comfortable and personalized environment for each passenger.

5.3 Evaluation

To validate the success of the experiments and the functionality of the developed prototype, we evaluate the outcomes based on the predefined metrics. This section will detail the evaluation process of the experiments and their outcomes.

5.3.1 Coverage of Functionality

Objective: To determine the extent of core functionalities' implementation in the prototype.

Procedure:

- Each core functionality, including Device Discovery and Connection, Data Exchange, Adapter Pattern and Backend Connector, and User Interface, was tested separately.
- The functionalities were then categorized based on their implementation status.

Table 5.3: Coverage of functionality

Functionality	Implemented	Not Implemented
Device Discovery and Connection	✓	
Data Exchange	✓	
Adapter Pattern and Backend Connector	✓	
User Interface	✓	

5.3.2 Device-specific Functionality Support

Objective: To ascertain the middleware's capability in supporting different device-specific functionalities.

Procedure:

- Experimental wearables were prepared to the integration platform.
- Each wearable's unique capability was triggered to see if the platform recognized and effectively handled it.
- The number of supported devices against the total tested devices was recorded.

Results:

Table 5.4: Device-specific functionality support

Wearable Device	Supported	Not Supported
Fitbit Inspire 2	✓	
Garmin Quatix 7		✓

Analysis: By addressing core functionalities, as detailed in Section 5.2.4, and demonstrating them through the user interface, as presented in Section 5.2.5, the prototype application serves as a preliminary proof-of-concept, showcasing the potential feasibility and core functionalities of the wearable-vehicle integration platform. However, due to access constraints, we only developed the connection and data exchange functions for one experimental device. Further experimentation with other devices would provide a more comprehensive validation.

5.3.3 Successful Data Transmission and Reception

Objective: To measure the rate of successful data transmission between wearables and the integration platform.

Procedure:

- Prepare the wearable device for connection to the integration platform.
- Initiate the connection.
- Once connected, the platform sends data requests that are relevant for wearable-vehicle integration and fall within the scope of authorization.
- Record the number of successful responses received out of the total requests sent.
- Repeat the above steps for 3 days.

Results:

Table 5.5: Successful data transmission and reception

Data	Day 1		Day 2		Day 3	
	Total Requests Sent	Successful Responses Received	Total Requests Sent	Successful Responses Received	Total Requests Sent	Successful Responses Received
Access Token	10	7	10	6	10	6
User Profile	10	10	10	10	10	10
Body Temperature	10	10	10	10	10	10
Heart Rate	10	10	10	10	10	10
Breathing Rate	10	10	10	10	10	10
Device Info	10	10	10	10	10	10
Sleep	10	10	10	10	10	10
Oxygen Saturation	10	10	10	10	10	10

Analysis:

As the middleware currently only supports one of our experimental wearable devices, this experiment is carried out solely with the Fitbit Inspire 2.

As indicated in Table 5.5, requests for access tokens did not consistently succeed. Initial attempts encountered failure, but subsequent trials consistently achieved success after the initial breakthrough. The successful attempts ensured the retrieval of access tokens, refresh tokens, expiration details, etc. Despite these observations, the exact cause of this periodic behavior remains unclear, possibly originating from factors within Fitbit's Authentication Server.

Upon gaining authorization for user data access, the process of data transmission and reception becomes reliably consistent. We consistently received successful responses, including comprehensive user details, device specifics, second-level heart rate data, and detailed sleep information, all in JSON format. Nevertheless, when requesting data such as body temperature, breathing rate, and oxygen saturation, the requests are successful, but the returned data is empty.

5.4 Challenges and Insights

The experiments conducted in this study faced multiple challenges, ranging from access constraints to hardware connectivity hurdles. Each challenge is described below, along with their implications. Insights are further provided to guide future development.

Limited Availability of Experimental Materials The experimental scope was restricted to two specific wearable devices, and access to the API documentation of one of these devices was not granted despite persistent efforts. The limited wearable range explored in this study can be attributed to several factors:

- **Vendors' Business Model and Confidentiality Constraints:** Wearable vendors often establish differentiated access restrictions to their documentation for individual researchers, which might be driven by the need to protect their business strategy and proprietary information.
- **Requirements for Fully Functional Applications:** Certain manufacturers necessitate that fully-developed applications be submitted, reviewed and approved before allowing integration. This can impose barriers to preliminary experimental studies.
- **Diverse Programming Knowledge:** Integration experiments demand proficiency across multiple programming environments, languages, and libraries. The diverse skill set required may pose challenges for researchers with limited experience in certain technologies.
- **Time Constraints:** The experimental phase of this study had a limited timeframe, given it was a first attempt at developing the integration platform. The time constraints have limited the number of wearable devices that could be included in the experiments.

To address these challenges and expand the scope of future research, it is imperative to evaluate these factors and seek sufficient business support, industry support and expertise in relevant technologies.

Data Sensitivity and Security Wearable devices collect sensitive data, and their integration with connected vehicles introduces safety and security concerns. To ensure the responsible handling of data, the wearable vendors would:

- **Limited Access to Wearable Data:** During the experimental phase and development, access to wearable data was restricted to the developers to safeguard sensitive information.
- **Legal Obligations when Sharing Data:** When sharing wearable data with a vehicle market vendor, the wearable vendor would clear the legal obligations and ensure that the data usage on the vehicle side complies with privacy and security regulations.

Vehicle manufacturers, while leveraging data from wearables, must emphasize responsible and secure data utilization. Given the high safety concerns associated with vehicles, the utilization of sensitive data requires research and careful design to guarantee secure and responsible data handling.

Connectivity Limitation Wearables, bacause of their characteristics and common usage (as a companion to another, more powerful device), it poses challenge for a common approach for automatic discovery and connection.

In our experiment for the device discovery and connection with Fitbit Inspire 2. Though the wearble supports Bluetooth, it is not an option for the integration. Because this wearable model works in combination with a smart phone via Bluetooth. And when paired, it is invisible for other devices.

Various connectivity options and proprietary protocols, such as Apple's Bonjour, should be further explored. A uniform IP-based standard, like Matter [MAT] for smart home, is current impractical. Most wearables, like the Apple Watch, do not always possess a standalone IP address. For example, when an Apple Watch is paired and in proximity to the paired iPhone, it relies on the iPhone's connection and doesn't have a separate IP address. When the Apple Watch is away from the iPhone but connected to a known Wi-Fi network directly, it can have its own IP address. However, not all Apple Watch models have the ability to connect to Wi-Fi networks directly. Some basic models might rely solely on the iPhone's connection.

5.5 Summary

This chapter presents the results of our experiments with a preliminary proof of concept that embodies the proposed high-level architecture for wearable-vehicle integration, as discussed in the previous chapter. The goal of the experiments was to test the feasibility of the integration concept and provide an initial glimpse of the core functionalities. Evaluating this proof of concept against predefined metrics, we discerned valuable insights about its strengths and limitations.

The challenges faced during the development of this proof of concept offered significant insights for refining the integration platform. These findings will serve as a foundation for future researchers to enhance the implementation, address the identified challenges, and evolve towards a more refined system.

6 Conclusion and Outlook

In this master's thesis project, the focus was on bridging the realms of wearables and connected vehicles, echoing the growing emphasis on hyper-connected lifestyles. Through this research, an effort was made to fill a research void by introducing an innovative wearable-vehicle integration platform. Up to this point, the integration of wearables, encompassing devices like smartwatches, fitness trackers, and smart rings, with connected vehicles remains an area yet to be extensively researched.

The research journey was segmented into three defining phases:

- The exploratory phase involved meticulous market research, where the unique facets of wearables were uncovered. This deep dive unraveled "the wearables fragmentation problem", a newly identified challenge, emphasizing the pressing need for a comprehensive solution linking wearables to connected vehicles.
- The conceptualization phase witnessed the design of a high-level architecture for the wearable-vehicle integration platform. This architecture, with its components like the device layer, device manager, service proxy, and backend connector, aimed to counteract the fragmentation problem, thus offering a flexible framework accommodating varying manufacturers and devices.
- The experimental phase saw the realization of the conceptualized architecture. Here, a prototype was developed, serving as an initial proof of concept. The challenges faced during this phase, which ranged from limited material availability to data security concerns and connectivity barriers, reveal the practical complexities of such an integration. However, these challenges also presented valuable insights, emphasizing the need for industry collaboration and tech expertise enhancement.

Through these three phases, the research offered crucial insights:

- The integration between wearables and vehicles is not just a technological challenge but is also laden with business considerations, especially concerning vendors' business models and confidential agreements.
- The variety in programming environments coupled with the specificity of wearables complicates the integration process, emphasizing the importance of expertise across diverse technologies.
- Safety and security remain paramount, especially when sensitive data from wearables intersect with vehicles, emphasizing the need for rigorous data handling protocols.

As technology continues to advance, and as we edge closer to a world where everything is connected, the discoveries from this research symbolize pioneering advancements for wearable technology and vehicles. It not only uncovers the potential for an integrated, intelligent ecosystem but also brings attention to the barriers to conquer, thus setting the stage for subsequent research and innovations in this realm.

With this thesis, a foundational stone has been laid in the vast expanse of wearable-vehicle integration, potentially guiding the next wave of innovation at the intersection of wearable technology and connected vehicles.

Bibliography

[ACS16] F. de Arriba-Pérez, M. Caeiro-Rodríguez, J. M. Santos-Gago. "Collection and processing of data from wrist wearable devices in heterogeneous and multiple-user scenarios". In: *Sensors* 16.9 (2016), p. 1538 (cit. on pp. 3, 11, 12, 39).

[ANT] *ANT (network)*. Wikimedia Foundation, Inc. URL: https://en.wikipedia.org/wiki/ANT_(network) (cit. on p. 14).

[APD] *Application design*. Fitbit LLC. URL: https://dev.fitbit.com/build/reference/web-api/developer-guide/application-design/#Application-Types (cit. on pp. 49, 50).

[APE] *Which Apple Watch is right for you?* Apple Inc. URL: https://www.apple.com/watch/compare/ (cit. on p. 9).

[AUN] *Authorization*. Fitbit LLC. URL: https://dev.fitbit.com/build/reference/web-api/developer-guide/authorization/ (cit. on pp. 32, 49, 50).

[Blu06] S. Bluetooth. "Bluetooth". In: *Go Faster. Go Further White Paper.* (2006) (cit. on p. 13).

[CL90] K. L. Calvert, S. S. Lam. "Formal methods for protocol conversion". In: *IEEE Journal on Selected areas in Communications* 8.1 (1990), pp. 127–142 (cit. on p. 37).

[COP] *Compatibility of Polar POLAR H9 / H10 / OH1 / Verify sense heart rate sensor*. Polar Electro. URL: https://support.polar.com/en/support/compatibility_of_polar_h6_h7_heart_rate_sensor?product_id=89452&category=settings (cit. on p. 15).

[COW] *Comparison of wireless technologies (Bluetooth, WiFi, BLE, Zigbee, Z-Wave, 6LoWPAN, NFC, WiFi. . .* Hackster.io. URL: https://www.hackster.io/news/comparison-of-wireless-technologies-bluetooth-wifi-ble-zigbee-z-wave-6lowpan-nfc-wifi-eece5593d80f (cit. on p. 13).

[CTM] *Global smartwatch shipments market share Q1 2023*. Counterpoint Technology Market Research. URL: https://www.counterpointresearch.com/global-smartwatch-shipments-market-share/ (cit. on p. 9).

[FAI] *Fetch API*. Zepp Health, Inc. URL: https://docs.zepp.com/docs/reference/side-service-api/fetch/ (cit. on p. 16).

[FIT] *Sense 2 - All features & specifications*. Fitbit. URL: https://www.fitbit.com/global/de/products/smartwatches/sense2?sku=521BKGB (cit. on p. 10).

[GAN] *All wearables & smartwatches*. Garmin Ltd. URL: https://www.garmin.com/c/wearables-smartwatches/ (cit. on p. 9).

[GTK+10] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, D. Savio. "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services". In: *IEEE transactions on Services Computing* 3.3 (2010), pp. 223–235 (cit. on p. 3).

[HDVL03] S. Helal, N. Desai, V. Verma, C. Lee. “Konark-a service discovery and delivery protocol for ad-hoc networks”. In: *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.* Vol. 3. IEEE. 2003, pp. 2107–2113 (cit. on p. 22).

[IDC1] *Wearable devices market share*. IDC Corporate. URL: https://www.idc.com/promo/wearablevendor (cit. on p. 7).

[KRG] *Overview of the K Ring contactless payment ring*. SMART RING NEWS OY. URL: https://smartringnews.com/smart-ring-products/k-ring/ (cit. on p. 11).

[LA02] C. Liu, K. P. Arnett. “An examination of privacy policies in Fortune 500 Web sites”. In: *American Journal of Business* 17.1 (2002), pp. 13–22 (cit. on p. 51).

[LOA] *List Of all operating systems that are running on different smartwatches (wearables) In 2023*. Wear To Track. URL: https://weartotrack.com/operating-systems-on-smartwatches-wearables/ (cit. on p. 16).

[LPI5] *Global wearable electronic devices market growth 2023-2029*. LP INFORMATION INC. URL: https://marketintelligencedata.com/reports/7603238/global-wearable-electronic-devices-market-growth-2023-2029 (cit. on p. 7).

[MAT] *Matter: The foundation for connected things*. Connectivity Standards Alliance. URL: https://csa-iot.org/all-solutions/matter/ (cit. on p. 61).

[MI4] *Smart wearable market - growth, trends, covid-19 impact, and forecasts (2023 - 2028)*. Morder Intelligencee. URL: https://www.mordorintelligence.com/industry-reports/smart-wearables-market (cit. on pp. 7, 8).

[MKK22] A. Mavrogiorgou, A. Kiourtis, D. Kyriazis. “A pluggable IoT middleware for integrating data of wearable medical devices”. In: *Smart Health* 26 (2022), p. 100326 (cit. on pp. 3, 32, 33).

[MOV] *Motiv Ring*. Motiv Inc. URL: https://mymotiv.com/ (cit. on p. 11).

[MT06] M. C. Mont, R. Thyne. “Privacy policy enforcement in enterprises with identity management solutions”. In: *J. Comput. Secur.* 16 (2006), pp. 133–163. URL: https://api.semanticscholar.org/CorpusID:11288581 (cit. on p. 51).

[OSK+21] A. Ometov, V. Shubina, L. Klus, J. Skibińska, S. Saafi, P. Pascacio, L. Flueratoru, D. Q. Gaibor, N. Chukhno, O. Chukhno, et al. “A survey on wearable technology: History, state-of-the-art and current challenges”. In: *Computer Networks* 193 (2021), p. 108074 (cit. on pp. 1, 3, 5, 14).

[PRB] *Protocol Buffers Documentation*. Google LLC. URL: https://protobuf.dev/ (cit. on p. 35).

[RIM07] R. Robinson, J. Indulska, T. McFadden. “Resource discovery in modern computing environments”. In: *International Journal of Pervasive Computing and Communications* 3.1 (2007), pp. 4–29 (cit. on p. 22).

[SAG] *The official website of Samsung*. Samsung. URL: https://www.samsung.com/ (cit. on p. 9).

[SRPA07] M. Strimpakou, I. Roussaki, C. Pils, M. Anagnostou. “COMPACT: middleware for context representation and management in pervasive computing”. In: *International Journal of Pervasive Computing and Communications* 2.3 (2007), pp. 229–246 (cit. on p. 22).

[TBT] *Google Maps' turn-by-turn navigation on Wear OS now works without a phone*. The Verge. URL: https://www.theverge.com/2023/1/5/23541504/google-maps-wear-os-turn-by-turn-navigation-no-phone (cit. on p. 6).

[TIN] *Tizen*. Wikimedia Foundation, Inc. URL: https://en.wikipedia.org/wiki/Tizen (cit. on p. 9).

[VLM+16] J. Venturini, L. Louzada, M. F. Maciel, N. Zingales, K. Stylianou, L. Belli. "Terms of service and human rights: an analysis of online platform contracts". In: 2016. URL: https://api.semanticscholar.org/CorpusID:169028191 (cit. on p. 51).

[WEA] *Web API Reference*. Fitbit LLC. URL: https://dev.fitbit.com/build/reference/web-api/ (cit. on p. 54).

[WEC] *Wearable Computer*. Wikimedia Foundation, Inc. URL: https://en.wikipedia.org/wiki/Wearable_computer (cit. on p. 15).

[ZVB] *Zigbee vs. Bluetooth: Choosing the right protocol for your IoT application*. Digi International Inc. URL: https://www.digi.com/blog/post/zigbee-vs-bluetooth-choosing-the-right-protocol (cit. on p. 13).

All links were last followed on August 14, 2023.

Declaration