LIÈGE université Library

https://lib.uliege.be

https://matheo.uliege.be

Optimization of Bluetooth transfer towards a lite-tech head-up display

Auteur : Bellafqih, Reda
Promoteur(s) : Boigelot, Bernard
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en informatique, à finalité spécialisée en "computer systems security"
Année académique : 2023-2024
URI/URL : http://hdl.handle.net/2268.2/20017

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.



University of Liège

Faculty of Applied Sciences

Master thesis conducted for obtaining the
Masters degree in Computer Science Engineering

Optimization of Bluetooth Transfer With a Lite-Tech Head-Up Display

---

Author:
Reda Bellafqih

Supervised by:
Bernard Boigelot
Antoine Malherbe

Academic year 2023-2024

Abstract

This master's thesis addresses the problem of optimizing Bluetooth Low Energy (BLE) technology in the development of aRdent smart glasses, with the end goal of elevating the user experience by ensuring efficient data transfer, minimal power consumption, and robust connection stability [30]. This research focuses on examining and adjusting the settings of BLE 4.1 [20], and exploring the improvements of BLE 5.0. Through a series of empirical research [44] and testing, the study rigorously benchmarks [14] performance enhancements, with a particular focus on the trade-offs between energy usage and data throughput. The comprehensive analysis extends beyond mere technical refinements, incorporating user-centric considerations to enhance in the practical application of wearable technology. The results of this research offer a substantive contribution to the field, showcasing the potential for substantial performance gains in BLE-enabled devices while paving the way for future innovations in smart eyewear technology.

Acknowledgments

I want to sincerely thank my internship supervisor, Antoine Malherbe, for guiding and supporting me in this work. I am grateful for his huge generosity of knowledge and know-how because he really helped render the experience I have had meaningful. Furthermore, Antoine was not only a project supervisor but also made me feel an important part of his team. Needless to say, I also thank my academic supervisor, Bernard Boigelot, who not only gave me the finest piece of advice and insights but also became an important character in developing this thesis. His remarks and advice have been of a great influence on my work and, in turn, of a great influence on the choice of approach to the project. I would also like to sincerely thank the team at Get Your Way for the support and help during my work.

Contents

1 Introduction **1**
- 1.1 Get Your Way . . . 1
  - 1.1.1 The Company . . . 1
  - 1.1.2 The Product . . . 2
  - 1.1.3 Use Cases Example . . . 2
- 1.2 Project Statement and Objectives . . . 3
- 1.3 Approach and Methodology . . . 3

2 Product Description **5**
- 2.1 Hardware Description . . . 5
- 2.2 Software Implementation Details . . . 7

3 Principles and Implementation of Bluetooth and BLE **9**
- 3.1 Understanding Bluetooth and BLE . . . 9
- 3.2 Communication Channels . . . 10
- 3.3 Protocol Stack Layers . . . 10
  - 3.3.1 Controller . . . 10
  - 3.3.2 Host Controller Interface (HCI) . . . 11
  - 3.3.3 Host . . . 11
  - 3.3.4 Applications . . . 11
- 3.4 Detailed BLE Packet Structure . . . 12
- 3.5 Connection Establishment . . . 12
- 3.6 Data Transfer in BLE . . . 14
- 3.7 BLE parameters . . . 15
  - 3.7.1 Connection Interval . . . 15
  - 3.7.2 Inter Frame Space . . . 16
  - 3.7.3 Attribute Maximum Transmission Unit . . . 16
  - 3.7.4 Connection Event . . . 17
  - 3.7.5 Data Transmission Modes: Write With and Without Response . . . 17
- 3.8 Theoretical Bluetooth Throughput Limitations . . . 17

4 Preliminary Performance Evaluation **19**
- 4.1 Tools . . . 19
- 4.2 Initial Throughput . . . 22
- 4.3 Modifiable BLE Parameters in aRdent 1 . . . 23
- 4.4 Analysis of Throughput Across Different Devices . . . 23
  - 4.4.1 Methodology . . . 23
  - 4.4.2 Results . . . 23
  - 4.4.3 Interpretation . . . 24

5 Optimizing BLE Parameters **25**
5.1 Optimizing the CI . . . 25
5.1.1 Modifying the CI . . . 25
5.1.2 Strategies for Optimizing the CI . . . 27
5.1.3 Experimental Setup . . . 27
5.1.4 CI Modification on Different Devices . . . 28
5.1.5 Analysis of Test Results . . . 29
5.1.6 Analysis of Test Results: Throughput . . . 31
5.1.7 Analysis of the CI in mobile phone . . . 32
5.1.8 Analysis with ST electronic software tool . . . 33
5.1.9 Theoretical Data Rate Evolution and Practical Data Rate Evolution Based on CI . . . 37
5.2 Optimizing ATT MTU . . . 37
5.2.1 Modifying ATT MTU . . . 38
5.2.2 Analysis with ST electronic software tool . . . 38
5.3 Optimizing Number of Packets per CE . . . 44
5.3.1 Modifying Number of Packets per Connection Event . . . 46
5.3.2 Analysis with ST electronic software tool . . . 46
5.3.3 Interpretation of the results . . . 54
5.4 Data Length Extension (DLE) . . . 55
5.5 LE 2M PHY . . . 57

6 Optimizing Write Without Response **58**
6.1 Analyzing Data with PacketLogger . . . 58
6.2 Investigating the Root Cause and Exploring Potential Solutions . . . 58
6.3 Investigating the Reception Buffer Capacity as a Potential Cause . . . 59
6.4 Continuous Monitoring and Dynamic Control of Data Transmission . . . 60
6.5 Modulating Transmission Speed to Enhance Data Handling . . . 60
6.6 Throughput Testing for Data Transmission . . . 61
6.7 Optimal Delay Determination for High Volume Data Transmission . . . 62
6.8 Focus on Disconnect Functionality . . . 63
6.9 Initial Problem and Mode Operation . . . 63
6.10 Understanding the Requirement for a 0.003 Second Delay . . . 64
6.11 Verification of Data Display on the Device . . . 65
6.12 Investigating the Source of Data Display Issues: Interference vs. Memory . . . 66
6.13 Testing Write Without Response Mode in Flutter . . . 67
6.14 Assessing BLE Behaviors on Alternate Devices . . . 67
6.15 Exploration of MCU Clock Speed Adjustments on Arduino . . . 68
6.16 Factors Influencing Write Without Response Mode Performance . . . 69
6.17 Note on BlueNRG-MS Chip Behavior in aRdents Glasses . . . 70
6.18 Challenges in Accurately Measuring BLE Throughput in Write Without Response . . . 70
6.19 Identifying and Resolving Firmware-Related Instabilities in Throughput Measurement . . . 72
6.20 BLE Throughput Analysis . . . 73
6.20.1 aRdent Glasses: . . . 73
6.20.2 Arduino: . . . 74
6.21 Data Transmission Strategy on aRdent Glasses . . . 76
6.22 Identification of Hardware Limitation on aRdent Glasses . . . 77
6.23 Investigation into Delay Requirement on MacOS . . . 78

7 aRdent 2 Setup and Optimization **79**
7.1 Introduction to BLE Performance Enhancement at Get Your Way 79
7.2 Historical Context and Rationale for Advancement 79
7.3 Technical Enhancements and Setup 79
7.4 Challenges and Diagnostic Approaches 80
7.5 Concluding Remarks and Future Steps 82

8 Interference in BLE Protocol and GYW Auto-Certification **83**
8.1 Introduction to Interference in BLE 83
8.2 Sources of Interference 83
8.3 Engineering and Scientific Tools for Managing Interference 83
8.4 Concluding Thoughts on Interference Management 84
8.5 Interference Management in aRdent and aRdent 2 Glasses 84
8.6 Potential Improvements and Challenges in Interference Management Algorithms 85
8.6.1 Proposed Enhancements in Interference Management 85
8.6.2 Limitations in Current Interference Management Strategies 85
8.7 Auto-Certification Strategy for aRdent Glasses 86
8.7.1 Objective of Auto-Certification 86
8.7.2 Planned Testing Procedure 86
8.7.3 Detailed Testing Procedure for Data Integrity and Interference Assessment 87
8.7.4 ECM Testing Procedure During Data Transmission Phase 87
8.7.5 Representation and Setup of the ECM Testing Procedure 89
8.7.6 Impact of Logging on BLE Performance 90
8.7.7 Results 90

9 Conclusion **93**

Acronyms

GYW Get Your Way

HMD Head-Mounted Display

HMI Human-Machine Interaction

BLE Bluetooth Low Energy

HCI Host Controller Interface

ATT MTU Attribute Maximum Transmission Unit

BR/EDR Bluetooth Basic Rate/Enhanced Data Rate

GATT Generic Attribute Profile

ISM Industrial, Scientific, and Medical

PHY Physical Layer

AFHSS Adaptive Frequency Hopping Spread Spectrum

L2CAP Logical Link Control and Adaptation Protocol

ATT Attribute Protocol

GAP Generic Access Profile

MIC Message Integrity Code

LL Link Layer

CCCD Client Characteristic Configuration Descriptor

UUID Universally Unique Identifier

SM Security Manager

CI Connection Interval

IFS Inter Frame Space

DLE Data Length Extension

CE Connection Event

IAS Interference Avoidance Schemes

AFH Adaptive Frequency Hopping

CSA Channel Selection Algorithms

IoT Internet of Things

ST Supervision Timeout

PL Packet Loss

EMC Electromagnetic Compatibility

MCU Microcontroller

Chapter 1

Introduction

1.1 Get Your Way

1.1.1 The Company

Get Your Way is a Belgian-based start-up founded by three ambitious entrepreneurs during their studies at the University of Liège. The team started with its idea of connected smart glasses guiding cyclists and runners during their activities. In the context of the StarTech program, they got a first prototype that enabled them to win the first prize in the final of the competition.

Following that success, the founders joined VentureLab in March 2020. With the important backing of VentureLab, they developed the first version of the product, as well as the application for their first client. That is how Get Your Way was officially founded in 2020.

GYW develops aRdent, connected smart glasses designed to increase the comfort, safety, and efficiency [41] of workers in various industries. aRdent is different from most of its competitors because it tries to give a simple and comfortable user experience. This assisted reality (aR) [10] helps the user in completing tasks without causing distractions.

aRdent eyes is made of an optical module to be put within the peripheral vision of users, displaying information while they are performing their work. According to the application requirements, this information may be textual or graphical. In order to have a very simple and fast communication with other devices like computers, smartphones, or portable keypads, using BLE, aRdent are controlled through a Bluetooth API [5]. Therefore, it is possible to offload the computing power available and the software's functionalities onto another device.

Despite these accomplishments, Get Your Way continued its journey with its official launch. This strategy distinguished the company in a market dominated by tech giants.

In 2022, Get Your Way received financing from industrial and Belgian actors and entrepreneurs, something that greatly increased its ability to develop its product. And so, thanks to the Walloon technology accelerator WSL, the company kept on growing.

In January 2023, Get Your Way achieved significant milestones with the development of the Minimum Viable Product for its aRdent smart glasses. The latter created the conditions for a first on-site deployment and participation in numerous projects with Pole Mecatech.

Today, Get Your Way continues to push the envelope. The team is on the lookout for projects where they can replicate the success of this solution in general. He sets his focus on becoming the simplest and most valuable assisted reality device at a valuable price.

1.1.2 The Product

The primary product developed by Get Your Way (GYW) is the aRdent smart glasses. These glasses are designed as a versatile Head-Mounted Display (HMD) that can provide information directly in the user's field of vision with maximal comfort.



Figure 1.1: aRdent smart glasses overview.

The aRdent smart glasses are designed based on several must-haves. The first is a focus on comfort. Professionals have to wear it for long periods. For example, a delivery person should be able to wear the glasses the entire workday without discomfort.

Second is efficiency. Being a wearable connected device, the smart glasses have to be small in size and highly efficient. The glasses last for at least 4 hours continuously with a single small battery. An impressive feature that the aRdent glasses have is the fact that, after a battery has run out, it can be replaced very quickly, causing no down time.

Lastly, it is designed keeping user-friendliness in mind. It should provide a simple and straightforward HMI that makes it easy for professionals to adopt and use.

As seen in Figure 1.1, aRdent smart glasses have been designed with simplicity and minimalism. This approach ensures that the glasses are lightweight and comfortable for long-term wear.

1.1.3 Use Cases Example

Order preparation, or picking, can be a complex task. It involves picking in the warehouse a number of items, each identified with its position, barcode, batch number and count. Information handling when picking items proves to be difficult and, in most cases, will require a paper list. aRdent smart glasses solve this problem by displaying the needed information in the user's field of vision, making the order assembly process hands-free.

Either when the worker is walking, or when using a forklift, they can focus on picking items without the need to check the paper list periodically. Each kind of item is picked one after the other, so no item is missed and a high safety level is maintained. As it can

be seen in the display of Figure 1.2, all needed details are shown, such as the location of the item, code of the item, its description and quantity. The worker can quickly see this information and start working right away. If the quantity he needs is not available, they can easily change the quantity with the digital keypad.

When the picking is over, the list can be uploaded on the company's system directly, so the error possibility generated by transcriptions is removed, and digitalization can occur right away.

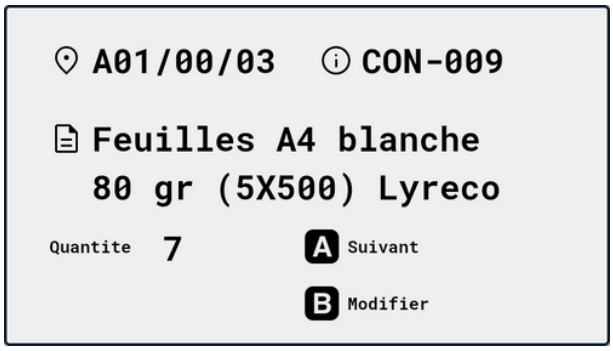


Figure 1.2: Display example shown on aRdent smart glass for order assembly.

1.2 Project Statement and Objectives

The objectives of this research project are threefold. The first is to optimize low-rate data transfer via Bluetooth to be able to increase throughput. A greater throughput allows for a wider range of applications for smart glasses and enables the transfer of larger files without significantly degrading the user experience. Additionally, higher throughput and lower latency improve the responsiveness of command transmission, which enhances and improves the user experience.

The second objective is to optimize energy utilization to extend the operational lifespan of devices. As smart glasses are worn on the head and rely on a battery, better energy efficiency will enable users to enjoy longer usage sessions, thus enhancing the overall user experience.

The third objective is focused on optimizing the stability of Bluetooth connections to increase reliability and improve the user experience. A stable and fast connection ensures that the glasses can be used seamlessly and comfortably.

1.3 Approach and Methodology

To optimize BLE for the aRdent smart glasses, a systematic methodology was followed. This methodology is comprised of distinct stages, each one derived from the findings of the preceding stage, ultimately leading to the practical realization of theoretical findings.

The project started with an important first step: the review of basic principles that operate behind Bluetooth and its low-energy versions. This included reading technical and multimedia documentations in order to derive baseline knowledge about Bluetooth technologies. The detailed working of BLE, and the improvements it brought over conventional, classic Bluetooth, were studied.

An in-depth analysis of available literature was done in this stage to gain knowledge out of some scientific articles dealing with BLE optimization techniques for different versions. This stage consisted of the reading of some academic paper systematically, in order

to identify the already applied optimization methods along with their corresponding results. A BLE versions comparison study was also conducted to demarcate the scope of improvements.

With the theoretical bounds well established, the research proceeded towards making informed predictions about BLE performance. This was done by predicting the outcomes of optimizing some of the documented BLE parameters. Hypotheses were generated about the potential improvements in data transfer efficiency and power consumption.

The changes were suggested, and actual experiments were conducted to test the theoretical hypothesis. A cycle of iterations was performed to refine the parameters based on differences between expected and observed results.

The last step of the research methodology was a synthesis of the findings. The practical test results were compared to the theoretical predictions to measure the efficiency of the optimization. The implications that the findings have analyzed for understanding the future implications of the development of the BLE technology in wearable devices.

Chapter 2

Product Description

2.1 Hardware Description

The performance of the aRdent glasses comes from the embedded Renesas RZ/A2M MCU [66]. This MCU utilizes the power of an ARM Cortex-A9 core, capable of reaching speeds up to 528MHz, and includes a dedicated DRP to enhance processing for image-related or AI-centric tasks. The device is equipped with a non-expandable 4MB of onboard SRAM [84], complemented by an additional 64MB of Macronix flash memory externally provisioned for both firmware [45] and application data. For connectivity, it employs the BlueNRG-M0A BLE module from ST Microelectronics, which aligns with BLE 4.2 protocols but lacks independent processing and memory, thereby relying on the primary MCU to handle security operations through an SPI [81] link. A JTAG port is also integrated for developmental purposes. However, it is not designed for consumer use as it necessitates a physical cable connection. Details of the MCU and connectivity components are illustrated in Figures 2.1, 2.2, 2.3 and 2.4 .

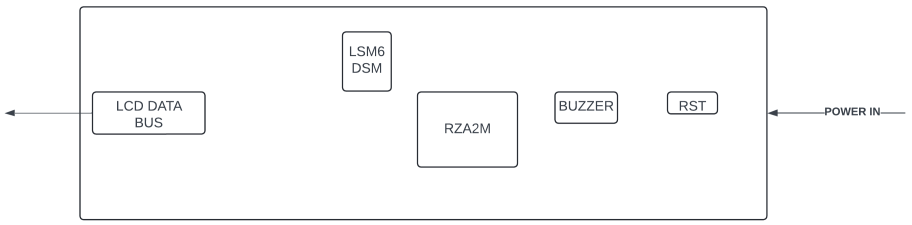


Figure 2.1: Design on the MCU side.

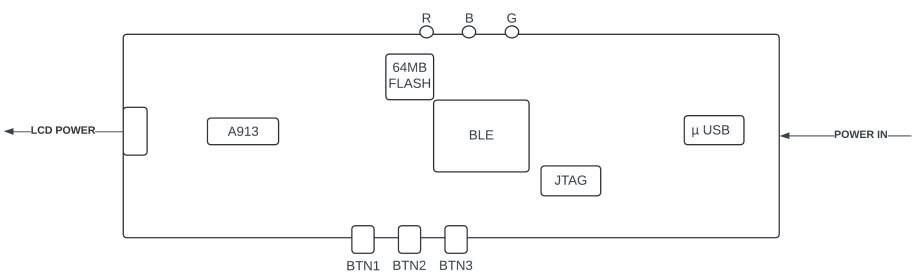


Figure 2.2: Design on the non-MCU side.

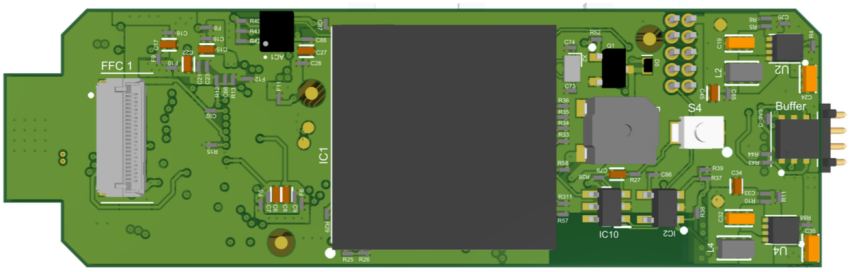


Figure 2.3: 3D rendering on the MCU side.

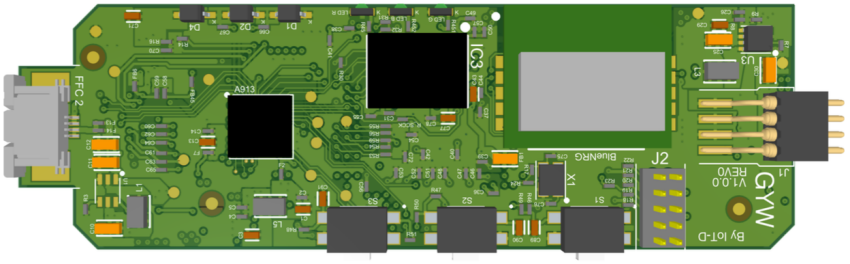


Figure 2.4: 3D rendering on the non-MCU side.

Bluetooth Module Overview

The aRdent glasses are powered by the BlueNRG-M0 Bluetooth module, a single-mode network processor that adheres to the Bluetooth 4.1 standard. This module serves as the communication core for the glasses, enabling both master and slave roles within the BLE protocol and supporting key features as illustrated in Figure 2.5:

- Role versatility with support for central, peripheral, observer, or broadcaster GAP roles.
- Comprehensive ATT/GATT client and server capabilities for flexible data management.
- Robust security protocols through the SM [80], including privacy, authentication, and authorization.
- Efficient data handling via the L2CAP [60].
- Reliable encryption and decryption at the Link Layer with AES-128 [4].

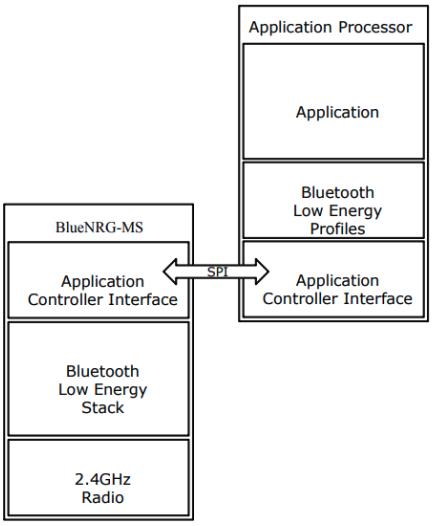


Figure 2.5: Bluenrg_ms application block diagram.

BLE Module Consumption

The energy efficiency of the aRdent glasses' BLE module was closely examined, yielding valuable insights into the device's power management. The key findings are summarized in Table 2.1, which details the power consumption components:

Component	**Power Consumption**
Display Average	525mW
MCU at 55MHz Reduction	355mW saved
BLE Off	0.051W saved (10mA at 5.1V)
Display Brightness Reduction	0.051W saved
Baseline (BLE Off, Display Off at 66MHz)	0.510W
Display Impact	0.559W (with BLE On at 66MHz)
BLE Impact	0.051W (with Display Off at 66MHz)

Table 2.1: Summary of power consumption components for aRdent glasses.

In a scenario where the total power draw is 1.120W with the display and BLE active at 66MHz, the display consumes approximately 49.91% of the total energy consumed, while the BLE module accounts for 4.55% of the total energy use. This delineation underscores the BLE module's comparative efficiency within the device's overall power architecture.

2.2 Software Implementation Details

Firmware Overview

The firmware for the aRdent smart glasses is built on the Free Real-Time Operating System (FreeRTOS). This system begins with a bootloader that checks for updates available via Bluetooth and applies them if possible, before launching the main operating system.

The firmware runs two primary tasks continuously. The first task is responsible for initializing the peripherals. After initialization, it signals success by blinking the green LED, or signals an error by blinking the red LED. The second task is dedicated to managing Bluetooth communications. It waits for notifications from the Interrupt Service

Routine (ISR), which is triggered whenever a Bluetooth packet is ready to be processed, and then handles the packet accordingly.

This structure ensures that the system efficiently manages peripheral initialization and Bluetooth communication using the FreeRTOS framework.

aRdent Data Transfer

We delve into the specifics of how the aRdent smart glasses communicate with external devices via Bluetooth. This advanced capability enables the transmission of various data types, empowering users with real-time information and control. Key aspects of data transfer, including data types, their respective sizes, and command specifics, are outlined below:

- **Text Display:** Text data is UTF-8 [92] encoded and sent to the Data Characteristic. Each message is followed by a control command that specifies the display parameters. The control command consists of 1 byte for the instruction type, 4 bytes each for horizontal and vertical positions, 3 bytes for font selection, 1 byte for font size, and 4 bytes for color, totaling 17 bytes.
- **Icon Display:** Icons use predefined names written to the Data Characteristic. The control command for displaying an icon includes 1 byte for the instruction, 4 bytes each for horizontal and vertical positions, 8 bytes for color, and 1 byte for scale, amounting to 18 bytes.
- **Image Display:** Images, either predefined or custom, are sent similarly to icons. The control commands mirror those of icons, accommodating the same sizes for positioning, color, and scaling.
- **Basic Commands:** Commands for screen manipulation, such as turning on the screen or resetting it, typically require only the control byte (1 byte) and, in some cases, additional parameters like color (8 bytes for screen reset).
- **Rectangles:** To draw rectangles, a control command comprising 1 byte for the instruction, 4 bytes each for position, 2 bytes each for width and height, and 4 bytes for color is used, totaling 15 bytes.
- **Spinners:** Displaying a spinner involves sending the spinner's name to the Data Characteristic and a control command that includes 1 byte for the instruction, 4 bytes each for position, 4 bytes for color, 1 byte for scale, and 1 byte for the animation timing function, summing up to 15 bytes.

The BLE protocol for the aRdent 1 glasses necessitates dividing larger payloads into blocks of 20 bytes for the Data Characteristic and constrains the Control Characteristic to 20 bytes per instruction due to their BLE module. This detailed breakdown clarifies the data management and display capabilities provided by the aRdent Bluetooth API, showcasing the versatility and precision of data transmission to these innovative smart glasses.

Chapter 3

Principles and Implementation of Bluetooth and BLE

3.1 Understanding Bluetooth and BLE

The primary distinction between classic Bluetooth and BLE lies in their operational paradigms and consumption profiles. Classic Bluetooth, or Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), was designed to facilitate continuous wireless connections with high data throughput. This makes it ideal for applications requiring steady streams of data, such as file transfers, audio streaming for headsets, or voice communication for hands-free calls.

On the other hand, BLE was introduced with the Bluetooth 4.0 version and is specifically tailored for applications that need to transmit small amounts of data intermittently while conserving energy. This low power consumption allows devices to run for months or even years on a small battery.

From a technical standpoint, these two technologies differ in various aspects:

- **Power Consumption**: BLE is optimized for low power consumption, using a fraction of the power required by classic Bluetooth during idle periods, which significantly extends battery life.
- **Communication Protocol**: BLE operates on a different protocol stack [74] that is much simpler than classic Bluetooth, with a reduced overhead that allows for quicker connections, which are established in a few milliseconds.
- **Data Throughput**: Classic Bluetooth offers higher data throughput compared to BLE, making it suitable for applications like streaming multimedia where large amounts of data are transferred continuously.
- **Range and Frequency**: While both technologies operate in the 2.4 GHz ISM band [53], BLE typically has a shorter range but can use frequency hopping to combat interference [93], making it robust in crowded wireless environments.
- **Application Profiles**: BLE does not support the wide range of profiles available with classic Bluetooth, focusing instead on a generic attribute profile (GATT) [49] which is more suited to its use-cases.

Understanding the differences between these two branches of Bluetooth technology is crucial for optimizing device performance for the intended application, whether it be for continuous data streaming or for the low-power, periodic transfer of small data packets.

3.2 Communication Channels

BLE operates within the Industrial, Scientific, and Medical (ISM) radio bands, occupying frequencies between 2400.2 and 2483.5 MHz.

This range is segmented into 40 distinct 2-MHz channels, numbered 0 through 39. This channelization is designed to mitigate potential signal interference with other devices sharing the ISM frequencies, such as traditional Bluetooth and Wi-Fi [94] technologies.

Specifically, the spectrum's tail end is reserved for primary advertising purposes, with channels 37 (2402 MHz), 38 (2426 MHz), and 39 (2480 MHz) dedicated to this role. Devices have the flexibility to broadcast their presence on any combination of these three channels, with the capability to be configured to limit advertising to only certain selected channels.

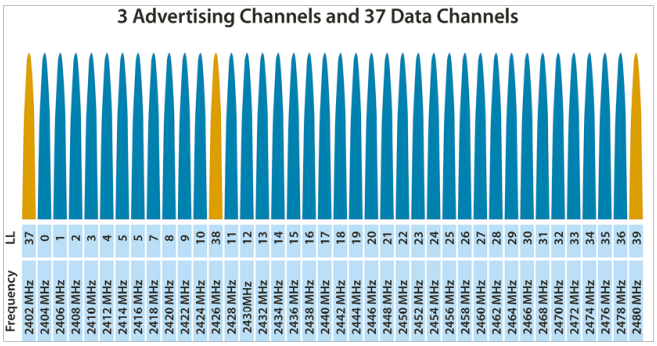


Figure 3.1: Distribution of BLE channels in the 2.4 GHz spectrum.

3.3 Protocol Stack Layers

The BLE Protocol Stack is an architecture that orchestrates the operation of BLE devices. Illustrated in the Figure 3.2, the stack is segmented into the Controller, the Host, and the application each containing multiple layers with distinct functionalities.

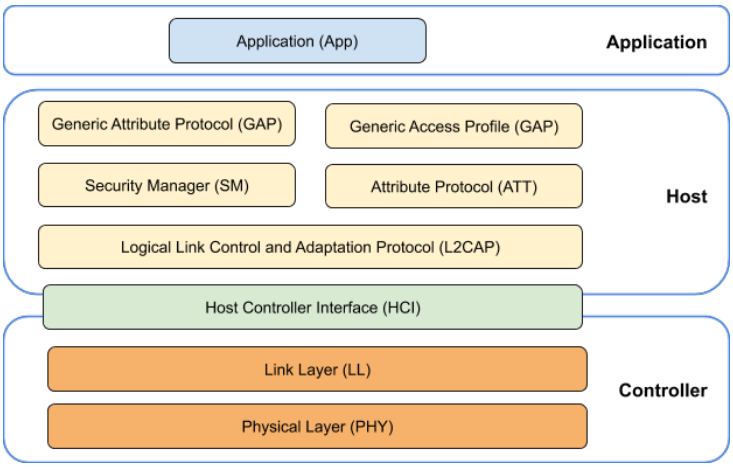


Figure 3.2: The BLE Protocol Stack.

3.3.1 Controller

Physical Layer (PHY): The Physical Layer [73] is tasked with the modulation and demodulation of radio signals, crucial for BLE's adaptive frequency hopping spread spec-

trum (AFHSS) [3], which mitigates interference and enhances robustness against multipath fading.

Link Layer (LL): The Link Layer [59] implements the BLE Link Layer protocol, managing role switching between Central and Peripheral, connection initiation, parameter negotiation, encrypted data transfer, and supervision timeout [88] for link loss termination.

3.3.2 Host Controller Interface (HCI)

The HCI [51] provides a standardized command interface for accessing the BLE baseband capabilities and controlling various aspects of the lower layers of the BLE stack, such as link control, baseband, and link manager. Essentially, it serves as a bridge between the host system, which could be a smartphone or computer, and the BLE module's controller. This separation allows for the host to send commands and process data without needing to manage the intricate details of BLE communication protocols, contributing to ease of development and system integration.

3.3.3 Host

Logical Link Control and Adaptation Protocol (L2CAP): L2CAP provides multiplexing of higher-level protocols, segmentation and reassembly of packets, and quality of service (QoS) [77] information, crucial for data encapsulation in variable-sized payloads [29].

Attribute Protocol (ATT): ATT lays the groundwork for client-server communication in BLE, defining a structured framework for data storage and transmission, which serves as the backbone for the GATT.

Security Manager (SM): At the core of BLE's security architecture, the SM administers pairing methods, key distribution, and encryption routines, employing Elliptic Curve Diffie Hellman (ECDH) for key agreement and AES-CCM for data protection.

Generic Access Profile (GAP): GAP [48] defines how BLE devices find and connect to each other, including advertising types, connection steps, security levels, and roles, which determine the device's visibility and interaction abilities.

GATT: GATT operates on top of ATT, organizing and managing services and characteristics, which are the basic data elements for BLE communication, allowing applications to exchange data in a standardized way.

3.3.4 Applications

The application layer [8] interacts directly with GATT, translating user actions into BLE operations and managing the presentation, manipulation, and storage of GATT data in a user-centric format.

3.4 Detailed BLE Packet Structure

BLE's efficient communication relies on its well organized data packets. As shown in in Figure 3.3, the structure of these packets is crucial for BLE operations.

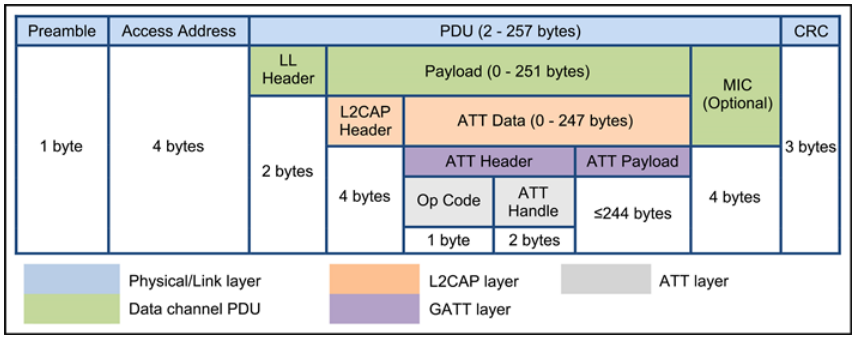


Figure 3.3: Detailed structure of a BLE packet.

Preamble: An important synchronization and modulation identifier that is 1 byte long. It signals receivers to demodulate the data correctly.

Access Address: A 4-byte unique identifier for each BLE advertising channel or connection, ensuring data is correctly recognized and filtered by the intended recipient.

CRC: A 3-byte Cyclic Redundancy Check sums up this structure, offering error checking to ensure data integrity [39] during transmission.

LL Header: This 2-byte Link Layer Header [28] contains control fields that delineate the structure and status of the data payload, such as its length and message integrity information.

MIC (Optional) [65]: For encrypted communications, an optional 4-byte Message Integrity Check may follow, providing authentication and integrity verification.

L2CAP Header: Positioned above the Link Layer, the 4-byte L2CAP Header specifies the length and channel ID, directing the data payload to the correct higher-layer protocol.

ATT Data: Within the L2CAP payload, the ATT Data is segmented into a 1-byte Op Code and a 2-byte Attribute Handle, followed by the ATT Payload, which can carry up to a maximum number of bytes defined by the ATT MTU parameter, allowing for rich and complex client-server interactions.

The packet layout is a detailed plan that ensures reliable, efficient, and secure data transmission in BLE networks, meeting the diverse needs of modern wireless communication.

3.5 Connection Establishment

The establishment of a BLE connection is a systematic process that involves both the central device (often referred to as the master) and the peripheral device (also known as the slave). The procedure ensures secure and efficient communication between the two entities. Below is an overview of the steps involved, as illustrated in Figure 3.4:

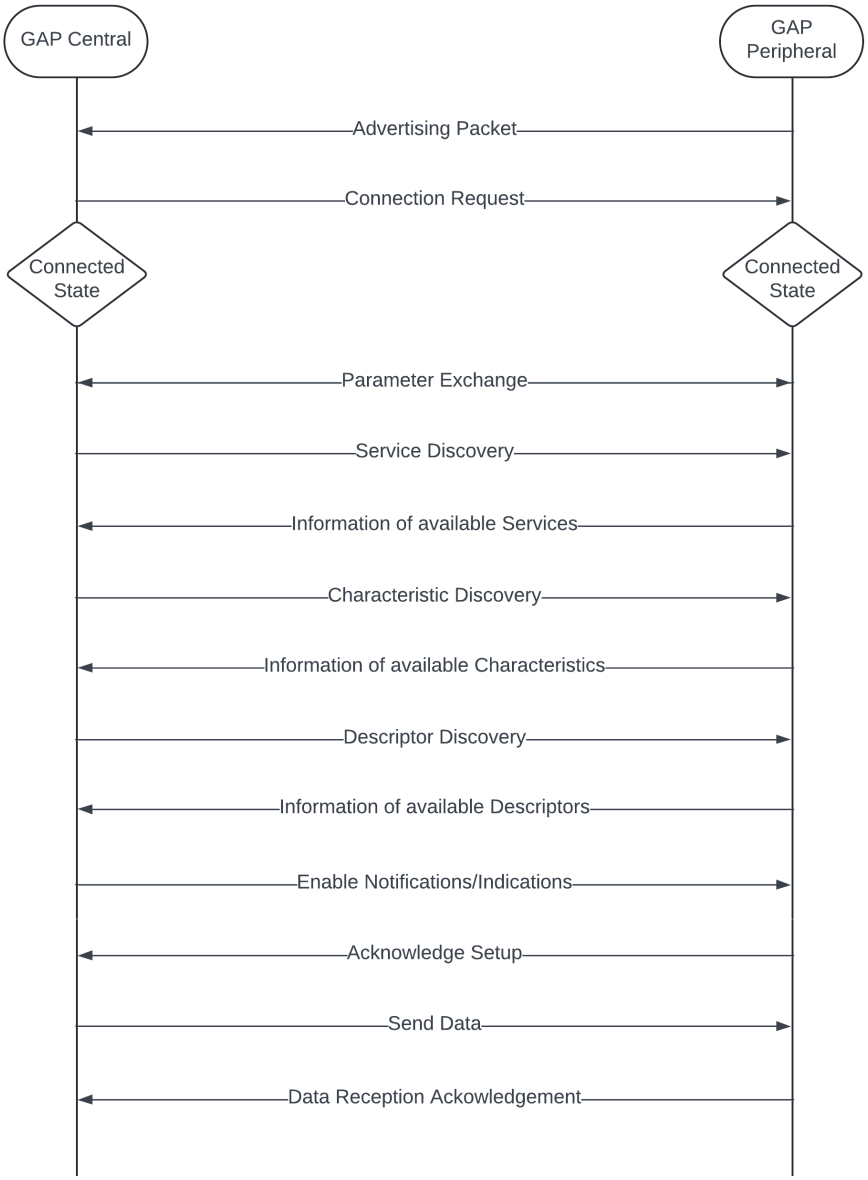


Figure 3.4: Illustration of the BLE connection procedure.

1. The GAP Central (master) device begins by scanning for available devices, specifically looking for advertising packets in active mode.

2. Once the desired advertising packet is received, the central device sends a connection request to the peripheral with the specific name and connection parameters.

3. The GAP Peripheral (slave) device, after broadcasting the advertising packet and receiving a connection request, transitions to a connected state, ceasing to advertise.

4. Following a successful connection, the central device halts the scanning process.

5. Both the GATT Client (in the central device) and the GATT Server (in the peripheral device) then exchange settings to improve their communication.

6. The GATT Client starts the service discovery process to determine the services provided by the GATT Server.

7. The GATT Server responds with the UUIDs [91] of the available services after receiving the discovery request.

8. Following service discovery, the GATT Client may further request to discover characteristics for the identified services. The GATT Server responds with the characteristics, each including its UUID, properties, handle, and optionally its initial value.

9. The GATT Client may also request to discover descriptors associated with the characteristics. The GATT Server responds with the descriptors, including their UUIDs, handles, and value information.

10. The GATT Client writes to the Client Characteristic Configuration Descriptor (CCCD) on the GATT Server to enable notifications or indications for specific characteristics.

11. The GATT Server acknowledges the write request to the CCCD, confirming that notifications or indications are enabled for the specified characteristics.

12. The GATT Server, now in a fully established communication state, can send notifications or indications to the GATT Client whenever the characteristic value changes.

13. The GATT Client receives the data updates, completing the connection establishment phase and enabling ongoing communication.

3.6 Data Transfer in BLE

BLE is a wireless communication technology designed for short-range communication, ideal for applications requiring periodic or intermittent data exchange with minimal power usage. The core concept of BLE data transfer involves "characteristics", which are foundational elements in the BLE version for managing data.

Characteristics Explained: In conceptual terms, a characteristic can be thought of as a distinct data point. It's akin to a variable in programming that holds a value. For instance, if the BLE device is a smart thermostat, one characteristic might be the current room temperature, while another could be the target temperature set by the user.

In computer science terms, a characteristic is a data structure that contains a value and several descriptors that describe the value's properties. These properties include permissions (e.g., read, write, notify) and metadata (e.g., whether the value represents temperature in Celsius or Fahrenheit).

Each characteristic has a UUID, which is a standardized format for a string ID used to uniquely identify information. It ensures that data points are distinguishable from one another across different devices and applications.

Data Transfer Mechanism: The process of data transfer in BLE involves interaction with these characteristics within a predefined structure known as the GATT. The GATT specifies how two BLE devices exchange data using concepts of "services" and "characteristics".

Here is a simplified step-by-step explanation of how data transfer typically occurs in BLE:

Services and Characteristics: A service is a group of characteristics. Each service has a UUID, making it recognizable across different devices and applications.

Establishing a Connection: Two BLE devices establish a connection. One acts as the "central" device and the other as the "peripheral" device. The central device discovers available services and their characteristics on the peripheral device.

Interacting with Characteristics: The central device can read or write to available characteristics based on permissions. Reading a characteristic asks the peripheral for its current value. Writing updates the peripheral with new information.

Notifications and Indications: Characteristics with the notify property can alert the central device when their value changes, providing real-time updates. Indications are similar but need the central device to acknowledge that the message was received.

3.7 BLE parameters

3.7.1 Connection Interval

The Connection Interval (CI) is a crucial parameter within BLE technology that defines the duration between two successive connection events [31]. It is essential for balancing energy efficiency and communication responsiveness. The CI determines when data is exchanged between two devices in a BLE network. While each connection is strictly one-to-one, a central device can maintain multiple such one-to-one connections simultaneously, as depicted in Figure 3.5.

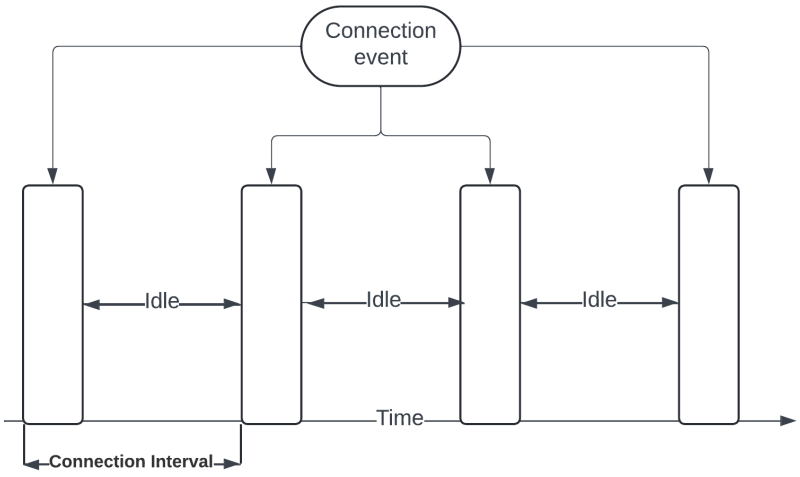


Figure 3.5: The CI principle.

This interval is adjustable, beginning at a minimum of 7.5 milliseconds and increasing in increments of 1.25 milliseconds. This framework allows for exact control over the timing of Communication Events. In these events, the central device starts the transmission and the peripheral device prepares to receive it, which is then followed by a response from the peripheral to the central device.

In essence, the CI is a dynamic and negotiable value post-connection that finely tunes the balance between power conservation and data transfer speed, ranging from 7.5ms up to 4 seconds, offering a variety of operational modes fit for different application requirements.

3.7.2 Inter Frame Space

The Inter Frame Space (IFS) [89] in BLE communication is essentially a pause or gap between the transmission of consecutive data packets. This brief interval is crucial for ensuring that devices have sufficient time to process each packet before receiving the next one. In BLE, the standard duration for the IFS is precisely 150 microseconds. This specific time frame helps to maintain orderly and efficient communication between BLE devices, allowing for the necessary processing and preparation for subsequent data packet receptions or transmissions.

3.7.3 Attribute Maximum Transmission Unit

The ATT MTU [11] is an important parameter within the BLE version that outlines the maximum size of data that the application layer can transfer during a single attribute protocol (ATT) [12] operation. It essentially dictates the maximum data payload that both transmitting and receiving devices can process and store in their buffers [38] during communication. Initially, the ATT MTU is set to a default size of 23 bytes, which includes the operation code and the attribute handle, allowing it to fit easily within a single Link Layer (LL) packet. This default setting permits an application data payload of 20 bytes, factoring in the 3-byte overhead for the ATT header, as illustrated in Figure 3.6.

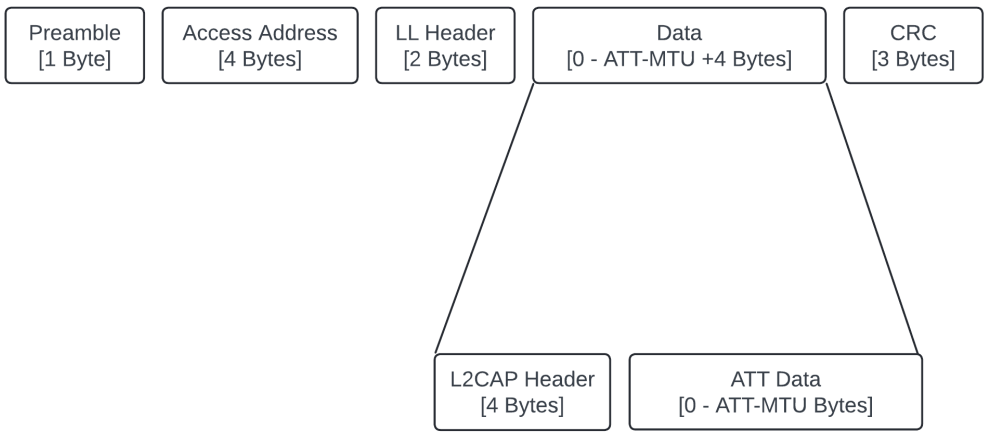


Figure 3.6: ATT packet format

However, the ATT MTU can be extended up to 247 bytes, enhancing data throughput by enabling longer payloads across multiple packets. It's important to note that leveraging the Data Length Extension (DLE) [40] feature allows for even more efficient use of available bandwidth, optimizing payload sizes to fit within the constraints of the MTU and LL packet sizes.

The process of determining the effective ATT MTU involves a negotiation phase after establishing a connection, where both client and server exchange their supported MTU sizes via the Exchange MTU Request/Response mechanism. The operational ATT MTU

for the session is then set based on the minimum value supported by both devices. This negotiation ensures compatibility and optimizes communication efficiency, balancing between the desire for larger data payloads and the need to minimize energy consumption.

3.7.4 Connection Event

In BLE communications, a connection event is a regular exchange of data between the Central and Peripheral devices. It happens at specific intervals, known as Connection Intervals (CI). During a connection event, the Central device sends a packet to the Peripheral device, and the Peripheral responds with its own packet. This exchange can include actual data, control information, or simply be an empty packet. Connection events are crucial for keeping the devices synchronized and ensuring the connection is maintained.

Importantly, a single CE can facilitate the exchange of multiple packets. This capability is significant for optimizing data throughput and efficiency. Specifically, the BlueNRG-MS chip exemplifies this by allowing up to 8 packets to be transmitted within a single CE, thus significantly enhancing the potential for data exchange within each interval.

Moreover, it is important to note that extending the CI can help save battery life by reducing the frequency of communication. However, this benefit must be balanced with the risk that packet loss could cause delays until re-transmission. This situation requires careful planning in BLE system design, especially when making dynamic adjustments to the length of connection events or using strategies to extend event lengths.

3.7.5 Data Transmission Modes: Write With and Without Response

In BLE communication, data can be transmitted between devices using two primary methods of writing: “Write with Response” and Write without Response.

The “Write with Response” method ensures that the sender receives an acknowledgment from the receiver once the data packet has been successfully delivered and processed. This mode provides reliability in data transmission, ensuring data integrity by confirming receipt.

Conversely, Write without Response allows for data to be sent to the receiver without waiting for any acknowledgment. This method increases the speed of data transmission but does not guarantee that the data has been successfully received or processed by the target device.

Both of these transmission modes offer distinct advantages and trade-offs in terms of data transfer speed and reliability. These methods will be explored in greater detail later in this work, examining their impact on application performance and efficiency in BLE communications.

3.8 Theoretical Bluetooth Throughput Limitations

Understanding why theoretical BLE speeds are not fully achievable in practice requires a comprehensive look at the various factors that influence actual data throughput. While the Bluetooth version outlines data rates of 1 Mbps (LE 1M PHY [57]) and 2 Mbps (LE 2M PHY [58]) as the peak speeds for radio transmission, several inherent limitations prevent these rates from being realized for application data throughput.

Firstly, there is a restriction on the number of packets that can be transmitted within each CI. This limit restricts the amount of data that can be exchanged during these windows, directly affecting throughput.

Secondly, the IFS, a mandatory delay of 150 microseconds between packets, further reduces the efficiency of data transmission. This delay is crucial for device synchronization and data integrity but comes at the cost of continuous data flow.

Additionally, BLE devices are required to send "empty packets" when there's no application data to transmit. These packets, while necessary for maintaining a connection, occupy transmission time without conveying useful information.

Moreover, packet overhead presents another challenge. Not all bytes in a BLE packet carry user data. A significant portion is used for headers and other protocol information, reducing the payload capacity of each packet.

Environmental interference also plays a pivotal role in diminishing BLE throughput. Wireless signals can be degraded by obstacles, other wireless devices, and various forms of electronic interference in the surrounding area. This degradation can lead to retransmissions and further reduce effective throughput.

Lastly, the hardware configuration of the involved devices influences throughput capabilities. The processing power, antenna quality, and BLE stack implementation can all vary significantly, affecting how efficiently data is transmitted and processed.

Chapter 4

Preliminary Performance Evaluation

4.1 Tools

In this section, the tools used to optimize the aRdent smart glasses are described. These tools include both hardware and software that help evaluate and improve the performance of the BLE module.

SEGGER J-Link

The SEGGER J-Link is a widely-used debug probe known for its high performance and reliability. It is used for debugging and programming the firmware on microcontrollers. The J-Link supports a variety of processors and microcontrollers, providing an essential interface for development and troubleshooting.

As shown in Figure 4.1, the SEGGER J-Link debug probe has a compact and robust design, making it suitable for various development environments.



Figure 4.1: SEGGER J-Link Debug Probe.

JLinkRTTViewer

JLinkRTTViewer is a tool that allows real-time terminal output from the microcontroller to be viewed on a PC. This is particularly useful for debugging and logging purposes, providing insights into the running state of the firmware.

As shown in Figure 4.2, the JLinkRTTViewer interface provides a clear and detailed view of the terminal output, facilitating effective debugging and analysis.

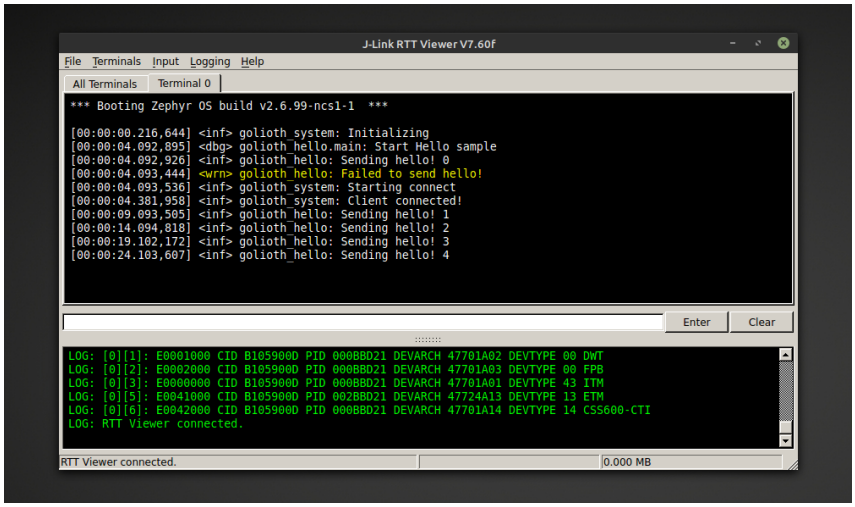


Figure 4.2: JLinkRTTViewer Interface.

PacketLogger

PacketLogger is a software tool from Apple used to capture and analyze Bluetooth packets. It helps in understanding the communication between devices and diagnosing any issues related to data transfer, and it is only available for Apple devices.

As shown in Figure 4.3, the PacketLogger tool provides a detailed view of the captured Bluetooth packets, allowing for in-depth analysis and troubleshooting.

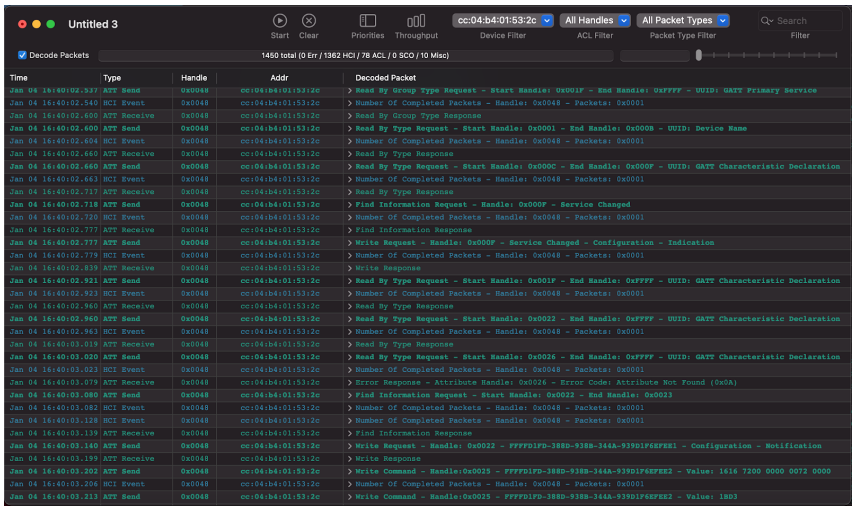


Figure 4.3: PacketLogger Tool.

Wireshark

Wireshark is a powerful network protocol analyzer used for network troubleshooting and analysis. It can capture and display the data traveling back and forth on a network in real-time, including Bluetooth traffic.

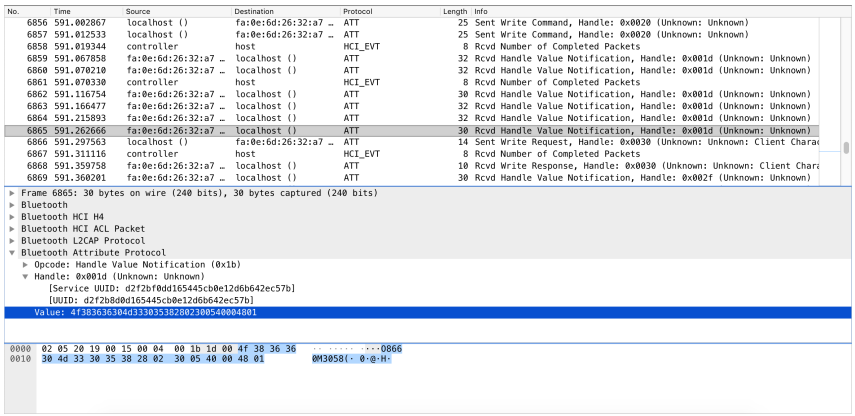


Figure 4.4: Wireshark Network Protocol Analyzer.

ESP32-S3 with Arduino

For some applications, GYW uses an ESP32-S3 chip with the Ardunio framework. This dual-core XTensa LX7 MCU runs at 240 MHz and includes 512 KB of internal SRAM. It also supports 2.4 GHz 802.11 b/g/n Wi-Fi and Bluetooth 5 (LE) connectivity. The ESP32-S3 has 45 programmable GPIOs and supports high-speed SPI flash and PSRAM, making it versatile for various applications.



Figure 4.5: Arduino with ESP32-S3.

Nordic nRF52840

The Nordic nRF52840 is a multiprotocol Bluetooth 5.4 System on Chip (SoC) that supports Bluetooth Low Energy, Bluetooth mesh, NFC, Thread, and Zigbee. It is built around a 32-bit ARM Cortex-M4 CPU running at 64 MHz with a floating point unit. The nRF52840 includes the ARM TrustZone CryptoCell cryptographic unit and supports various digital peripherals, making it suitable for complex applications requiring high security and multiple communication protocols.

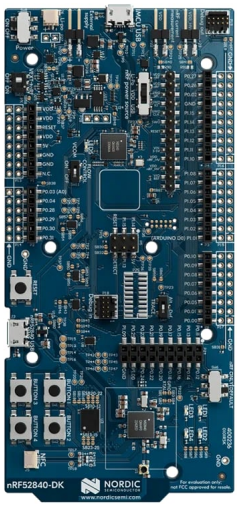


Figure 4.6: Nordic nRF52840 SoC.

Current Consumption Estimation Tool by ST Electronics

ST Electronics [36] has developed a sophisticated software utility known as the Current Consumption Estimation Tool, depicted in Figure 4.7. This software provides users with the ability to select from a range of ST's BLE chips and to configure various parameters for simulation purposes. Users can hold certain parameters constant while varying others, enabling them to observe the resultant effects on several key performance metrics.

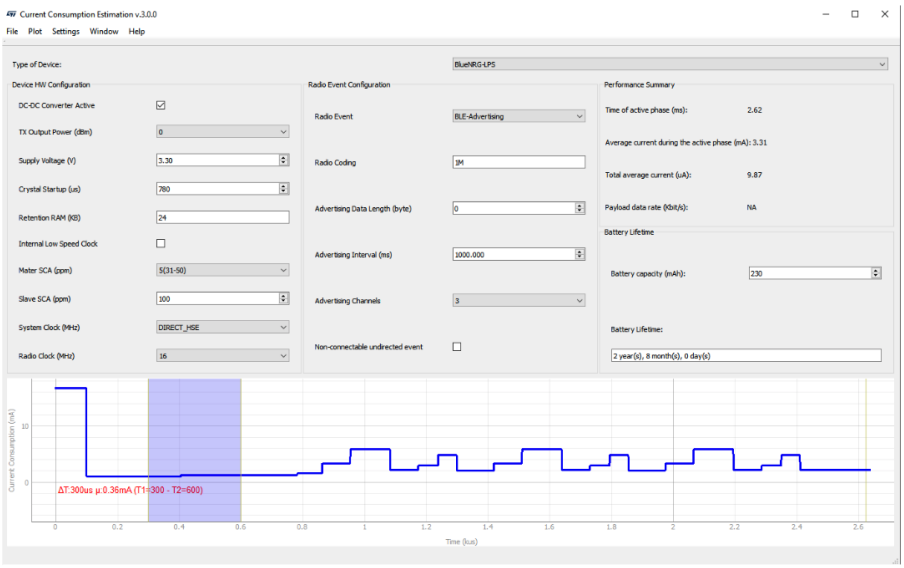


Figure 4.7: Current Consumption Estimation Tool by ST Electronics.

4.2 Initial Throughput

Before discussing the optimization parameters and improvements in this research, it is important to first establish a baseline for throughput. This initial throughput will act as a reference point, allowing for a comparative analysis of the improvements achieved.

At the start of this study, the throughput was measured under controlled conditions. On an Asus PC running a Linux operating system, the initial throughput recorded was

0.2 kilobytes per second (KB/s). In contrast, a measurement taken on a 2019 MacBook Pro yielded a slightly higher throughput of 0.3 KB/s.

These observed disparities raise intriguing questions: What factors contribute to the differences in throughput across devices? How can these rates be augmented? These questions, along with how they were investigated and the results, will be fully covered in the next sections of this thesis.

4.3 Modifiable BLE Parameters in aRdent 1

The BLE 4.1 chip in the aRdent glasses allows for changes to a few key settings that affect communication and power use. The adjustable settings are:

Connection Interval: This setting controls how often two BLE devices communicate when connected.

Write Modes: This setting lets you choose between "Write with response" and "Write without response", which affects the speed and reliability of data transmission.

These limitations highlight the chip's design focus on efficient and flexible communication. Therefore, optimization efforts for the aRdent glasses aim to make the best use of these adjustable settings.

4.4 Analysis of Throughput Across Different Devices

This section presents an empirical investigation into how fixed parameters namely, CI set to 30ms, ATT MTU at 23 bytes, and the "Write with Response" method impact the throughput of data transmission across devices with varying operating systems and programming languages. A notable aspect of this study is the observation of the maximum number of packets per CE, with the BlueNRG_MS chip supporting up to 8 packets.

4.4.1 Methodology

The experiment measured the time and associated throughput for transmitting a payload of 1000 bytes across different devices. The analysis tried to understand how the operating system, development framework, and device capabilities affect the achievable data rates.

4.4.2 Results

The following devices were tested:

- **Macbook Pro 2019 (MacOS 13.1, Python)**: Achieved a throughput of 0.2949 KB/s over 6.273 seconds, using 8 packets per CE.
- **iPhone 12 (iOS 16.6, Flutter)**: Achieved a throughput of 0.1412 KB/s over 13.10 seconds, using 4 packets per CE.
- **PC Asus Linux (Ubuntu 20.4 LTS, Python)**: Achieved a throughput of 0.29086 KB/s over 6.306 seconds, using 8 packets per CE.
- **OnePlus 6 (Lineage OS Android 13, Flutter)**: Achieved a throughput of 0.292 KB/s over 6.33 seconds, using 8 packets per CE.

4.4.3 Interpretation

The experiment reveals that, with fixed parameters, the throughput across devices is largely similar, suggesting that neither the operating system nor the programming language significantly influences the data rate. The crucial determinant of throughput under these conditions is the maximum number of packets per CE. This was particularly evident in the case of the iPhone 12, where the lower number of packets per CE (a factor not directly controllable) resulted in reduced throughput. It's also noted that different OSs might have distinct limitations on parameters like CI, as observed with MacOS, indicating an area for further investigation.

Chapter 5

Optimizing BLE Parameters

5.1 Optimizing the CI

5.1.1 Modifying the CI

At the beginning of establishing a connection, there's a crucial negotiation phase between the central and peripheral devices. This phase aims to agree on a CI that balances the needs for efficient data transmission with energy savings.

The CI's adaptability is key, allowing adjustments to meet the evolving demands of the operation or specific application needs. This project takes advantage of a feature from ST Electronics that enables the renegotiation of connection parameters, including the CI, at any time after the initial connection. The update connection parameter function is used right after establishing the connection to improve BLE communications for both efficiency and speed.

Typically, the central device has the final say in setting the connection parameters, including the CI, leaving the aRdent glasses with limited direct influence. However, to ensure flexibility for the aRdent glasses, the project uses the update connection parameter function. This strategy is important because it is impractical to customize the BLE stack for every possible central device. Therefore, using this function from the aRdent glasses side offers a practical way to influence connection settings, improving both performance and power management.

The ability to adjust connection parameters before establishing a connection varies by operating system. For example, Linux may allow changes to the central device's parameters through system commands, whereas macOS lacks an interface for pre-connection adjustments. This variance highlights the importance of the peripheral device's ability to use the 'update connection parameter' function. It effectively bypasses restrictions from specific operating systems, ensuring optimizations are possible, independent of the master device's operating system.

Implementation of Dynamic Connection Parameters:

Listing 5.1: BLE Connection Parameter Update Example

```
void GAP_ConnectionComplete_CB(uint8_t* peer_bdaddr, uint16_t conn_handle)
{
    debug_printf("Connection␣complete␣callback\r\n");

    // Request connection parameter update
    tBleStatus ret = aci_l2cap_connection_parameter_update_request(
        conn_handle,    // connection handle
        54,             // conn_interval_min (54 * 1.25 ms = 67.5 ms)
        55,             // conn_interval_max (55 * 1.25 ms = 68.75 ms)
        0,              // slave_latency (number of connection events that
            can be skipped)
        800             // supervision_timeout (800 * 10 ms = 8000 ms or 8
            seconds)
    );

    if (ret != BLE_STATUS_SUCCESS) {
        error_printf("Connection␣Parameter␣Update␣Request␣failed:␣0x%x\r\n",
            ret);
    } else {
        debug_printf("Connection␣Parameter␣Update␣Request␣sent␣
            successfully\r\n");
    }
}
```

Parameter Explanation:

- **conn_interval_min and conn_interval_max:** These parameters specify the minimum and maximum intervals between two consecutive connection events. Adjusting these values allows for balancing energy efficiency and data transmission delay.
- **slave_latency:** This parameter determines the number of connection events the aRdent glasses can skip without affecting the connection. A higher slave latency can improve power consumption but may increase data delay.
- **supervision_timeout:** This parameter specifies the maximum time between two received data packets before the connection is considered lost. A longer timeout is beneficial for devices that expect intermittent communication but increases the time to detect a lost connection.

Based on the JLinkRTTViewer [54] output, as shown in Figure 5.1, the successful execution of the connection parameter update is confirmed. The initial CI set at 12 transitions to 54 after the update. These values represent the number of 1.25 ms intervals, translating the intervals from 15 ms (12 * 1.25 ms) to 67.5 ms (54 * 1.25 ms) for the updated CI. This change illustrates a significant shift in the timing of data exchanges, indicating a strategic move to balance energy consumption against communication frequency.

```
00> [DEBUG] Subevent value: 1
00> [DEBUG] *** Connection complete ***
00> [DEBUG] Connection status = 0
00> [DEBUG] Connection role = 1
00> [DEBUG] Connection handle = 2049
00> [DEBUG] Connection interval = 12
00> [DEBUG] Connection latency = 0
00> [DEBUG] Connection sup timeout = 36
00> [DEBUG] Master clock accuracy = 1
00> [DEBUG] Connection complete callback
00> [DEBUG] Connection Parameter Update Request sent successfully
00> [DEBUG] Subevent value: 3
00> [DEBUG] *** Connection update complete ***
00> [DEBUG] Connection status = 0
00> [DEBUG] Connection handle = 2049
00> [DEBUG] Connection interval = 54
00> [DEBUG] Connection latency = 0
00> [DEBUG] Received data length: 2
00> [DEBUG] CPU clock: 440MHZ.
00> [DEBUG] Received data for handle 4
00> [DEBUG] attribute modified handle not managed : 5
00> [DEBUG] CPU clock: 55Mhz.
```

Figure 5.1: JLinkRTTViewver view after an update function call.

5.1.2 Strategies for Optimizing the CI

Determining the optimal strategy for modifying the CI involves considering whether adjustments should be initiated by the master, the slave, or the slave post-connection establishment. While the negotiation process at the start of a connection aims for a consensus on CI, allowing for subsequent adjustments by the slave via the 'update connection parameter' function provides flexibility and dynamic adaptability to changing requirements or conditions. This method requires careful timing and consideration of the impacts on both power consumption and data transmission latency.

Furthermore, when proposing a range for the CI, with a specified minimum and maximum interval, the ultimate selection within this range often aligns with the master device's operational objectives. For instance, on mobile devices, there is a tendency to favor the higher end of the CI range. This preference aims to maximize energy efficiency and extend device battery life, demonstrating a strategic compromise between maintaining a reliable connection and optimizing power usage.

5.1.3 Experimental Setup

To carefully evaluate the effects of CI adjustments, we designed a series of tests, each intended to show different aspects of BLE communication under varying CI settings. The tests were conducted with the following fixed BLE parameters across all experiments: ATT MTU size at 23 bytes, and utilizing the "Write with Response" mode for data transmission. These constants make sure that our findings are due to changes in the CI and not other factors.

Test Descriptions

The tests were structured to measure both the temporal efficiency and throughput capacity of BLE communication under different CIs. Here is a brief overview of the tests conducted:

1. **Display Interface Commands:** This test measured the time taken to send simple display commands like adjusting contrast and brightness settings. The goal was to assess the efficiency of basic BLE operations with small data payloads.

2. **Screen Drawing Operations:** This test measured the time taken to send commands to draw elements on a screen, such as text, icons, and color changes, which require larger data packets.

3. **Mixed Media Transfer:** This test measured the time taken to send a combination of text and graphical icons in various configurations, simulating a common use case in smart device interfaces to assess the communication overhead for mixed content types.

Test Setup

The testing protocol was executed using a MacBook Pro from 2019, serving as the central device to interface with the aRdent glasses under examination. Throughout the testing phase, BLE parameters were uniformly maintained to ensure consistency in the results. The ATT MTU was fixed at 23 bytes within the firmware of the BLE devices, and the "Write with Response" mode was persistently employed. This controlled setup allowed us to accurately assess the impact of the CI on the BLE communication process, removing potential variability from changing BLE parameters or different host hardware capabilities.

Assessment Metrics

The primary metrics for assessment will include:

- **Transmission Time:** The duration required to complete each test case, providing insight into the temporal efficiency of BLE communication under different CIs.

- **Data Throughput:** Calculated as the total amount of data successfully transmitted per unit of time, offering a measure of communication capacity.

5.1.4 CI Modification on Different Devices

The adjustment of the CI is subject to the constraints and versions of the device in use. While the CI typically starts at a minimum of 7.5 ms and increases in increments of 1.25 ms, device manufacturers may impose their own limits and increment steps.

For instance, when the aRdent glasses request an update of the CI on a Linux PC, the device faces no restrictions and can accept any value that meets the minimum threshold of 7.5 ms, incrementing in steps of 1.25 ms.

On Apple computers, the behavior depends on the macOS version. For macOS versions preceding Sonoma, the CI behaves similarly to that on Linux PCs, adhering to the same minimum value and increment steps.

However, with macOS Sonoma and beyond, the rules change. The minimum CI begins at 15 ms and increments are fixed at steps of 15 ms.

Android devices exhibit a behavior similar to Linux PCs, with the minimum CI set at 11.25 ms, and no restrictions on increment steps.

On iPhones, the parameters for a successful CI update are quite specific. The general guidelines for connection parameter requests on Apple devices are as follows:

- Peripheral Latency should not exceed 30 CIs.
- Supervision Timeout (ST) must be within 6 to 18 seconds.
- Interval Min should be at least 15 ms.
- Interval Min must be a multiple of 15 ms.
- One of the following conditions must be met:
  - Interval Max is at least 15 ms greater than Interval Min.
  - Interval Max and Interval Min are both set at 15 ms.
  - Interval Max does not exceed the product of (Peripheral Latency + 1) and 6 seconds.
- ST should be greater than the product of Interval Max and (Peripheral Latency + 1) times 3.

This behavior across different devices and operating systems highlights the importance of understanding device-specific BLE implementations when attempting to modify CIs for optimal performance.

5.1.5 Analysis of Test Results

Transmission Time

The conducted experiments produced useful data on the relationship between the CI and BLE transfer times for various screen configurations. The graphical analysis below shows the detailed interaction between CI values and the associated transfer times, revealing patterns and performance implications.

Figure 5.2 compares transfer times for a plain white screen and various text configurations. The trend suggests that more complex screen elements directly contribute to longer transfer times as the CI increases. This is indicative of the added data payload and processing required for rendering text and images, which escalates with the complexity of the screen content.

Figure 5.3 elucidates how the spacing of texts impacts the transfer time. Interestingly, multi-spaced texts exhibit a notable increase in transfer time, highlighting the overhead introduced by additional data formatting requirements.

Figure 5.4 focuses on combinations of white screen configurations with text and icons. It's observed that the inclusion of icons, either preceding or following the text, causes an uptick in transfer times, suggesting a discernible impact of graphical data on transmission efficiency.

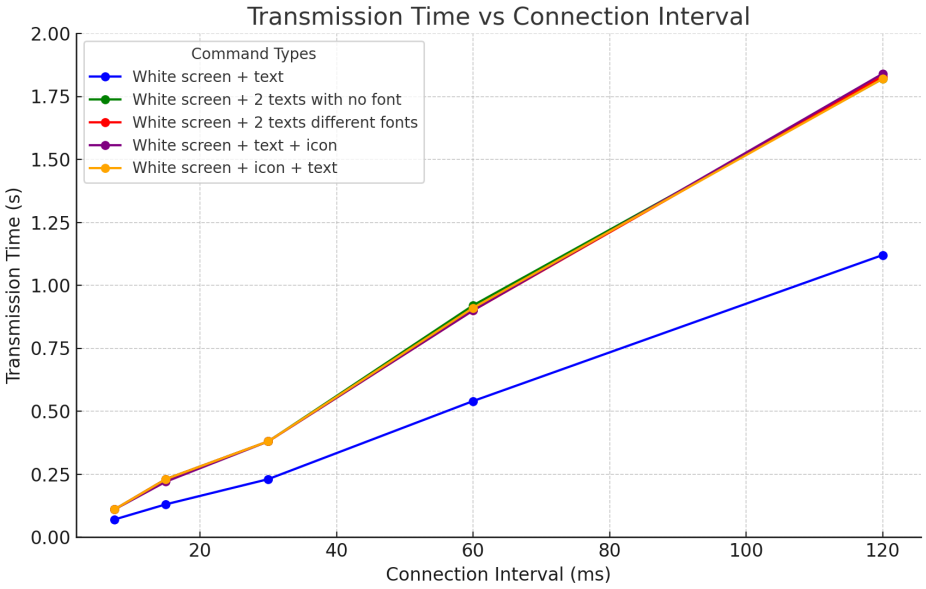


Figure 5.2: BLE Transfer Time for Various Screen Configurations.

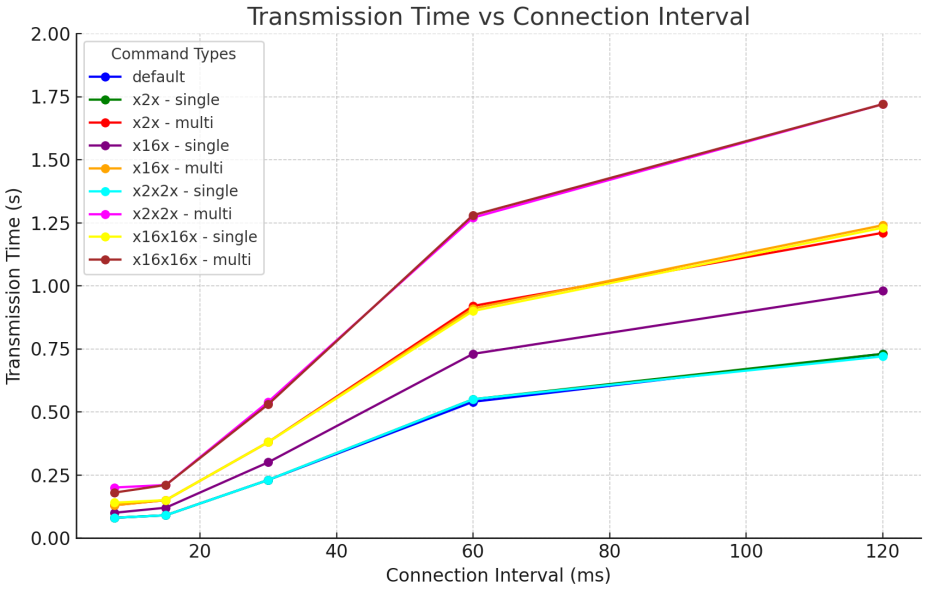


Figure 5.3: BLE Transfer Time for Spaced Texts.

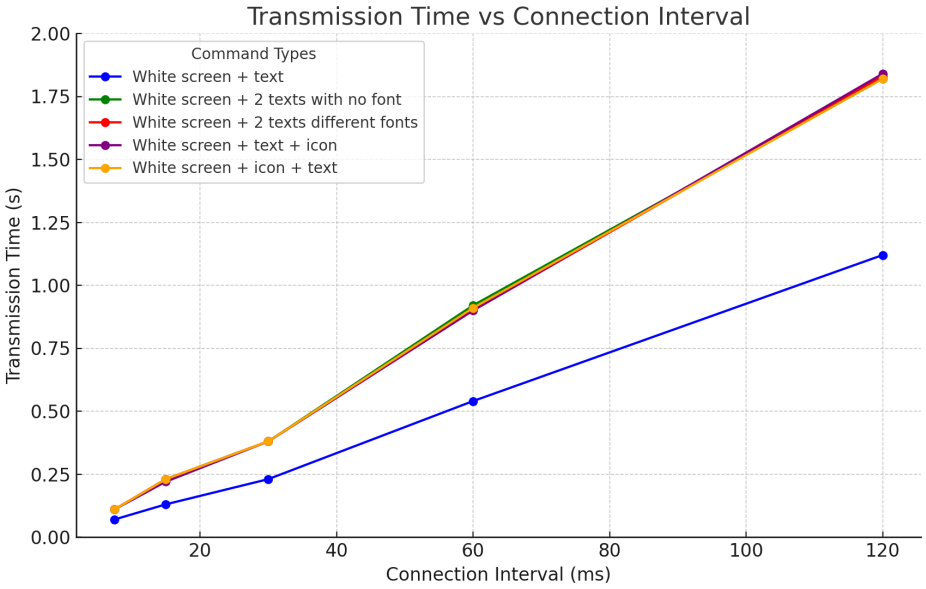


Figure 5.4: BLE Transfer Time for Various White Screen Configurations.

5.1.6 Analysis of Test Results: Throughput

To calculate the data rate, the payload is broken into chunks of 20 bytes, the maximum size per BLE packet. If the total data length isn't a multiple of 20 bytes, an extra packet is needed to send the remaining data. The data rate is calculated based on the total time taken to send the payload and the extra time needed for control commands. For every 20-byte data packet, an additional 17 bytes of overhead are considered for the control command, as mentioned earlier in the document.

The formula used to calculate the throughput is:

$$\text{Throughput} = \frac{(\text{len}(\text{data}) + (\text{num_chunks} \times 17))}{\text{time_taken}}$$

where time_taken is the time to send the data, and num_chunks is the number of data chunks.

The practical data rates obtained from our tests are shown in the table below:

CI (ms)	Data Rate (KB/s)	Data Rate (Kb/s)
7.5	1	8
11.25	0.77	6.16
15	0.61	4.88
30	0.30	2.4
45	0.20	1.6
60	0.15	1.2
75	0.115	0.92
90	0.099	0.79
105	0.086	0.69
120	0.074	0.6
135	Timeout	Timeout
150	Timeout	Timeout

Table 5.1: Practical Data Rates at Different CIs

For the CIs of 135 and 150 ms, the BLE module from ST Electronics does not allow these values, resulting in a timeout during our tests.

All tests were conducted by transmitting a 10 KB data payload using the "Write with Response" mode.

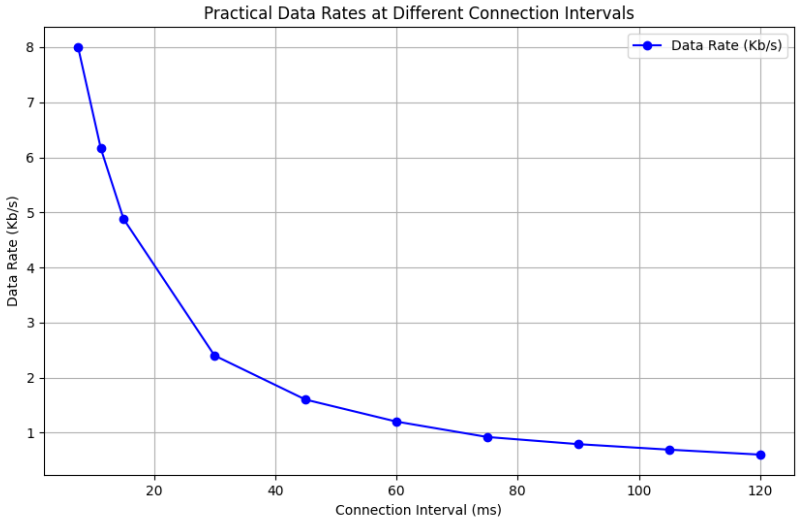


Figure 5.5: Practical Data Rate Variation Based on CI.

From the practical standpoint, as the CI increases, there is a noticeable decline in the data rate. Longer intervals between data transmission events lead to lower data rates. This outcome is consistent with the expected behavior, where lengthier intervals result in reduced data rates.

5.1.7 Analysis of the CI in mobile phone

In the case of the iPhone, practical throughput improvements were not observed when attempting to optimize the CI. This is primarily because the iPhone's BLE implementation defaults to a CI of 30ms, which is already at the minimum threshold that the device permits without invoking the 'update connection parameter' function. As a result, even

with attempts to optimize, the CI remains fixed at this base value, yielding no practical improvement in data transmission rates.

On the other hand, with Android devices, there was a noticeable improvement. Initially, the Android system set a default CI of 48.75ms. However, by using the update connection parameter function, it was possible to reduce the CI to 11.25ms. This change led to a significant increase in performance.

This difference between the two operating systems' BLE behaviors shows the importance of understanding each platform's limitations and capabilities when optimizing BLE performance. The ability to adjust the CI can lead to significant improvements, as demonstrated by the Android example, but this depends on the flexibility provided by the device's operating system and its BLE stack implementation.

5.1.8 Analysis with ST electronic software tool

In advancing our examination of BLE performance, we will be utilizing the Current Consumption Estimation Tool provided by ST Electronics. While the tool's simulations are based on an isolated BLE module, distinct from the comprehensive hardware ecosystem of the aRdent glasses, it serves a pivotal role in our test protocol.

The primary utility of this software in our testing framework is not to extract exact comparative values but to understand the behavioral trends of the BLE chip in response to varying CIs. It's important to note that the simulation ignores real-world complexities like interference from other devices and extra hardware components, focusing only on the BLE module's performance.

Despite these limitations, the software is important for predicting the relationships between the CI and various parameters of interest. By simulating these parameters, we gain insights into the theoretical underpinnings of BLE performance, which aids in drawing parallels with observed tendencies during physical testing. This comparative analysis will allow us to discern the intrinsic patterns of BLE behavior and identify optimal configurations for the CI, contributing to the overall optimization of BLE functionality within the aRdent glasses.

It is important to note that the software only simulates unanswered data communication. It therefore does not take packet acknowledgement into account, resulting in a higher throughput than can be expected in practice.

For the test simulations, a battery capacity of 450mA was assumed, consistent with the actual versions of the aRdent glasses. The increments of the CI were set at 15ms, a value influenced by the operational characteristics of Apple mobile devices. Tables 5.2, 5.3, and 5.4 illustrate the effects of different CI settings on theoretical autonomy, average current during the active phase and total average current, and payload data rate, respectively.

CI (ms)	**Autonomy (time)**
7.5	19 days
11.25	28 days
15	38 days
30	75 days
45	112 days
60	148 days
75	183 days
90	217 days
105	251 days
120	284 days
135	317 days
150	349 days

Table 5.2: Autonomy at Different CIs

CI (ms)	**Current (active phase) (mA)**	**Total Current (µA)**
7.5	5.14	974.55
11.25	5.15	651.53
15	5.16	490.02
30	5.16	247.58
45	5.16	166.67
60	5.16	126.36
75	5.16	102.11
90	5.2	86.03
105	5.2	74.48
120	5.2	65.81
135	5.2	59.07
150	5.2	53.68

Table 5.3: Current Consumption at Different CIs

CI (ms)	**Data Rate (Kbit/s)**
7.5	21.33
11.25	14.22
15	10.67
30	5.33
45	3.56
60	2.67
75	2.13
90	1.78
105	1.52
120	1.33
135	1.19
150	1.07

Table 5.4: Data Rate at Different CIs

The graph displayed in Figure 5.6, created based on the data presented in Table 5.2, illustrates the relationship between the device autonomy and the CI. As the CI increases, there is a clear trend indicating a rise in device autonomy. This pattern suggests that a higher CI, while reducing the frequency of communication events, significantly conserves energy, thus extending the device's operational life before a recharge is needed.

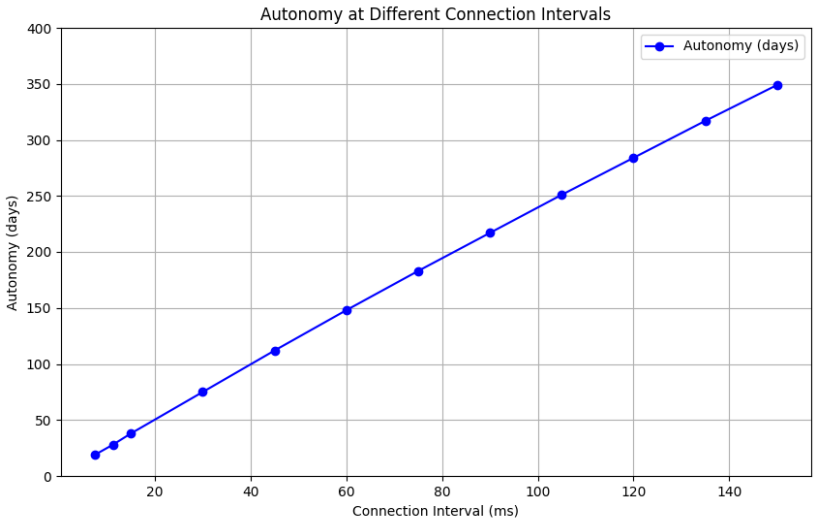


Figure 5.6: Evolution of the autonomy according to the CI.

The gradual rise of the graph from lower CIs to higher CIs shows a nearly steady improvement in battery life.

It's important to consider that while the extension in battery life is advantageous, it may come at the cost of increased latency in data transmission. Hence, a balance must be found between the desired device battery life and acceptable communication delays. The optimal CI setting would therefore depend on the specific requirements of the use case and the importance of power consumption versus data speed.

The graph depicted in Figure 5.7, created based on the data presented in Table 5.3, demonstrates the theoretical changes in total average current as the CI is varied. The trend shown indicates a sharp decline in current consumption as the CI increases from the lowest value up to around 30 ms. Beyond this point, the current stabilizes and the reduction becomes more gradual.

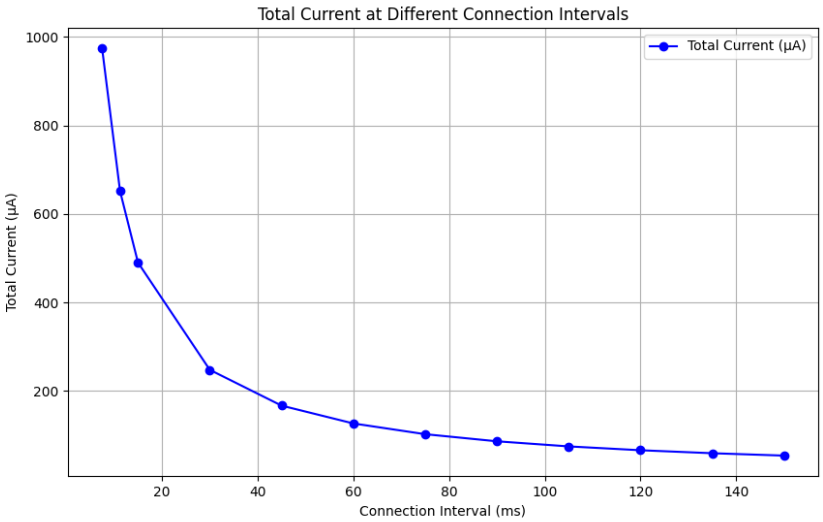


Figure 5.7: Evolution of the total average current according to the CI.

This substantial decrease at lower intervals can be attributed to the higher rate of connection events, which necessitates more frequent radio activity and, consequently, higher energy usage. As the CI extends, the radio is active less often, leading to a decrease in the total average current required by the device.

While the initial sharp decline in current use suggests a rapid gain in power efficiency, the leveling off at higher intervals indicates a point of diminishing returns. Beyond a certain CI, further increases result in negligible improvements in current consumption, implying that there exists an optimal CI range where power efficiency is maximized without compromising the device's operational requirements.

The chart illustrated in Figure 5.8, created based on the data presented in Table 5.4, exhibits a noticeable decline in data rate as the CI is extended. The data rate experiences a precipitous fall-off initially as the CI moves away from its minimum value, before the rate of decline slows and the curve starts to flatten out beyond a CI of approximately 30 ms.

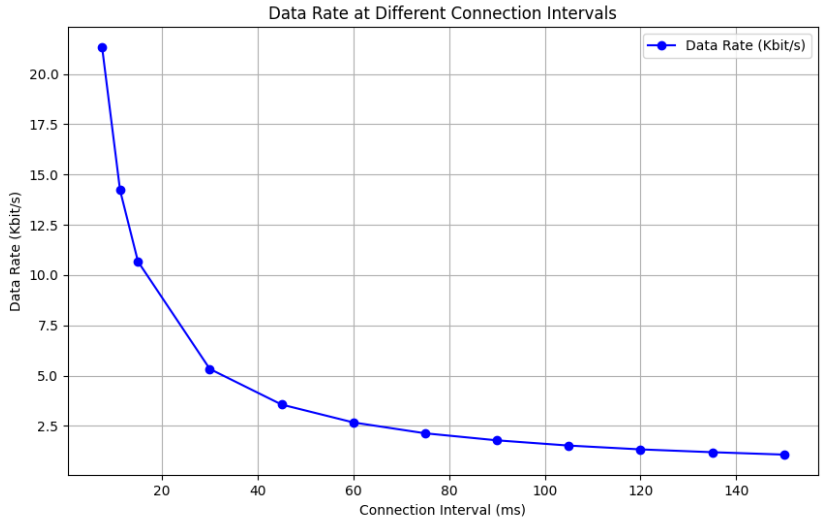


Figure 5.8: Evolution of the data rate according to the CI.

This trend can be interpreted as a reflection of the inverse relationship between CI and data rate. As the interval lengthens, the opportunity for data transmission within a given time frame diminishes, resulting in a lower data rate. This reduction is particularly pronounced at lower CIs, where data packets are sent more frequently, allowing for higher throughput. However, as the CI increases, the reduced frequency of transmissions becomes less impactful, and the data rate settles into a gradual decrease.

For applications where a high data rate is very important, this graph suggests that keeping a shorter CI is beneficial. However, since the CI is a compromise between energy efficiency and latency, the optimal CI setting must be determined based on specific application requirements and power constraints. A balance must be found to ensure that the CI provides the necessary data rate without quickly draining the battery.

This analysis will inform subsequent decisions about the CI configuration in BLE devices, particularly those that are energy-sensitive or require rapid data exchanges, such as real-time monitoring systems.

5.1.9 Theoretical Data Rate Evolution and Practical Data Rate Evolution Based on CI

The graphs show the theoretical and practical data rates at different CI. When comparing the theoretical data rates with the practical data rates, several observations can be made:

- Both graphs show high data rates at lower CIs. This means that when the time between data transmissions is short, the data rate is high.
- As the CI increases, both graphs show a steady decline in data rates. This indicates that longer intervals between data transmissions lead to lower data rates.
- The practical data rates closely follow the trend of the theoretical data rates, showing that the theoretical model accurately predicts real world performance.

The comparison between the theoretical and practical data rates shows that the theoretical predictions are reliable. The trends observed in the practical tests confirm the accuracy of the theoretical model. This validation is important for future optimization of BLE communication parameters in the aRdent smart glasses, ensuring they operate efficiently at different connection intervals.

5.2 Optimizing ATT MTU

In the context of BLE communication for the aRdent glasses, the ATT MTU plays a significant role in data transmission. As previously mentioned in the 4.3 section, ATT MTU is not among the parameters that can be adjusted by the glasses. Thus, the ATT MTU remains fixed at 23 bytes, leaving no room for optimization from the glasses' firmware.

According to the information obtained from [24] [62] resources on maximizing BLE throughput, the data moves through the BLE protocol stack in a specific way. At the GATT layer, application data is structured into attributes and characteristics which are then encapsulated within ATT packets. These packets are subsequently packaged into L2CAP messages with a fixed header size of 4 bytes, before being transmitted over the Link Layer. The BLE packet's data field contains the L2CAP message, which includes the ATT packet with the application data.

The key takeaway is that the ATT MTU defines the maximum size of an ATT packet and can be negotiated between the client and peripheral during connection establishment. For instance, an iPhone 6 or 6S has an ATT MTU of 185 bytes, allowing for a payload of 182 bytes after including the L2CAP and ATT headers, leading to reduced overhead and potentially higher throughput.

In essence, leveraging larger ATT MTUs can diminish the frequency and necessity of sending multiple ATT layer overhead bytes, effectively replacing them with usable data. This not only increase the throughput by enabling the transfer of more application data in fewer packets but also can contribute to lower power consumption due to reduced packet transmission.

In conclusion, adopting larger ATT MTUs can provide a throughput and autonomy increase. The efficiency gain comes from minimizing the overhead of the ATT layer and optimizing the payload capacity per packet.

5.2.1 Modifying ATT MTU

As previously mentioned, the version of BLE used by the aRdent smart glasses is 4.1. This version does not allow for the modification of the ATT MTU. Consequently, we are unable to conduct practical tests to evaluate the impact of modifying the ATT MTU directly on the device.

Despite this limitation, we will simulate the effects of changing the ATT MTU parameter in our subsequent analysis. By doing so, we aim to study its behavior and understand how different MTU sizes could influence the performance of data transfer in Bluetooth communication. This simulation will help us gain insights into the potential benefits and drawbacks of varying the ATT MTU, even though we cannot alter this parameter in the current hardware setup of the aRdent smart glasses.

These simulations will provide valuable information that can be applied in future versions of the device or in other systems where the ATT MTU can be modified, thus contributing to our overall understanding of BLE performance optimization.

5.2.2 Analysis with ST electronic software tool

Despite the fixed ATT MTU size of 23 bytes in the aRdent glasses, the ST Electronics Current Consumption Estimation Tool offers a platform for theoretical examination and simulation of potential enhancements. This software tool is pivotal in assessing the impact of varied ATT MTU sizes on energy consumption and data throughput, despite the lack of this feature in the glasses' current BLE chip.

By allowing the simulation of different ATT MTU sizes, the tool provides a glimpse into the potential improvements in performance and efficiency that could be achieved with an upgraded BLE chip capable of adjusting the ATT MTU.

The theory suggests that larger ATT MTU values could lead to more efficient data transmissions, as they decrease the number of packets required for sending equivalent amounts of data. This, in turn, suggests a reduction in power consumption and an increase in battery life, thus extending the device's autonomy. While these benefits are currently theoretical for the aRdent glasses, the observations are invaluable for strategic planning concerning future hardware upgrades.

In essence, the Current Consumption Estimation Tool by ST Electronics equips us with analytical observation. It provides a predictive model for evaluating the potential advantages of adjustable ATT MTU sizes and supports informed decision-making for future developments in the aRdent glasses' BLE capabilities.

Variations of the ATT MTU will be simulated and its impact on data throughput and device autonomy analyzed. Initially, the CI is set to 7.5 ms to allow for the highest possible data rate. Fixing the CI at a defined value is strategic as it maximizes throughput by using the best settings of both parameters (CI and ATT MTU) and isolates the impact of ATT MTU on throughput.

For the test simulations, a simulated battery capacity of 800 mA will be used. The results of these simulations provide a detailed view of how different ATT MTU settings affect autonomy, current consumption, and data transmission rates. Table 5.5 details the autonomy for different ATT MTU values, Table 5.6 presents the corresponding average current during the active phase and total average current, and Table 5.7 illustrates the payload data rate achievable at each ATT MTU setting.

ATT MTU Value	**Autonomy (days)**
20	34
30	30
40	26
50	24
60	22
70	20
80	18
90	17
100	16
110	15
120	14
130	13
140	12
150	12
160	11
170	11
180	10
190	10
200	9
210	9
220	8
230	8
240	8

Table 5.5: ATT MTU and Associated Autonomy

ATT MTU Value	**Average Current (Active Phase) (mA)**	**Total Avg Current (µA)**
20	4.73	971.86
30	5.12	1106
40	5.47	1240.49
50	5.79	1374.77
60	6.08	1508.44
70	6.35	1643.13
80	6.6	1778.18
90	6.82	1910.18
100	7.03	2043.95
110	7.23	2179.2
120	7.41	2312.47
130	7.58	2446.36
140	7.74	2580.53
150	7.89	2714.68
160	8.04	2852.04
170	8.17	2985.28
180	8.29	3117.54
190	8.41	3252.36
200	8.52	3385.76
210	8.63	3521.51
220	8.73	3655.43
230	8.83	3791.47
240	8.92	3925.25

Table 5.6: ATT MTU and Current Consumption Metrics

ATT MTU Value	Payload Data Rate (Kbit/s)
20	21.33
30	32
40	42.67
50	53.33
60	64.0
70	74.67
80	85.33
90	96
100	106.67
110	117.33
120	128.0
130	138.67
140	149.33
150	160
160	170.67
170	181.33
180	192
190	202.67
200	213.33
210	224
220	234.67
230	245.33
240	256

Table 5.7: ATT MTU and Associated Data Rate

Autonomy

In theoretical terms, increasing the ATT MTU is often expected to enhance device autonomy within BLE communications. This expectation stems from the efficiency gained in transmitting larger amounts of data per packet, which can reduce the number of packets needed for a given amount of data. Fewer packets mean fewer transmissions, which naturally consume less power due to decreased radio usage and reduced overhead for packet headers.

However, the simulations from the ST Electronics Current Consumption Estimation Tool present a different scenario where autonomy decreases with increasing ATT MTU sizes, as shown in Figure 5.9, created based on the data presented in Table 5.5. This outcome can be attributed to the simulation parameters, which assume continuous data transmission until the battery is theoretically depleted. In real-world applications, BLE devices do not continuously transmit data; instead, they operate intermittently, sending data as required by the application's functionality. Therefore, the simulation might not accurately reflect practical usage patterns where increased ATT MTU could indeed improve autonomy by reducing the energy cost per byte of transmitted data.

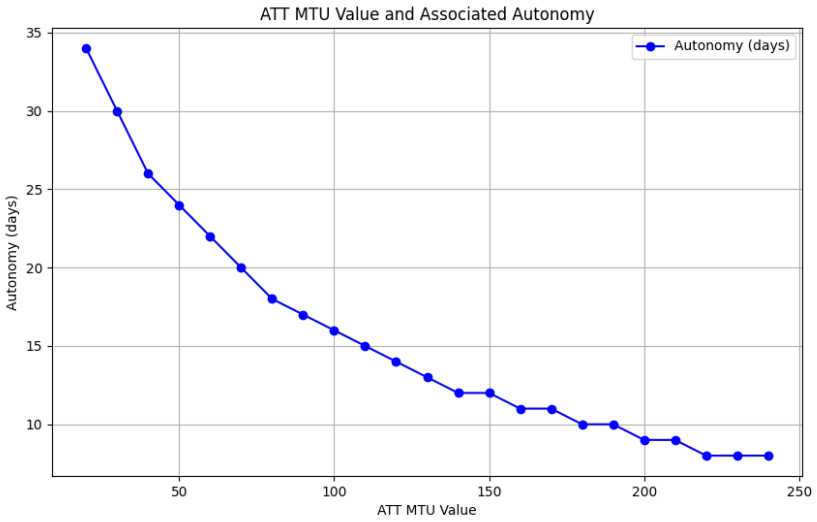


Figure 5.9: Theoritical Autonomy Variation Based on ATT MTU.

These results highlight a critical aspect of using simulation tools they can provide valuable insights into the hardware's capabilities and limitations under specific conditions but might not fully capture the nuances of real-world device operation. As such, while the theoretical benefits of a larger ATT MTU suggest improvements in power efficiency, the practical impact on autonomy will significantly depend on the actual usage pattern of the BLE device.

Current Consumption

After examining the graph that shows the ATT MTU value against the average current in the active phase, as illustrated in Figure 5.10, created based on the data presented in Table 5.6, a clear upward trend is observable. This indicates that as the ATT MTU value increases, so does the average current consumed during the active transmission phase. The trend can be explained by understanding that a larger ATT MTU allows for more data to be sent in a single BLE packet. While this may reduce the total number of packets sent and potentially decrease overhead, each packet with a larger payload consumes more energy to transmit.

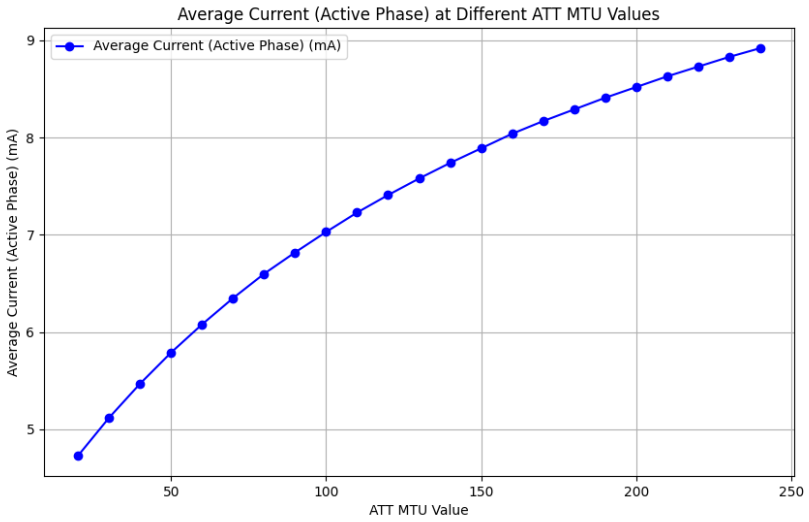


Figure 5.10: Theoretical Average Current In Active Phase Variation Based on ATT MTU.

As the ATT MTU value increases, the total average current consumed by the BLE module also rises, as shown in Figure 5.11, created based on the data presented in Table 5.6. This pattern suggests that with each increment in ATT MTU, which allows for more data to be sent in a single packet, the BLE module must maintain its active state for longer periods to handle the larger data volumes. Consequently, this results in higher overall power consumption, reflected in the increase in average current.

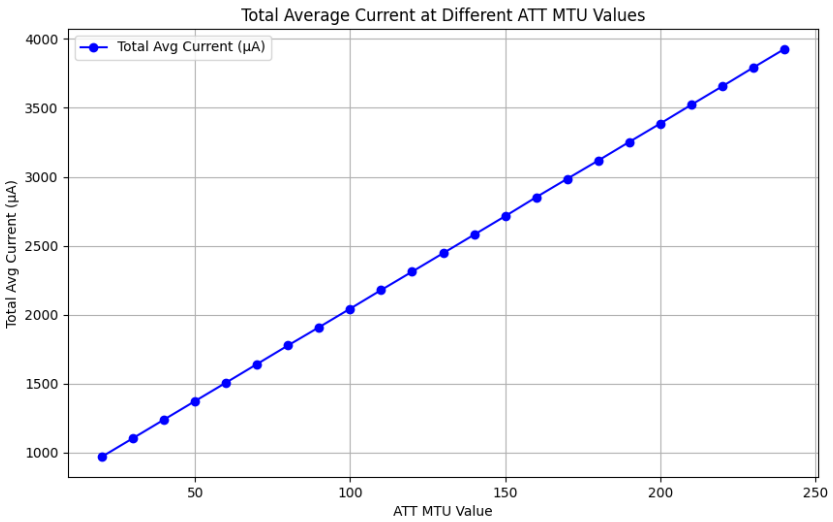


Figure 5.11: Theoretical Total Average Current Variation Based on ATT MTU.

This behavior is consistent with the understanding that larger data packets necessitate more energy for processing and transmission. Each increase in the ATT MTU demands more from the battery as the BLE module engages in more extended periods of high power activity. Therefore, while larger ATT MTUs can improve data throughput efficiency by reducing the number of required packets, they concurrently elevate power demands, potentially diminishing the device's battery life if such transmissions are frequent.

Data Rates

Analyzing the graph depicted in Figure 5.12, created based on the data presented in Table 5.7, we observe a clear linear correlation: as the ATT MTU size increases, so does the data rate. This trend is in line with expectations, as larger ATT MTU sizes enable the transmission of more bytes of data per packet, effectively enhancing the throughput.

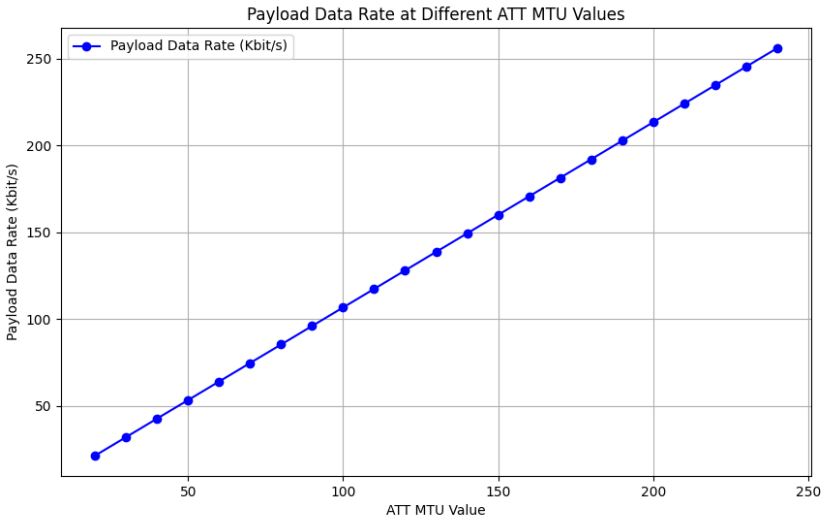


Figure 5.12: Theoretical Data Rates Variation Based on ATT MTU.

With smaller ATT MTU sizes, the data rate is constrained, limiting the efficiency of data transfer. Each packet can carry only a modest amount of application data, with a larger percentage of the communication being occupied by headers and other overhead.

As the ATT MTU size grows, however, the ratio of application data to overhead in each packet improves, leading to more efficient utilization of the available bandwidth. This is because a fixed amount of overhead is spread over a larger payload, reducing the relative impact of that overhead on the total data transferred.

This relationship has significant implications for the design of BLE devices like the aRdent glasses, especially in applications where high data rates are crucial. It suggests that selecting a larger ATT MTU size can be a valid strategy for boosting throughput, provided the device's BLE chip supports such configurations.

5.3 Optimizing Number of Packets per CE

In this section, the relationship between the number of packets per connection event and data throughput is explored. Various sources [24] [18] [61] have observed that increasing the number of packets per connection event leads to an increase in data throughput.

This observation indicates a direct correlation between the efficiency of packet transmission during connection events and overall data throughput. By maximizing the number of packets transmitted per connection event, the data transfer rate can be effectively increased, optimizing the Bluetooth communication system for higher performance.

In the figures presented below, Figure 5.13 shows a connection event with a single packet, and Figure 5.14 illustrates a connection event with multiple packets. These visual representations help to understand why increasing the number of packets per connection event results in higher throughput.

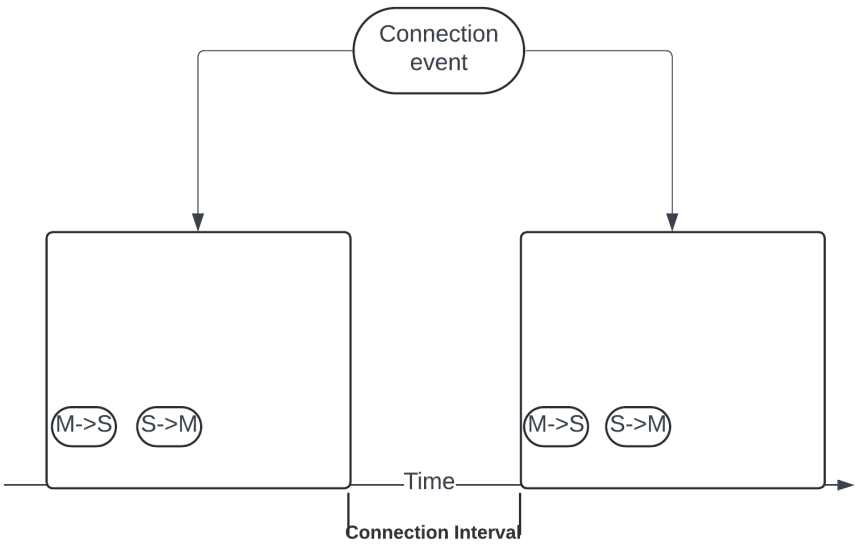


Figure 5.13: Single Packet per CE.

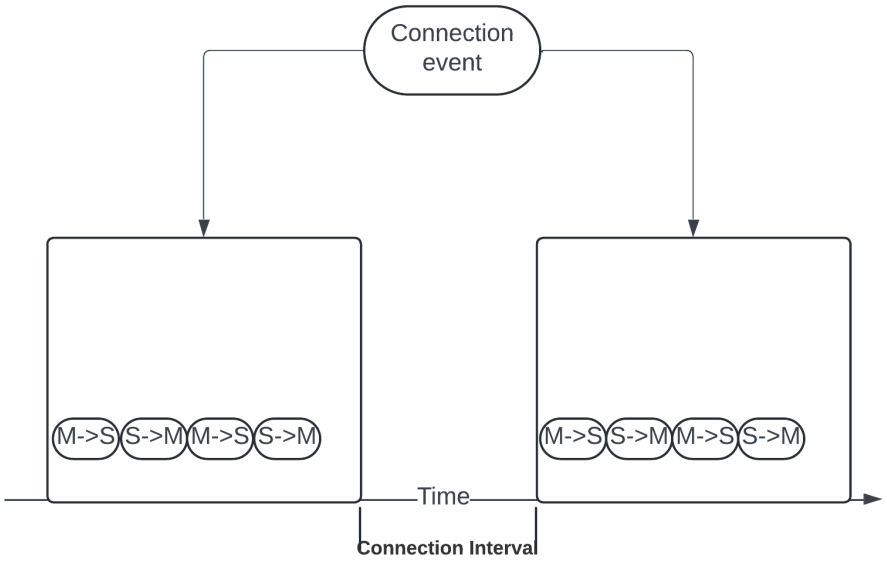


Figure 5.14: Multiple Packets per CE.

Firstly, when a connection event involves only a single packet, there's typically a significant overhead associated with establishing and maintaining the connection, compared to the actual data transmission. This overhead includes processes such as channel initialization, frequency hopping, and acknowledgment mechanisms. As a result, the effective data transmission time within the connection event is relatively low, limiting the overall throughput.

However, when multiple packets are transmitted within a single connection event, the overhead remains constant while the actual data payload increases. This means that a larger portion of the connection event duration is used for transmitting useful data, resulting in a more efficient allocation of the available bandwidth and consequently higher

throughput.

Furthermore, transmitting multiple packets per connection event allows for better use of packet aggregation techniques. Packet aggregation refers to combining multiple smaller packets into a single larger packet, reducing overhead and making better use of the available bandwidth. This optimization becomes particularly advantageous in scenarios where short data packets are common, such as in IoT applications.

Therefore, by maximizing the number of packets per connection event, we not only reduce the relative overhead but also exploit packet aggregation mechanisms more effectively, resulting in higher data throughput and overall system performance.

The objective of this section is to understand the behavior of the parameter "number of packets per connection event" and to explore methods for optimizing it both theoretically and practically for the aRdent smart glasses.

By combining theoretical insights with practical experimentation, the goal is to achieve an optimal balance between data throughput, energy efficiency, and reliability, ensuring seamless and robust Bluetooth communication for the aRdent smart glasses in diverse usage scenarios.

5.3.1 Modifying Number of Packets per Connection Event

The ability to modify the number of packets per Connection Event (CE) in BLE communication depends on the shared capacity between the master and the slave devices. In the case of the aRdent smart glasses, the number of packets is limited by the Bluetooth chip to 8

The master device dictates the number of packets that can be exchanged in each Connection Event based on its own capabilities and the capabilities of the slave device, which in this case are the aRdent smart glasses.

This means that, in some scenarios, fewer than 8 packets may be used if the master device decides on a lower number during the negotiation.

Understanding and modifying the number of packets per Connection Event is crucial for optimizing data transfer efficiency and ensuring reliable BLE communication between devices. This parameter plays a significant role in achieving the desired performance levels in various applications of the aRdent smart glasses.

5.3.2 Analysis with ST electronic software tool

In this section, it was discovered that the two parameters that have been analysed previously, namely CI and ATT MTU, drive what is the maximum packet per connection event.

As a result of this observation, the subsequent parts of this section will further explore how this parameter evolves in relation to the ATT MTU and CI. It will be investigated whether it is always preferable to maximize the number of packets per connection event to achieve the best throughput.

By analyzing how changes in ATT MTU and CI affect the maximum number of packets per connection event, strategies can be devised to adjust these parameters to specific requirements and constraints.

Moreover, it will be examined whether maximizing the number of packets per connection event consistently leads to improved throughput or if there are trade-offs to consider in terms of power consumption, latency, or other performance metrics.

Here are a few intuition that explain the relationships between these parameters:

- **ATT MTU vs Number of Packets:** If the ATT MTU is large, each packet can accommodate more data, potentially reducing the need to send multiple packets.
- **CI vs Number of Packets:** A shorter CI increases data exchange opportunities but may limit the number of packets per connection event due to the shorter interval duration.
- **Energy Consumption Optimization:** To optimize battery life, increasing the CI to reduce device wake-ups may be desired, but this reduces throughput and increases latency.
- **Throughput Optimization:** To increase throughput, reducing the CI and increasing the number of packets per connection event may be favorable, but this will raise energy consumption.

In practice, there's often a trade-off between throughput, latency, and energy consumption. Hardware capabilities and limitations, as well as device-specific BLE stack constraints (e.g., some devices may not support large ATT MTU values or high data exchange frequencies), must also be considered.

In preparation for the integration of a new BLE chip in the system, the BlueNRG-2, simulations were conducted using the ST Electronics Software Tool at the request of GYW to understand the potential performance enhancements this new chip can bring. The study aimed to analyze the data rate and autonomy changes when varying the number of maximum packets per connection event while keeping the ATT MTU fixed. The ATT MTU was set at 23, corresponding to a payload data of 20 bytes as specified by the BLE standard. Table 5.8 showcases the outcomes of this simulation, assuming an 800mA battery capacity for the aRdent 2 glasses.

CI (ms)	**Max Packets per CE**	**Data Rate (Kb/s)**	**Autonomy (800mA)**
7.5	9	192	5 days
11.25	15	213.33	4 days
15	20	213.33	4 days
30	42	224	4 days
45	65	231.11	4 days
60	87	232	4 days
75	109	232.53	4 days
90	131	232.89	4 days
105	153	233.14	4 days
120	176	234.67	4 days
135	198	234.67	4 days
150	220	234.67	4 days

Table 5.8: Data Rate and Autonomy Variation with Fixed ATT MTU and Varied Packet Numbers per CE

These results highlight the direct impact of the number of packets per connection event on the data throughput and battery life.

The initial observation from the Figure 5.8 is that as the CI increases, so does the data rate. This appears contradictory since previous tests yielded the best data rates with the

lowest CI settings. This could indicate an interaction between the maximum number of packets per connection event and the CI that warrants further investigation.

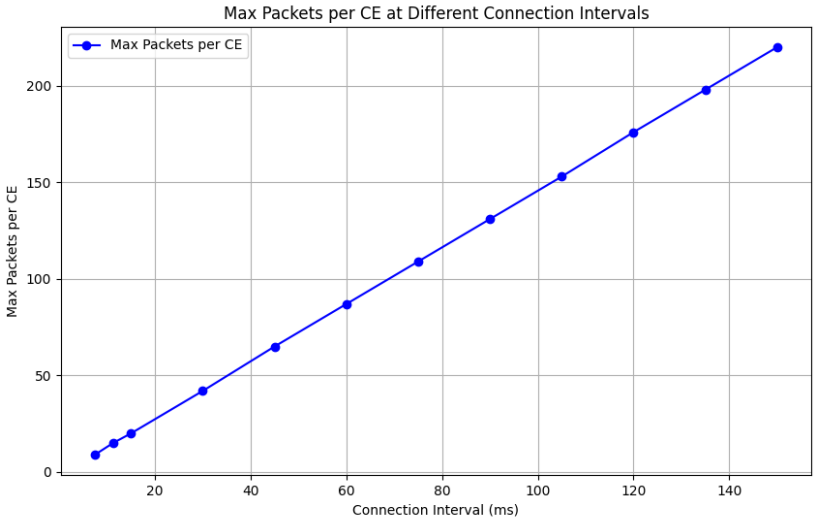


Figure 5.15: Theoritical Maximum Number of Packet per CE Based on CI for ATT MTU = 23.

In Figure 5.15, a linear relationship between the CI and the maximum number of packets that can be sent per CE is observed. As the CI increases, there is a proportional increase in the number of packets that can be accommodated in each CE.

This trend can be explained by the fact that a larger CI allows more time between connection events, which in turn can be utilized to transmit a greater number of packets. This suggests that if the system is designed to optimize data transmission within a given power budget, increasing the CI can be an effective strategy to maximize the data throughput per CE.

It's important to note, however, that while increasing the CI may allow for the transmission of more packets per CE, it can also result in a lower overall data rate and potentially increased latency. The optimal setting would, therefore, be a balance between the desired data rate, acceptable latency, and power consumption constraints.

The graph illustrates that adjustments to the CI have a direct and predictable impact on the number of packets per CE, which is a crucial parameter in optimizing BLE communication for applications like the aRdent glasses, where efficiency and power management are essential.

The Figure 5.16, created based on the data presented in Table 5.8, illustrates the relationship between the data rate and the number of packets per connection event. It reveals that the data rate rapidly increases as the number of packets rises, up to a certain point. Beyond this point, the increase in data rate begins to slow down despite further increases in packet numbers. This suggests that there is a level of saturation where adding more packets per connection event yields diminishing returns in terms of data rate improvements. This plateau could be due to a variety of factors, including limitations of the BLE chip's processing power, maximum data rates, or even the overhead involved in managing a larger number of packets.

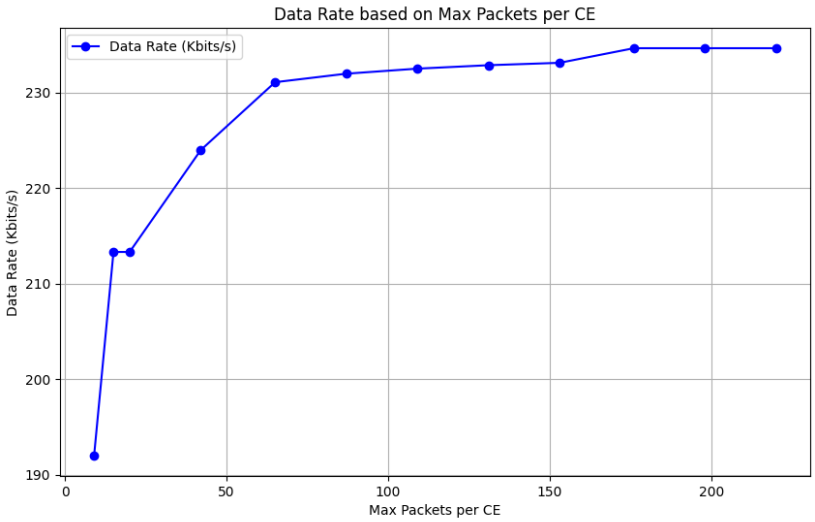


Figure 5.16: Theoritical data rates Based on Number of Packet per CE for ATT MTU = 23.

Now, by holding the CI at 15 ms, let's examine the relationship between ATT MTU and the maximum number of packets per connection event. This will demonstrate how variations in ATT MTU affect both the data rate and the autonomy of the system with a fixed CI. The results are summarized in Table 5.9.

Initially, it is observed that the data rate increases as the ATT MTU size increases, which is typical behavior indicating that larger packet sizes can transmit more data per connection event, thus enhancing throughput. However, beyond a certain MTU size, the trend becomes irregular with fluctuations in the data rate. This suggests that while larger MTUs generally facilitate higher data rates, there may be a threshold beyond which increases in MTU size do not consistently result in higher data rates. These irregularities could be attributed to potential inefficiencies or limitations in the system, such as protocol overhead or hardware constraints, which begin to negate the benefits of larger data packets. This pattern highlights the complexity of optimizing data transmission in BLE systems and the need for careful consideration of MTU size in relation to overall system performance.

The Figure 5.17 shows the Maximum Packets per CE against the ATT MTU Value for a BLE system.

ATT MTU	Max Packets per CE	Data Rate (Kbit-s/s)	Autonomy (800mA battery)
20	20	213.33	4 days
30	18	288	4 days
40	16	341.33	4 days
50	15	400	4 days
60	14	448	3 days
70	13	485.33	3 days
80	12	512	3 days
90	11	528	3 days
100	10	533.33	3 days
110	10	586.67	3 days
120	9	576	3 days
130	9	624	3 days
140	8	598.33	3 days
150	8	640	3 days
160	7	597.33	3 days
170	7	634.67	3 days
180	7	672	3 days
190	6	608	3 days
200	6	640	3 days
210	6	672	3 days
220	6	704	3 days
230	5	613.33	3 days
240	5	640	3 days

Table 5.9: ATT MTU Variations with Fixed CI of 15 ms (bluenrg_2)

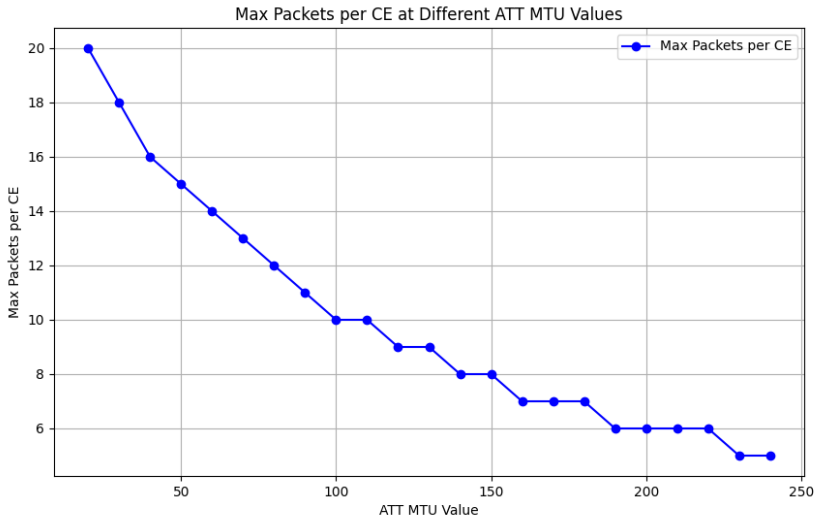


Figure 5.17: Theoritical Maximum Number of Packet per CE Based on ATT MTU for CI = 15ms.

The graph presents a clear downward trend as the ATT MTU value increases, the maximum number of packets per connection event decreases. This inverse relationship

is as expected because with a larger MTU, each packet can carry more data, so fewer packets are needed or can be sent within each connection event given the fixed bandwidth and time constraints.

The trend appears to be quite linear at the start, with a steeper slope from the lowest ATT MTU values, indicating a rapid decrease in the number of packets per connection event as the ATT MTU size initially increases. This steep decline starts to level off gradually, which can be seen as the curve becomes less steep after the ATT MTU value exceeds approximately 100. The line begins to become steady, indicating that the rate of decrease in packets per connection event slows down as the ATT MTU value gets larger.

This behavior could suggest that, at lower MTU values, increments in MTU size have a more significant impact on the number of packets that can be sent in each connection event. As the MTU value continues to rise, each additional increase in MTU size has a proportionally smaller effect on the number of packets per connection event. The graph does not show any anomalies or unexpected spikes or dips, which implies consistent performance across the range of MTU sizes tested.

This information could be valuable in optimizing BLE performance, as it suggests there may be a "sweet spot" where increasing the ATT MTU size provides a beneficial trade-off between the amount of data per packet and the number of packets per connection event. Beyond a certain point, further increases in MTU size may yield diminishing returns in terms of packet count efficiency.

The next step of the investigation consists in reviewing how the data rates are affected when we utilize the maximum number of packets per connection event.

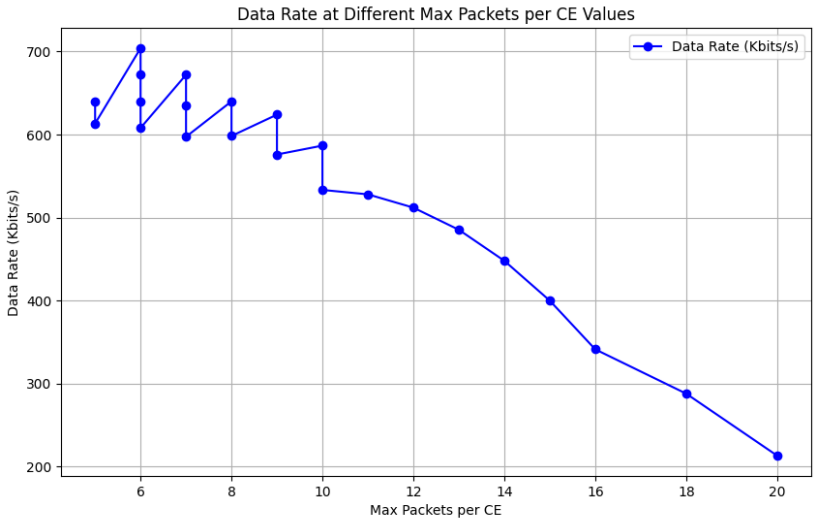


Figure 5.18: Theoritical Data rates Based on the Maximum Number of Packet per CE for CI = 15ms.

Analyzing Figure 5.18, which shows the theoretical data rates versus the maximum number of packets per connection event with a fixed CI of 15 ms, the following observations can be made:

The data rate shows a fluctuating pattern as the maximum number of packets per connection event increases. Initially, the data rate increases, then decreases, then increases again, and continues this oscillating behavior. However, when observing the overall trend, the data rate shows a general decline.

This pattern can be explained by several theoretical factors:

- **Initial Increase**: At the beginning, increasing the number of packets per connection event allows more data to be transmitted within the same time frame, leading to higher data rates.

- **Fluctuations**: The observed fluctuations in the data rate can be attributed to the complexities of the BLE protocol and the interaction between different parameters. For instance, handling more packets might temporarily improve throughput but could also introduce overhead and processing delays that reduce efficiency.

- **Overall Decline**: Despite the short-term increases, the overall decline in data rate suggests that as the number of packets per connection event continues to rise, the overhead and inefficiencies become more significant. This could be due to the increased time needed to manage and process each packet, resulting in a net decrease in data rate.

- **Complex Interactions**: The observed pattern highlights the complex interactions between BLE parameters, where the theoretical benefits of increasing packet numbers are mitigated by practical limitations such as protocol overhead and processing delays.

In conclusion, Figure 5.18 illustrates that while increasing the number of packets per connection event can sometimes boost the data rate, the overall trend shows a decline due to the inefficiencies introduced.

Both graphs show an initial increase in data rate as the number of packets per connection event rises. This is expected as more data can be transmitted within the same time frame when more packets are sent.

For the fixed CI graph (Figure 5.18), the data rate peaks early and then declines with fluctuations, suggesting inefficiencies and overhead associated with managing more packets. In contrast, the fixed ATT MTU graph (Figure 5.16) shows the data rate continues to rise and then stabilizes without significant decline, indicating better efficiency in handling multiple packets.

The fixed CI graph shows a general decline in data rate after an initial peak, highlighting the limitations of increasing the number of packets per connection event beyond a certain point. On the other hand, the fixed ATT MTU graph shows a leveling off after a steady increase, suggesting that the system can handle a larger number of packets more effectively when the ATT MTU is fixed.

The fixed CI scenario suggests that there is an optimal number of packets per connection event, beyond which performance degrades due to overhead and inefficiencies. The fixed ATT MTU scenario suggests that increasing the number of packets per connection event can be beneficial up to a higher limit, after which the performance stabilizes but does not degrade significantly.

Initially, by adjusting the ATT MTU in increments of 10, we discern that the maximum data throughput is achieved with an ATT MTU of 220. This specific configuration, which corresponds to a payload of 220 bytes and a maximum of 6 packets per connection event, yields a data rate of 704 Kbits/s. This represents the highest data rate achievable within the current system constraints without significantly increasing the CI, which would lead to undesirable latency.

However, further refinement and precision in tuning the ATT MTU value reveal that an even higher data throughput is possible. By employing a simulation software that allows for finer adjustments, a superior data rate is attained with an ATT MTU of 247 (which equates to a payload of 244 bytes) and a CI of 30 ms, as is standard on devices like

the iPhone. This more granular approach results in a data rate of 761.6 Kbits/s, which stands out as a more optimal configuration, balancing high throughput with acceptable latency within the parameters of the simulation.

The autonomy behavior in relation to the number of packets per connection event remains the same whether we fix the CI at 15 ms or the ATT MTU at 23.

The CI will be fixed at 15 ms and the ATT MTU at 220 to theoretically study BLE behavior and explore potential optimizations. The results are summarized in Table 5.10.

Number of Packets per CE	**Data Rate (Kbits/s)**	**Autonomy (800mA)**
1	117.33	18 days
2	234.67	9 days
3	352	6 days
4	469.33	4 days
5	586.67	3 days
6	704	3 days

Table 5.10: BLE Performance Metrics with Fixed CI at 15 ms and ATT MTU at 220

The CI is fixed at 15 ms and the ATT MTU at 220 to theoretically study how BLE would behave and explore potential optimizations. The evolution of data rates as a function of the number of packets per connection event is shown in Figure 5.19, based on the data presented in Table 5.10.

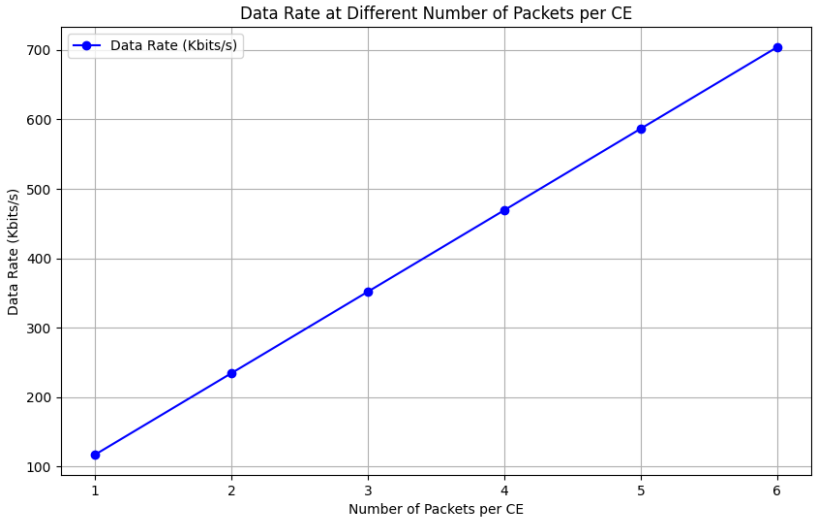


Figure 5.19: Theoritical Data rates Based on the Maximum Number of Packet per CE for CI = 15ms and ATT MTU = 23.

The data rate increases linearly with the number of packets per connection event. This linear relationship indicates that there is a direct proportionality between the number of packets and the data rate, with no apparent signs of diminishing returns within the range shown. For each incremental packet added per connection event, there is a corresponding increase in the data rate, which suggests an efficient utilization of bandwidth.

There is no indication of leveling off or decrease in data rate, which would be expected if there were any inefficiencies or limitations being reached in the system. This implies

that, up to six packets per connection event, the system is likely not hitting any significant protocol or hardware constraints that would prevent the linear increase in data rate.

This trend shows that data throughput can be optimized by increasing the number of packets per connection event within the specified range. It also offers a way to predict data rates for a given number of packets, assuming other factors stay the same and the system continues to behave consistently beyond the shown range.

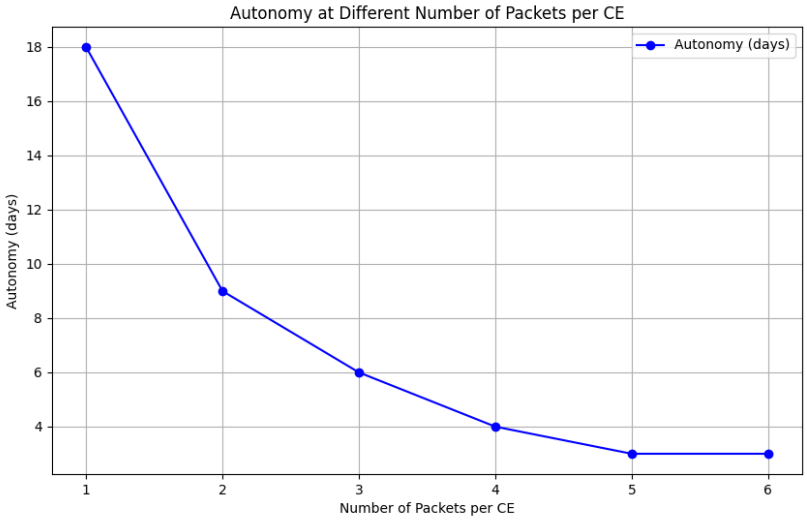


Figure 5.20: Theoritical Autonomy Based on the Maximum Number of Packet per CE for CI = 15ms and ATT MTU = 23.

From the Figure 5.20, there is a steep decline in autonomy as the number of packets per connection event increases from 1 to 2. This suggests that doubling the number of packets significantly impacts the system's energy consumption. However, after this initial drop, the rate of decrease in autonomy slows down between 2 and 4 packets per connection event.

As we move from 4 to 6 packets per connection event, the autonomy seems to stabilize, indicating that the additional packets have a negligible impact on autonomy. This could mean that the energy cost per packet decreases when adding packets to a connection event after a certain point, or it could indicate that the system has reached an efficient state where the energy usage stabilizes despite the increased number of packets.

The initial steep drop followed by a leveling off suggests that there's a threshold of energy efficiency gains to be had by increasing the number of packets per connection event. After this threshold, the benefits in energy efficiency become less pronounced.

5.3.3 Interpretation of the results

The extensive testing conducted provides valuable insights into the optimization of BLE parameters for devices with constrained connection events, such as an iPhone, which limits the number of packets per connection event to four. These findings will be crucial in determining the optimal ATT MTU size or CI to employ for the best throughput under such limitations.

Moreover, it is important to recognize that altering the number of packets per connection event would require the capability to adjust the connection event length parameter on

both devices involved in the communication. However, in practical scenarios, this adjustment is often not feasible as we typically do not have control over the device acting as the BLE master, such as phones or computers, where direct programming of the Bluetooth chip is not possible.

On the other hand, the chip used by Get Your Way is designed to support up to eight packets per connection event, working to maximize this potential for both the current chip and the forthcoming BLE 5.0 chip.

Given these constraints, for the current card with a maximum ATT MTU of 23, the most optimized data rate is achieved with a CI of 7.5 ms, aligning with the results observed in our previous throughput tests.

Looking forward to the capabilities of the new card, if it supports an ATT MTU of 247, then the combination of an ATT MTU of 247 and a CI of 7.5 ms would likely yield a significant improvement in BLE performance.

5.4 Data Length Extension (DLE)

The DLE feature is a key enhancement in the BLE specification introduced with version 4.2. It allows for an increase in the maximum length of data packets transmitted over the link layer, significantly improving data throughput and energy efficiency of transmissions [63].

The DLE enables the extension of the maximum length of BLE data packets. Traditionally, the size of BLE data packets was limited to 27 bytes. With the introduction of DLE, this limit can be increased up to 251 bytes. This extension reduces the number of transmissions needed to send large amounts of data, thereby decreasing protocol overhead and improving overall throughput.

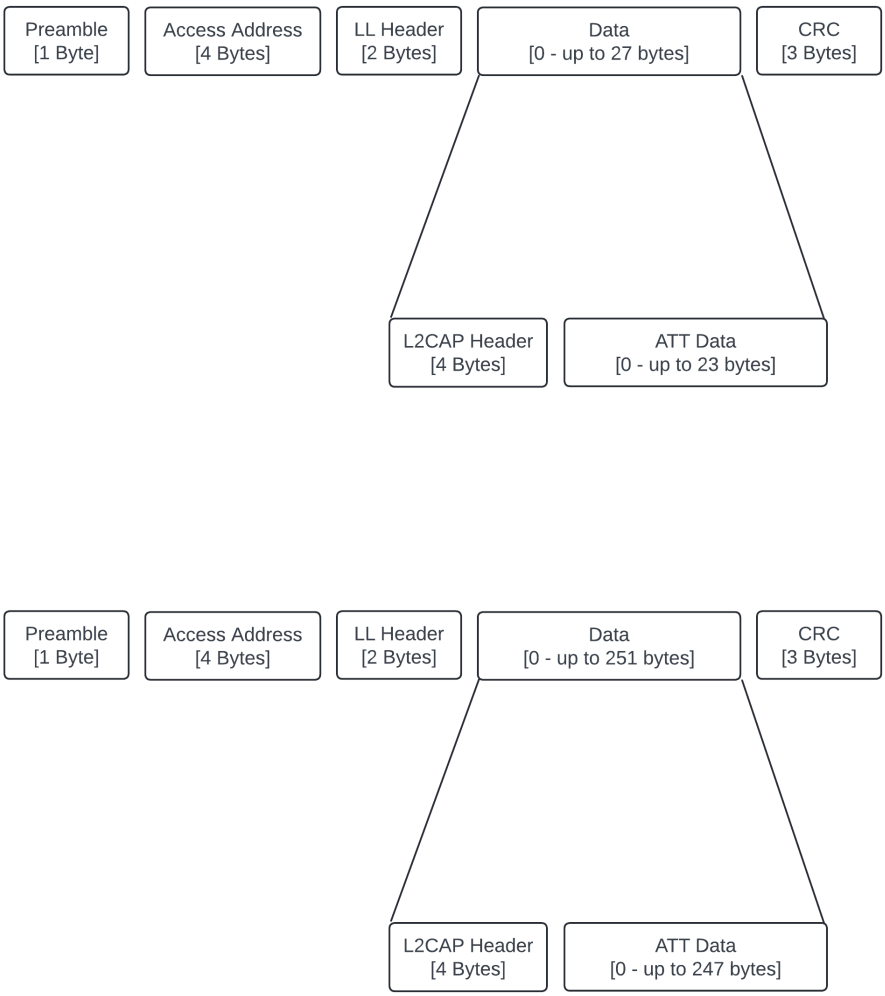


Figure 5.21: Comparison of packet sizes with and without DLE

It is crucial to distinguish between increasing the size of the ATT MTU and enabling DLE, as these two mechanisms enhance data throughput in different ways.

The ATT MTU concerns the maximum payload size of ATT protocol packets (used for data exchanges between BLE devices). By default, this size is 23 bytes, but it can be negotiated between devices to reach higher values. Increasing the ATT MTU allows more user data to be transmitted in a single ATT packet, reducing the number of packets needed to transfer large amounts of data.

On the other hand, DLE affects the maximum length of data packets at the link layer. By enabling DLE, the length of BLE packets at the link layer can increase from 27 bytes to 251 bytes.

In our specific case, we were unable to utilize this parameter on the aRdent 1 glasses, which aligns with the fact that the ATT MTU is locked at 23 on these devices. However, the aRdent 2 glasses take advantage of this DLE feature, allowing for a larger ATT MTU and significantly better data throughput, as will be discussed later in this report.

By combining these two techniques, it is possible to significantly optimize the data transmission performance over a BLE link. Increasing the ATT MTU allows more data to be encapsulated in each ATT packet, and enabling DLE allows these packets to be larger at the link layer, further reducing the total number of necessary transmissions.

The combination of these two improvements enables maximum data throughput, while

optimizing energy usage and reducing transmission latency. Readers should understand that although these two mechanisms are distinct, they complement each other and align to offer optimal performance in BLE communications.

5.5 LE 2M PHY

The LE 2M PHY is an enhancement introduced in the Bluetooth 5.0 specification, allowing devices to communicate using a physical layer that supports a 2 Mbps data rate. This represents a significant improvement over the standard 1 Mbps rate provided by the LE 1M PHY, effectively doubling the data throughput [71] [64].

The LE 2M PHY is a high-speed data rate mode in BLE communication, capable of achieving a maximum data rate of 2 Mbps. This mode is designed to enhance the speed of data transmissions, thereby improving the efficiency and performance of BLE devices.

The primary advantage of LE 2M PHY is its ability to transmit data at twice the speed of the standard LE 1M PHY. This increase in data rate directly translates to higher throughput, allowing for faster data transfers between devices. Additionally, the reduced transmission time for the same amount of data can lead to lower power consumption, as the radio can be turned off more quickly after each transmission. This makes LE 2M PHY not only a performance enhancement but also a potential improvement in energy efficiency for BLE devices.

Unfortunately, for both the aRdent 1 and aRdent 2 glasses, the current chipsets do not support the activation of the LE 2M PHY mode. This limitation restricts the devices to using the standard 1 Mbps rate, preventing them from taking advantage of the potential throughput improvements offered by LE 2M PHY. However, this mode remains an interesting consideration for future versions of the aRdent glasses, where updated chipsets could incorporate this feature to enhance performance.

Adding LE 2M PHY in future versions of the aRdent glasses could significantly boost data transfer rates, leading to smoother and faster communication. This upgrade would be particularly beneficial for applications requiring high data throughput, such as real-time video streaming or large data transfers, increasing the overall user experience.

Chapter 6

Optimizing Write Without Response

6.1 Analyzing Data with PacketLogger

In practice, the initial attempts to optimize the Write Without Response mode did not proceed as expected. When attempting to send a payload of 1000 bytes consisting of the character string "AAA...", only the first few bytes were visible. Specifically, instead of seeing the character "A" repeated 1000 times on the screen, only about ten characters were displayed. This indicates that the majority of the packets were missing in the transmission. Therefore, simply enabling the Write Without Response mode on both devices was not sufficient to ensure proper functionality, highlighting the need for a more in-depth study.

By using PacketLogger, the goal was to capture the messages sent during the test transmissions to understand why only a fraction of the intended data was being received. The analysis of these packets would help identify the exact cause of the failure in data transmission under the Write Without Response mode.

The comparison between the two test scenarios reveals significant details about the data transmission issues. In the first scenario, where the device operates in "Write With Response" mode, all data packets appear to be sent and acknowledged as expected. This suggests that the system is initially capable of handling the intended transmission method.

However, in the second scenario, although the first 12 data packets are transmitted successfully without acknowledgment, the transmission stops abruptly after these packets. No further data are visible, and the device disconnects. This sudden stop and the absence of more data packets indicate a potential issue in maintaining long transmissions without acknowledgments, highlighting a critical area for further investigation and adjustment in the communication protocol or device configuration.

6.2 Investigating the Root Cause and Exploring Potential Solutions

To understand why only a limited number of packets are being transmitted and to find a solution to this issue, a series of tests will be performed to investigate what causes this unusual behavior.

The first test involves altering the Python script running on the MacBook. Specifically, the call to the 'disconnect' function within the script will be removed to observe any changes in behavior. This modification aims to assess whether the premature termination of the connection is a contributing factor to the incomplete data transmission observed.

After modifying the Python script to omit the 'await disconnect' call, a significant

change was observed in the packet reception. This alteration allowed for more packets to be received, showing more than just the initial 12 packets of the "AAA..." character string on the display.

Furthermore, if the command to update the parameter forcing the CI to 7.5 ms is not issued, the quantity of "AAA..." displayed on the screen is also greater. This suggests that the default communication interval may be allowing for a more efficient transfer of data.

These observations resulted to the hypothesis that in the Write Without Response mode, where there is no acknowledgment requirement, if an issue arises, the master device continues to send data to the slave. This could mean that data is being sent too rapidly to the glasses, potentially overfilling the reception buffer and causing subsequent packets to be refused by the glasses. Additionally, the decision not to send the parameter update command likely allowed the device's buffer to accumulate less "unnecessary" data, enabling it to receive more raw data before the reception buffer became full.

This scenario is further illustrated in Figure 6.1, which depicts the representation of a full buffer as per my initial intuition.

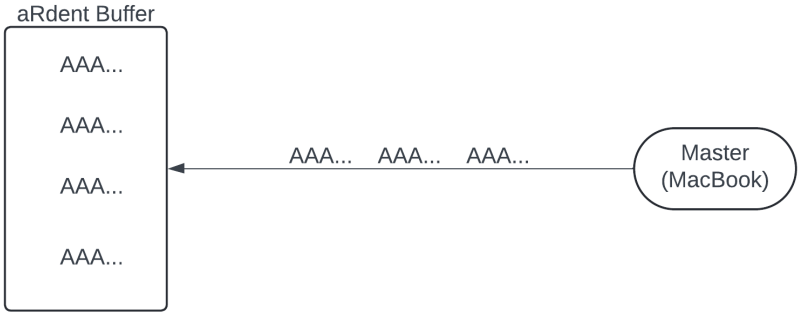


Figure 6.1: Buffer full representation for my first intuition

Based on these findings, the next series of tests will transition from using "AAA..." to numbers. This change will allow for a more precise indication of where packet reception halts, providing clearer insights into the buffer capacity and transmission limits.

6.3 Investigating the Reception Buffer Capacity as a Potential Cause

To determine if the reception buffer is indeed the underlying cause of the data transmission issues, modifications were made directly in the firmware code of the glasses. Initially, the reception buffer was set with a capacity of 5. Under these conditions, when sending 1000 bytes of numerical data, the transmission process halted after successfully receiving and displaying the number 63 on the glasses.

In response, the buffer size was increased to 200 to observe any changes in behavior. With this adjustment, the numbers displayed and received extended up to 96, marking a clear improvement in the capacity to handle more data. This experiment confirms that the size of the reception buffer in the firmware significantly impacts the problem being faced.

However, increasing the buffer size alone does not completely resolve the issue, as not all expected numbers are being displayed. It is important to note that the maximum capacity for this buffer is 255. Pushing the buffer size close to this maximum has occasionally caused the device to freeze and cease functioning. Consequently, the buffer size is maintained around 200 to balance improved data reception and system stability.

6.4 Continuous Monitoring and Dynamic Control of Data Transmission

Following the buffer size tests, an idea emerged. Given that the glasses appeared unable to handle rapid data transmission, a viable strategy could involve continuously monitoring the buffer's fill level. When the buffer approaches full capacity, a signal could be sent to the master device (in this case, the MacBook) to slow down or halt data transmission until the slave device (the aRdents glasses) requests a resumption at full speed.

To implement this, the plan was to monitor the buffer fill status in real-time and display the fill percentage using the JLinkRTTViewer application. However, contrary to expectations, when a large amount of data was sent and the data transfer was prematurely cut off as previously described the buffer was never more than 10% full.

This observation was perplexing as it contradicted the expected behavior of an overflowing buffer that fails to process incoming data. Further investigations using the PacketLogger software suggested that the issue might not be due to buffer overflow [23] but rather due to a communication being truncated too early. This unexpected finding has led us to explore the principles of disconnection, which will be detailed in future section.

Additionally, the role of issuing a 'disconnect' command in this scenario is questioned. Specifically, how could sending a 'disconnect' after the data transmission is complete potentially interrupt the reception of the data? Furthermore, when the 'disconnect' command is removed from the script, the glasses still disconnect, although slightly later, yet still before all data has been received. This behavior raises further questions about the underlying mechanisms controlling device disconnection and data reception integrity under these conditions.

6.5 Modulating Transmission Speed to Enhance Data Handling

Given that rapid data transmission in Write Without Response mode appeared to cause issues with the aRdents glasses, regardless of the specific cause of this problem, it was decided to experiment with slowing the data transfer rate a bit.

To moderate the pace of data transmission, the Python script that generates and sends 20-byte packets was modified by introducing a delay between each send operation. This was achieved by incorporating an `await asyncio.sleep()` statement with a delay of 0.1 seconds between each packet transmission. This method simulates a slower data transfer from the computer to the device.

The outcome of this adjustment was significantly positive, enabling the complete transmission of all data in Write Without Response mode without any issues. This suggests several potential explanations for why adding a delay facilitated successful data reception and transmission:

The firmware on the glasses may have limitations in terms of how quickly it can process incoming data. Additionally, the management of system resources such as memory and CPU plays a crucial role. If data arrives too rapidly, these resources might be overwhelmed, leading to data loss and diminished effectiveness of data handling.

In Write Without Response mode, there is no built-in mechanism to control the flow of data between the master and the slave. Consequently, it falls to the application to manage this flow to prevent overwhelming the slave device. Furthermore, BLE is optimized for low-energy data transmission and may not be ideal for high-throughput data streams without proper flow management.

6.6 Throughput Testing for Data Transmission

To assess the impact of transmission speed on data handling, throughput tests were conducted on a 2019 MacBook Pro using a Python script. The tests varied the inter-send delay in a Write Without Response mode with and without the 'await disconnect' command. The CI was set at 15ms and the ATT MTU at 23 bytes.

Test Configuration

- CI: 15ms
- ATT MTU: 23 bytes
- Data sent: Numerical data ranging from 1 to 1000 for throughput measurement.

Delay (s)	**Throughput (KB/s)**	**Observations**
0.05	0.7227	Full transmission
0.04	0.8992	Full transmission
0.03	1.1891	Full transmission
0.02	1.7378	Full transmission
0.01	3.2911	Full transmission
0.009	3.6519	Full transmission
0.008	4.0365	Full transmission
0.007	4.5785	Full transmission
0.006	5.2685	Stops at 897
0.005	6.1921	Stops at 682
0.004	7.6876	Stops at 732
0.003	9.6861	Stops at 677
0.002	14.5009	Stops at 682
0.001	24.9228	Stops at 461

Table 6.1: Throughput Test Results Without 'await disconnect'

Delay (s)	Throughput (KB/s)	Observations
0.05	0.7187	Full transmission
0.04	0.8938	Full transmission
0.03	1.1812	Full transmission
0.02	1.7333	Full transmission
0.01	3.3106	Full transmission
0.009	3.6607	Full transmission
0.008	4.1228	Full transmission
0.007	4.5103	Stops at 997
0.006	5.2309	Stops at 781
0.005	6.1021	Stops at 668
0.004	7.6456	Stops at 714
0.003	9.6387	Stops at 645
0.002	15.6789	Stops at 698
0.001	23.6238	Stops at 454

Table 6.2: Throughput Test Results With ‘await disconnect‘

The data reveal that reducing the delay between packet transmissions consistently increases throughput. However, at very short delays (below 0.007 seconds), we observe an abrupt stop in data transmission, indicating potential buffer overflow or processing limitations in the aRdents glasses. Interestingly, the inclusion of the ‘await disconnect‘ command does not significantly alter throughput but slightly improves the reliability of data transmission at higher speeds.

Although the transmission stops after transmitting a certain amount of data, which is not an expected behavior, the Write Without Response mode is currently functional for smaller data sizes. This could be practical for applications that do not require large continuous data streams.

The throughput values reported here may not represent the absolute performance metrics and should be interpreted with caution. Later sections of this thesis will explain the reasons behind these discrepancies. For now, the main point to focus on should be the relative improvement in throughput as the inter-send delay decreases, rather than the absolute values.

6.7 Optimal Delay Determination for High Volume Data Transmission

In the quest to optimize data transmission in Write Without Response mode, a series of tests were conducted to determine the best delay setting for transmitting a large dataset without interruption. The primary goal was to successfully send numerical data ranging from 1 to 10,000.

Initial tests revealed that any delay shorter than 0.008 seconds was insufficient to transmit the full range from 1 to 10,000 without issues. Delays of 0.008 seconds and longer, however, consistently allowed for complete and uninterrupted transmission.

After further investigation into delays shorter than 0.008 seconds, a recurring issue was identified: data transmissions were consistently halting just short of completion. This interruption occurred regardless of the exact number of data points sent, and interestingly, it persisted even when the ‘disconnect‘ command was omitted from the master script. This

phenomenon suggested an underlying limitation or blockage in the transmission process that was not immediately attributable to any obvious system parameters.

For more detailed analysis, the numbers from 1 to 10,000 were sent again under various shorter delays to identify when the sending process would stop. At a delay of 0.007 seconds, the transmission was unpredictable, sometimes completing successfully, but more typically stopping around an average of 9980. Delays of 0.006 seconds consistently stopped at about 9600. For both 0.005 and 0.004 seconds, the process halted around 9100. A delay of 0.003 seconds resulted in stopping at 8400. At 0.002 seconds, the transmission ceased at 6800. The shortest tested delay of 0.001 seconds saw the process halt at 4100.

Further tests were conducted to send numbers from 1 up to 1,000,000 to examine the consistency of these stopping points under various delays. Surprisingly, with a delay of 0.005 seconds, it was possible to send all numbers up to 20,000 without interruption. This result was perplexing, as it suggested that if the issue were related to buffer capacity or slave device processing speed, the stopping point would remain consistent regardless of the total number of digits sent. This anomaly reinforced the suspicion that the problem might be related to the timing of the 'disconnect' signal, although the issue remained unclear since the problem persisted even without the 'disconnect' command in the script.

6.8 Focus on Disconnect Functionality

The initial step in addressing the suspected issues with the disconnect command was to remove it from the code to observe the changes in behavior. This modification allowed more data to be sent, but the reason for this improvement was not immediately clear.

Closer inspection, aided by tools such as PacketLogger, revealed that when the disconnect command was removed from the master side, the disconnect still occurred, although slightly later than if the command had been left in the code. Interestingly, the subsequent disconnect originated from the slave device, in this case, the aRdents glasses.

The time between the last data received and the disconnect initiated by the glasses was measured. This measurement highlighted a direct correlation with the supervision timeout set in the update function, which adjusts the CI. It was noted that approximately three seconds elapsed between the last reception of data and the disconnect, exactly matching the supervision timeout duration. However, this did not explain why the timeout was triggered while significant amounts of data were still pending transmission.

The next test involved increasing the supervision timeout to five seconds while continuing to omit the disconnect command from the master script. This change allowed for the transmission of significantly more data, successfully sending numbers from 1 to 10,000 with a delay of 0.005 seconds.

These findings confirmed that premature disconnection was indeed a limiting factor in the utilization of the Write Without Response mode, in addition to previously identified limitations related to buffer capacity.

Despite these adjustments, the question remained why the master device would disconnect before sending all the data. After further reflection, we addressed this issue with the following comprehensive explanation.

6.9 Initial Problem and Mode Operation

Using "Write Without Response" mode in BLE to send data from 1 to 10,000 resulted in only the first 100 numbers being received by the remote device.

The primary issue was not a full buffer but the timing of how data send commands and disconnect commands were processed by the BLE device:

- Commands are processed sequentially by the BLE device. If the disconnect command is processed before all data has been transmitted, it can prematurely interrupt the transmission process.

Introducing a delay (sleep) before the disconnect command ensures that all sent data is transmitted before the disconnection is initiated. This delay compensates for the lag in processing between sending data and executing the disconnect command.

The introduced delay acts as a temporal buffer ensuring the disconnect command does not execute until data transmission is complete, preventing premature data transfer interruption.

Strategic use of delays can be adjusted based on BLE device performance characteristics to ensure all data is transmitted before any connection management commands are executed. Implementing checks to confirm all data has been sent before issuing a disconnect command can also help prevent premature interruptions.

By focusing on the timing and synchronization of commands, this approach maximizes the efficiency of data transmission in "Write Without Response" mode, avoiding disruptions caused by premature disconnect commands.

After extensive testing and adjustments, it has been determined that the Write Without Response mode functions optimally on a MacBook Pro with a delay of 0.003 seconds. This configuration allows for an unlimited amount of data to be sent efficiently without encountering the issues previously noted at lower delay settings. This delay ensures that data transmission is fast yet stable enough to prevent buffer overflow and premature disconnection, facilitating a seamless data transfer experience.

This finding is significant as it demonstrates the practical upper limit of performance within the current hardware and software constraints, ensuring reliable data transfer in applications requiring continuous high-volume data streams without response acknowledgment.

6.10 Understanding the Requirement for a 0.003 Second Delay

A pertinent question arises: why is a 0.003-second delay necessary when using the MacBook Pro? To investigate further, additional tests were conducted using an alternative device for data transmission. A Linux-based Asus laptop was chosen to determine whether the delay requirement was a broader issue or specific to the MacBook.

Surprisingly, the "Write Without Response" mode functioned perfectly on the Asus laptop without any need for a delay. This flawless operation led us to hypothesize that the behavior requiring a delay might be specific to the MacBook Pro.

Despite extensive searches, no relevant information could be found in Apple support documentation or elsewhere that directly addressed this discrepancy. This lack of information suggested that the issue might come from an inherent limitation or a unique operational characteristic of the MacBook hardware when handling rapid data transmissions.

While the "Write Without Response" mode is now fully functional, the implementation differs significantly between devices. On a MacBook Pro, a delay of 0.003 seconds between each packet of data is necessary to ensure all data is transmitted. On the other hand,

the Linux system requires no such delay, highlighting a notable difference in how these devices manage rapid data transmissions.

The adjustments made to the buffer size and the introduction of inter-packet delays were crucial in resolving the issues previously encountered with the "Write Without Response" mode. These modifications allowed for the complete transmission of all data, thereby enhancing the reliability and effectiveness of this mode.

These findings not only highlight the variability in BLE performance across different hardware platforms but also emphasize the importance of adjusting system parameters to the specific characteristics of the device in use. By understanding and addressing the unique needs of each device, it is possible to optimize data transmission processes to achieve consistent and reliable performance across various platforms.

6.11 Verification of Data Display on the Device

Having successfully sent all numeric data to the glasses in "Write Without Response" mode, the next step was to ensure that the data was displayed correctly to the user. To this end, multiple identical strings of characters were sent consecutively to be displayed one after the other on the device.

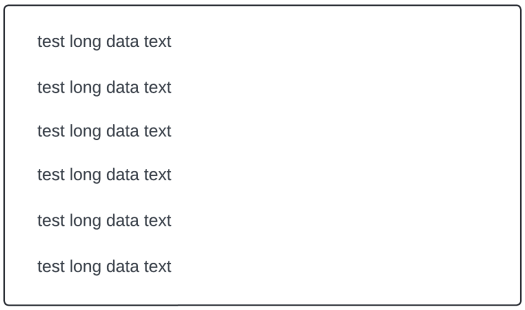


Figure 6.2: Data received in Write With Response mode

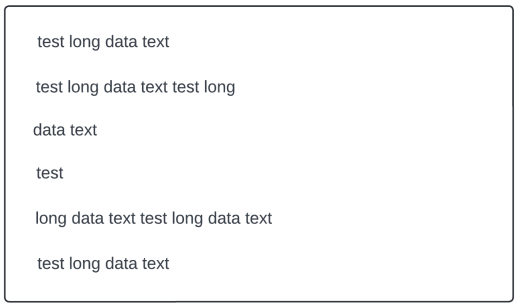


Figure 6.3: Data received in Write Without Response mode

From the comparative analysis of the two images (Figure 6.2 and Figure 6.3), it is evident that the "Write With Response" mode is stable and significantly slower, without any noticeable data integrity issues. However, discrepancies are apparent when examining the display of the glasses that received data in Write Without Response mode, where some data seem to disappear.

Our initial assumption was that the missing data could be due to interference, leading to data loss. This was considered because the practical difference between the two modes is the presence or absence of packet acknowledgment, which ensures that each packet is received before the next one is sent.

Several questions arose from the presence of these issues. Could the use of CRC [37] checks that request the retransmission of erroneous data resolve the issue? Are these truly interference problems? Subsequent tests revealed that it is always the same lines that exhibit issues when they occur. By definition, if interference were the cause, the behavior should be random and affect data randomly. Is there a problem with the data buffer managing high-speed packet arrival incorrectly, potentially leading to overwrite issues?

These questions laid the groundwork for further investigation, aiming to eliminate these possibilities and identify the exact cause of the discrepancies observed.

6.12 Investigating the Source of Data Display Issues: Interference vs. Memory

To determine whether the issues observed were due to interference or memory-related problems, tests were conducted by manipulating the delay between transmissions. The hypothesis was that if interference was causing the artifacts, changes in the delay should not impact the occurrence of these artifacts, since there is no acknowledgment in "Write Without Response" mode that could mitigate interference effects.

Here are the findings from the delay tests:

- Delay of 0.5 seconds: No artifacts observed on the display.
- Delay of 0.05 seconds: No artifacts observed on the display.
- Delay of 0.005 seconds: Artifacts present.

These results, alongside the non random appearance of artifacts, allowed the interference hypothesis to be confidently dismissed. This suggests that the problem likely comes from issues related to buffer handling or memory when dealing with high-speed data transmission.

To determine whether the issue was specifically related to text data buffer handling or a more general problem with high-speed data management, an experiment was conducted by sending an image instead of text.

An entire image was successfully transmitted in "Write Without Response" mode with a delay of 0.003 seconds (the delay that allows uninterrupted sending on a MacBook), and the glasses were commanded to display it. The displayed image was perfect, with no signs of data loss or corruption.

This experiment has thus not only eliminated the possibility of interference but also ruled out the hypothesis of poor data handling in memory when receiving data at high speed. If this were the case, the same problem observed with text data would have occurred with the image.

The final hypothesis considered was a potential race condition [78] in the firmware related to text display. However, further investigation into firmware-level issues was not pursued because GYW's priority was to optimize the BLE performance of the glasses, and due to time constraints, nothing was done to address this problem.

During the image transmission experiment, the transmission times for an image of 38,193 bytes in both modes were also compared:

- **Write With Response mode:** 58.33 seconds
- **Write Without Response mode:** 21.17 seconds

These comparative results further demonstrate the efficiency gains achievable with the "Write Without Response" mode, confirming its effectiveness for high-speed data transfer while underscoring the necessity of optimizing system parameters to prevent data display issues.

6.13 Testing Write Without Response Mode in Flutter

Expanding the scope of the investigation, the performance of the Write Without Response mode was evaluated using Flutter, a popular mobile development framework. This test aimed to understand how the mode performs under different programming environments and settings.

The configuration for the test in Flutter was set with a CI of 30 ms and allowed for sending 4 packets per connection event, instead of the maximum of 8 packets. This setup was chosen to examine the behavior under constrained conditions specific to the iPhone 12 used in the test.

The throughput in various scenarios was measured as follows:

- **Write With Response:** 0.141.22 KB/s
- **Delay of 10 milliseconds:** 0.23 KB/s
- **Delay of 9 milliseconds:** 0.236 KB/s
- **Delay of 8 milliseconds:** 0.238 KB/s
- **Delay of 0 milliseconds:** 0.27 KB/s

Notably, a delay of 0 milliseconds was the most effective, suggesting that the larger CI might have contributed to a lower data rate, allowing for stable transmissions without any delay. Furthermore, the gradual increase in throughput with decreasing delays was not as significant as expected, hinting at potential underlying factors influencing the performance.

The key question remains: why does the Write Without Response mode with zero delay provide a better throughput, but not as high as expected and only slightly better than the "Write With Response" mode? This is especially puzzling when compared to the values observed on a computer using Python. Potential factors could include the language's runtime management, how Flutter handles BLE operations, or the specifics of the device's Bluetooth hardware.

The differences in behavior depending on the CI, delay, and whether the update function is applied or not, suggest that the Write Without Response mode exhibits variable performance characteristics based on several parameters.

6.14 Assessing BLE Behaviors on Alternate Devices

To determine if the observed characteristics of the "Write Without Response" mode were exclusive to the aRdents glasses or inherent to the BLE protocol itself, tests were initiated using a BLE server configured on alternative hardware. This approach aimed to provide a broader understanding of BLE behaviors across different devices.

The initial attempt to establish a BLE server was made on an Asus PC running Linux. The complexity involved in managing BlueZ [21] the official Linux Bluetooth stack which supports all core Bluetooth protocols and layers proved prohibitive. BlueZ configuration and low-level Bluetooth chip programming presented significant challenges, prompting us to seek simpler alternatives.

Next, existing GitHub projects that promised ready-made BLE server functionalities on Linux were explored. While these projects allowed successful device connections, they

fell short in supporting functional data transmissions or manipulations, which were critical for the tests.

Given the shortcomings of the previous attempts, an Arduino [9] was used for its ease of setting up a BLE server. This choice allowed for the quick implementation of a basic server to explore if packet handling on the aRdents glasses specifically influenced BLE behavior.

The Arduino was configured with the following BLE parameters:

- CI: 15ms
- MTU: 517
- ATT MTU: 247

With these parameters, we conducted throughput tests:

- **Write Without Response:** Only 36,864 bytes of a 100KB data transfer were received.
- By reducing packet size to 514 bytes, all data was successfully transmitted, achieving a throughput of 11.3 KB/s.

Further tests in the Write Without Response mode included:

- Without delay, not all data was transmitted, similar to issues observed with the aRdents glasses.
- A delay of 0.018 seconds was required for complete transmission of 100KB data, resulting in a throughput of 26.8 KB/s.
- For 1KB data, transmission was successful even without delay, with a throughput of 721 KB/s.

These results suggested that factors such as the slave device's memory capacity, RAM speed, and the number of packets sent significantly influenced the required delay for successful data transmission. Optimizing packet size was effective in reducing the need for delays, suggesting that memory management and buffer handling played crucial roles.

The investigation confirmed that the Write Without Response mode's performance can vary significantly based on hardware and configuration. The need for delays and the successful transmission of data packets are greatly influenced by hardware capabilities and BLE settings. These insights underscore the importance of customized system configurations to achieve optimal BLE performance across different devices. Further tests will focus on refining these configurations to enhance reliability and efficiency in BLE communications.

6.15 Exploration of MCU Clock Speed Adjustments on Arduino

Suspicions arose that the speed at which packets are processed during reception could prevent the buffer from emptying quickly if it becomes full, leading to potential packet loss in "Write Without Response" mode. The possibility that adjusting the clock frequency of the Arduino's MCU might influence these symptoms was explored. It was hypothesized

that a lower MCU clock frequency might necessitate an increase in the delay imposed between data transmissions to ensure all data is sent successfully.

To test this hypothesis, experiments were conducted where the clock [27] frequency of the Arduino's MCU was varied. These tests were designed to observe whether changes in the processing speed of the MCU would affect the required delay for successful data transmission without packet loss.

Contrary to expectations, the tests concluded that the necessary delay for complete data transmission remained consistent, regardless of the MCU clock frequency. Even with significant variations in clock speed, the delay required to ensure all packets were sent did not change.

These findings eliminate the MCU clock speed as a parameter influencing the behavior of the "Write Without Response" mode. This result suggests that the packet loss issues are not directly related to the speed at which the MCU processes the incoming data, indicating that other factors may be contributing to the observed packet loss. As such, the focus will shift to exploring these other potential influences, which may include factors like Bluetooth stack implementation specifics, data handling efficiencies at the software level, or even hardware-related limitations not directly tied to the MCU's clock speed.

6.16 Factors Influencing Write Without Response Mode Performance

Ongoing efforts to optimize the Write Without Response mode for BLE communications have identified several parameters that may influence its performance. These parameters range from controllable aspects of the communication setup to inherent device-specific limitations.

Several of the critical factors affecting the Write Without Response mode are inherently tied to the hardware characteristics of both the master and the slave devices. These include:

- **Slave Buffer Size:** The capacity of the data buffer on the slave device is crucial.
- **Packet Processing Speed at Reception:** The speed at which the slave device processes received BLE packets directly impacts performance, especially in scenarios where data arrives faster than it can be handled.
- **Packet Sending Speed from Master's Buffer:** This is influenced by the CI, which dictates how often the master device can send packets. A shorter CI allows for faster data transmission.
- **Data Quantity:** The total amount of data being sent affects both the buffer capacities and the timing before a disconnect command might be issued. Larger data transfers are more susceptible to encountering Packet Loss issues.
- **Number of Packets:** Sending smaller payload packets increases the total number of packets required to transmit the same amount of data. This can increase buffer and processing limitations, as each packet incurs overhead for handling.

Understanding these parameters helps in diagnosing issues related to Write Without Response mode and guides the development of strategies to mitigate potential problems.

For instance, optimizing the size of the data packets and adjusting the CI can help manage the data flow more effectively, reducing the likelihood of buffer overflow and packet loss.

Future work will focus on exploring additional controllable factors and refining the approach to BLE communication to ensure more reliable data transmission. The impact of potentially adjusting other parameters, such as enhancing buffer management algorithms or dynamically adjusting packet sizes based on real-time performance metrics, will also be considered.

6.17 Note on BlueNRG-MS Chip Behavior in aRdents Glasses

The BlueNRG-MS chip in the aRdents glasses handles packet transport integrity at the Link Layer, which is crucial for ensuring reliable data transmission. In this layer, the client (or master device) cannot proceed to push packet N+1 unless packet N has been successfully received. This mechanism ensures that from the client to the BlueNRG-MS, data transmission is sequential and no packets are missing, affirming the integrity of the data sent.

The primary concern, however, arises in the interaction between the BlueNRG-MS chip and the host MCU of the glasses. The operational flow within the system is such that once a packet is received, it is immediately stored in RAM. Simultaneously, a signal is sent to the main MCU to prompt the reading and processing of this packet. The MCU then requests the BLE module if it has a received packet in its buffer to process it.

If the MCU is engaged in other intensive activities, it may not process the incoming packet promptly. This delay can result in the packet remaining in RAM without being flushed. Over time, if the MCU does not address these packets swiftly enough, the RAM may become overwhelmed with unprocessed data, leading to potential packet loss. This situation underscores a critical bottleneck in the system where data integrity at the Link Layer does not necessarily equate to data preservation at the application level.

This understanding of the behavior of the BlueNRG-MS chip highlights the importance of not only ensuring data transmission integrity at the Link Layer but also ensuring that the host MCU is adequately optimized to handle incoming data efficiently. It is crucial for the overall system design to facilitate swift data handling by the MCU to prevent data overflow and loss in RAM, ensuring reliable data communication in real-world applications.

6.18 Challenges in Accurately Measuring BLE Throughput in Write Without Response

The throughput measurements mentioned so far do not accurately represent the "on-air" packet transmission rates of BLE. Instead, they incorporate the overhead associated with packet management both on the master's side during transmission and on the slave's side after reception.

Estimating the exact on-air transmission rate is complex due to the intricacies of packet management at the reception. To approximate the on-air rate more closely, one strategy is to reduce the packet management load at reception when calculating throughput.

In "Write With Response" mode, the method for calculating throughput is considered accurate because it measures the real transmission time. This measurement is taken from

the moment the first packet is sent to the moment the last acknowledgment packet is received. This encompasses the entire duration required for all packets to be transmitted and acknowledged.

Conversely, in "Write Without Response" mode, since there are no acknowledgment packets, the timing function that surrounds the write operation only accounts for the time it takes for data to move from the application layer to the transmission buffer or even during lower layers like the link layer. For the application layer, once the data is offloaded to the transmission buffer, it is considered as sent, resulting in a significantly underestimated transmission time.

This miscalculation was evident during practical tests where data sent in "Write Without Response" mode appeared to complete instantly from the perspective of the Python script, while the aRdents glasses continued to receive data for a considerable duration afterward. This is illustrated in Figure 6.4.

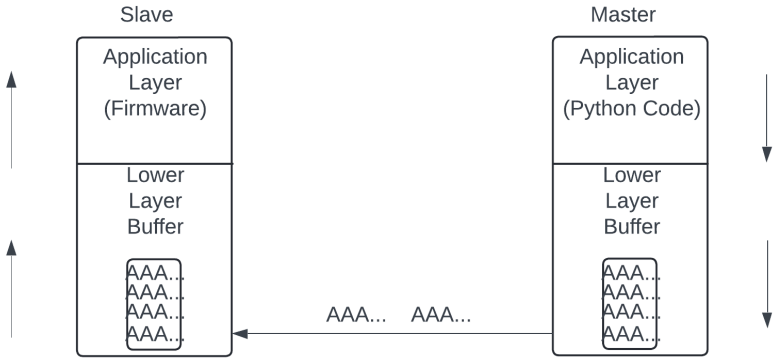


Figure 6.4: Packet travel representation

To address this discrepancy and achieve a more accurate throughput measurement in "Write Without Response" mode, the following procedure is proposed:

- Send a sequence of packets consisting of "AAA..." with the final character being a "B".
- Once this final character "B" is received, the slave device will send a notification back to the master indicating that the last packet has been successfully received.

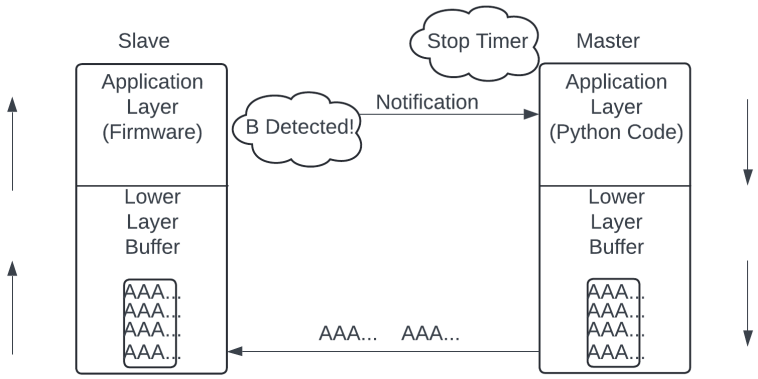


Figure 6.5: Throughput solution representation

This approach ensures that the timing measurement includes not just the data offloading to the buffer but also acknowledges the actual reception of the data on the slave side, thus providing a more complete and accurate measure of the transmission duration and throughput. This is illustrated in Figure 6.5.

6.19 Identifying and Resolving Firmware-Related Instabilities in Throughput Measurement

During the establishment and optimization of the "Write Without Response" mode, it was observed that the throughput calculations were less consistent when compared to the stable measurements recorded on an Arduino, specifically in relation to the aRdents glasses. The consistent performance on the Arduino suggested that the issue of variability might not come from the BLE protocol itself but rather from a firmware-related problem within the system being developed.

After thorough investigation, the source of this instability was identified. Originally, the MCU was programmed to check 1,000 times per second to determine if a packet had been received and was available in the buffer. This approach meant that in the worst-case scenario, there could be a delay of 1 ms due to the time taken for each check. Moreover, since this checking process was implemented twice within the code, it accumulated to a total potential delay of 2 ms in the worst-case scenario.

To address and eliminate this source of delay and instability, the firmware was modified to use hardware interrupts instead of the continuous checking loop. By implementing interrupts, the system could respond immediately to the event of data reception, thereby reducing the response time and eliminating the unnecessary delays previously caused by the frequent checking. This change significantly stabilized the throughput measurements, aligning them more closely with the consistent performance observed on the Arduino platform.

This discovery and subsequent modification underscored the critical impact of firmware behavior on the performance and reliability of BLE communication systems. By shifting from a polling method to an interrupt-driven approach, we not only optimized the system's responsiveness but also enhanced the overall data handling efficiency. This improvement has broader implications for similar BLE applications, suggesting that careful attention to firmware design can lead to substantial enhancements in system performance.

6.20 BLE Throughput Analysis

The BLE throughput performance of the aRdent glasses and an Arduino board is analyzed, considering various test scenarios and conditions.

6.20.1 aRdent Glasses:

The analysis was initiated by examining the BLE throughput performance on a MacBook Pro 2019 running MacOS and an Asus laptop running Linux.

MacBook Pro 2019 (MacOS) Parameters:

- CI: 15ms
- ATT MTU: 23

The throughput results for the MacBook Pro are shown in the tables below. The first table (Table 6.3) presents throughput values in different modes, while the second table (Table 6.4) shows how throughput varies with different transmission delays.

Mode	**Throughput (KB/s)**
With response (Notify)	0.6022
Without response, delay 0.003	5.5

Table 6.3: BLE Throughput on MacBook Pro 2019 (MacOS)

Delay (s)	**Throughput (KB/s)**
0.003	5.5
0.004	5
0.005	4.3
0.006	3.6
0.007	2.8
0.008	2.5
0.009	2
0.010	1.7

Table 6.4: BLE Throughput on MacBook Pro 2019 (MacOS) with Different Delays

Asus Laptop (Linux) Parameters:

- CI: 7.5ms
- ATT MTU: 23

The tables below detail throughput results for the Asus laptop. The first table (Table 6.5) provides throughput measurements in different modes, and the second table (Table 6.6) explores the effects of varying transmission delays on throughput.

Mode	**Throughput (KB/s)**
With response (Notify)	1.20
Without response, delay 0	6

Table 6.5: BLE Throughput on Asus Laptop (Linux)

Delay (s)	Throughput (KB/s)
0	6
0.001	6
0.002	5.4
0.003	5.2
0.004	4.8
0.005	4.5
0.006	4
0.007	3.3
0.008	2.5
0.009	2
0.010	1.5

Table 6.6: BLE Throughput on Asus Laptop (Linux) with Different Delays

The throughput remains relatively consistent across different delays despite the CI being twice as small, which should ideally result in higher throughput. This observation raises a question: how can the maximum throughput achieved on MacOS with a delay of 0.003 (required for the operation of the mode without response on MacOS) be 5.5 KB/s with a CI of 15 ms, while on the Linux PC, only 6 KB/s is achieved with a CI of 7.5 ms, even after removing the delay? This suggests that there might be a limiting factor on the device that is beyond the scope of BLE.

The same tests will now be conducted on the Arduino for comparison.

6.20.2 Arduino:

We further examined the BLE throughput performance on an Arduino board to compare its performance against the aRdent glasses.

MacBook Pro 2019 (MacOS) Parameters:

- CI: 15ms
- ATT MTU: 23

The throughput results for the Arduino on a MacBook Pro are detailed in Table 6.7 for different modes and Table 6.8 for various delays, illustrating how throughput changes with the introduction of response delays.

Mode	Throughput (KB/s)
With response (Notify)	0.64
Without response, delay 0.003	6

Table 6.7: BLE Throughput on Arduino (MacOS)

Delay (s)	Throughput (KB/s)
0.003	6
0.004	5.5
0.005	4.8
0.006	4
0.007	3.4
0.008	2.7
0.009	2.3
0.010	3

Table 6.8: BLE Throughput on Arduino (MacOS) with Different Delays

Asus Laptop (Linux) Parameters:

- CI: 7.5ms
- ATT MTU: 23

Throughput results on an Asus Laptop are shown in Table 6.9 and Table 6.10, demonstrating the performance of the Arduino with and without response delays.

Mode	Throughput (KB/s)
With response (Notify)	0.89
Without response, delay 0	10.1

Table 6.9: BLE Throughput on Arduino (Linux)

Delay (s)	Throughput (KB/s)
0.000	10.1
0.001	8.2
0.002	7.8
0.003	7
0.004	5.9
0.005	4.7
0.006	3.8
0.007	3
0.008	2.3
0.009	2
0.010	1.3

Table 6.10: BLE Throughput on Arduino (Linux) with Different Delays

On the MacBook Pro 2019 (MacOS), the Arduino showed a slightly higher average throughput compared to the aRdent glasses, both with and without response. Interestingly, despite having a smaller CI, the Arduino exhibited higher throughput.

On the Asus laptop (Linux), the Arduino significantly outperformed the aRdent glasses, especially in the mode without response. The impact of a lower CI on throughput was evident, with higher throughput observed at reduced CI values.

The notable difference in throughput between the aRdent glasses and the Arduino, especially on Linux, suggests potential hardware limitations affecting BLE performance, as even with identical BLE parameters, the Arduino demonstrated superior throughput.

The analysis of BLE throughput on different devices revealed varying performance under different conditions. Despite similar parameters, differences were observed between MacOS and Linux environments, suggesting potential hardware limitations affecting throughput. Further investigation is warranted to understand these discrepancies and optimize BLE performance across platforms.

6.21 Data Transmission Strategy on aRdent Glasses

As a reminder, data is transmitted based on "data" and "control" as explained in Section 5.4 (aRdent Data Transfer).

Here, the current data transmission approach of the aRdent glasses will be addressed, focusing on alternately sending data on different characteristics and with different processing for these data types.

It is worth noting that this test and its results will be specific to the aRdent glasses. Tests on Arduino show no difference between writing to one or two characteristics.

First, alternate transmission (i.e., sending on two characteristics alternately, one for data and one for control) and continuous transmission (i.e., sending all data on one characteristic with a single control command at the end) will be compared.

A high delay will be used in this test to ensure stable transmission and simply observe if there is any difference. A 10 KB data transfer will be tested. It is essential to note that although in practice, more data is sent due to control commands after each data command in alternate mode, this additional data is considered in the throughput calculation for a fair comparison.

Here are the results of the two comparisons on a MacBook Pro 2019, with a CI of 15 ms and an ATT MTU of 23, using the mode of writing without response:

- Alternate transmission: 2.5 KB/s
- Continuous transmission: 2.8 KB/s

Next, the percentage difference in throughput will be stated.

Testing if the imposed delay on MacBook varies depending on whether the mode is alternate or continuous: According to the tests, the required delay for all the data to be sent on a MacBook in alternate and continuous mode is 0.003 seconds, as mentioned previously.

However, it is important to note a difference in throughput: For a delay of 0.003 seconds in alternate mode, a throughput of 5.5 KB/s was obtained. For a delay of 0.003 seconds in continuous mode, a throughput of 5.9 KB/s was obtained.

This achieves the same average throughput as in the Arduino tests in continuous mode without response.

It is noteworthy that during data transmission in continuous mode, text processing still occurs after each reception. This processing can be intensive and slow down the data throughput.

Next, continuous data transmission on a characteristic with lighter data processing, which is the characteristic for sending images from the device, will be tested.

Here are the results: The throughput remains the same, around 5.9 KB/s. Thus, it can be concluded that the data processing on the aRdent glasses is not intensive enough to slow down the BLE data throughput.

6.22 Identification of Hardware Limitation on aRdent Glasses

Based on extensive studies, tests, and results, it is believed that the limiting factor of the aRdent glasses originates from a hardware source. With all the gathered clues, it is strongly suspected that it stems from a limitation in the device's buffer.

To demonstrate this phenomenon, consider the tests conducted using the Asus laptop running Linux, which allows for the highest throughput in mode without response and without applying any delay. The results are as follows:

- Delay of 0.001: 6 KB/s
- Delay of 0.000: 6 KB/s

To analyze and confirm this, the following procedure will be performed. Firstly, the Python script will be modified to display when the transmission of data from the application layer is completed. If the buffer is overwhelmed with data, the data will be retransmitted in mode without response, but only via the Link Layer and invisible to the application layer. Thus, any delay between when all data has been sent from the application layer and when it is received on the aRdent glasses can be compared and observed. If this delay is unusually long, it indicates retransmissions. However, in mode without response, there are no acknowledgment packets, so how could packets be retransmitted? If they are, there must be a lower-level retransmission process.

After a closer examination of the BLE packet structure, a layer allowing for data integrity preservation is identified. The Nordic Nrf 52840 [67] tool, which enables packet sniffing in the environment and can be used as a port for software like Wireshark [95], will then be used. This allows for observation of what is happening on all layers of transmission.

Using this tool, retransmissions after a certain number of packets sent have been observed. Subsequently, on Wireshark, all BLE devices will be filtered to confirm these retransmissions. Additionally, the firmware code will be modified to alert when the internal buffer is filled.

Through this procedure, the following observations are made:

- A significantly long latency is observed between the completion of data transmission at the application layer and the completion of data reception at the aRdent glasses.
- Using Wireshark with the Nordic Nrf 52840 as a detection tool, retransmissions are observed.
- At the moment when retransmissions are observed on Wireshark, the buffer is filled on the firmware side.

What happens is that as the reception buffer fills up, packets are not rejected on the aRdent glasses. The Link Layer, responsible for managing data integrity, will resend the unacknowledged data until it's accepted on the aRdent glasses, i.e., until the reception buffer is emptied.

The limiting factor here explaining the difference in throughput between the Arduino and the aRdent glasses for exactly the same BLE parameters is that the reception buffer fills up faster on the aRdent glasses, and the MCU needs time to process all packets for the buffer to empty and accept new data.

6.23 Investigation into Delay Requirement on MacOS

To conclude the examination of the mode without response, the goal is to understand why a delay of 0.003 seconds is needed on MacOS. The data structure is analyzed using the Nordic NRF 52840. When the data is sent with a delay of 0.003 seconds and any packet is analyzed, the MORE DATA section is set to true.

The term "MORE DATA" in BLE indicates whether more data is expected in subsequent packets. In the last packet, the MORE DATA is false. Next, the packets are analyzed if a delay of 0 seconds is set. After a few packets, the MORE DATA is found to be false. However, the reason for this remains undetermined.

To further understand why a delay of 0.003 seconds is needed on MacOS, the framework used was investigated. The Bleak library [16] is utilized for sending BLE data. Bleak works across different operating systems, including MacOS and Linux.

The compatibility of Bleak on both OS was intriguing. After attempting to create a BLE server, complexities in managing drivers and the BLE hardware specific to the computer being used were encountered. For instance, on Linux, Bluez had to be used. On MacOS, it was even more complex as it had to be done using only the Objective C language and the CoreBluetooth [33] framework.

CoreBluetooth is Apple's framework for interacting with Bluetooth peripherals on MacOS and iOS.

After delving deeper into how the Bleak library operates, it was discovered that it utilizes a bridge to translate Python code and adapt it to the respective OS. For instance, to translate Python code and use the Apple framework, the library developer employs PyOBJ [75].

PyOBJ is a Python package that provides tools for interfacing Python with the Objective-C language and Cocoa framework on MacOS.

By using PyOBJ, Bleak enables the utilization of Bluetooth on MacOS via Python.

Thus, an attempt was made to create PyOBJ code without relying on the Bleak library to determine if this delay was due to a programming error or compatibility issue with Bleak and MacOS.

Custom PyOBJ code was created and data was sent to the glasses. The result was exactly the same as when using Bleak, indicating that the delay is inherent to MacOS.

Therefore, the data sending systems were configured to recognize when the device is running MacOS and impose a delay of 0.003 seconds if that is the case.

Chapter 7

aRdent 2 Setup and Optimization

7.1 Introduction to BLE Performance Enhancement at Get Your Way

The strategic decision to adopt the `BlueNRG-2` chip, certified under the latest Bluetooth 5.3 standards, marks a pivotal advancement in product development. This transition not only signifies a move towards more robust hardware but also reflects an ongoing commitment to optimizing communication efficiency in wearable devices.

7.2 Historical Context and Rationale for Advancement

In previous iterations, particularly with the aRdent glasses, notable success was achieved in maximizing data throughput, reaching up to 6 KB/s by meticulously tuning the BLE parameters and optimizing the "write without response" mode. These optimizations were pivotal in achieving reliable performance under constraints typical of real-world conditions.

After receiving the BETA version of the new electronic board featuring the `BlueNRG-2`, efforts have been made to integrate and refine these optimizations further on the aRdent2 board, aiming to exceed previous benchmarks.

7.3 Technical Enhancements and Setup

To harness the full capabilities of the `BlueNRG-2`, we implemented several advanced settings:

- The ATT MTU was increased to 244 to allow larger packets of data to be transmitted, reducing the overhead and increasing the efficiency of our communications.
- The smallest possible CI of 7.5 ms was utilized, enhancing the responsiveness and potential data transfer rate.
- The activation of the Write Without Response mode and DLE further maximized the throughput by minimizing the acknowledgment overhead for transmissions.

This comprehensive approach led to a breakthrough, with the new setup achieving a remarkable data rate of 40 KB/s, a substantial increase from our previous capabilities.

7.4 Challenges and Diagnostic Approaches

Despite the improved throughput, the implementation faced challenges with packet losses observed during transmission. Unlike issues faced with aRdent 1, where packet loss occurred at the end of the data streams, the aRdent 2 experienced losses intermittently throughout the stream. This phenomenon was initially evident from the analysis using JLinkRTTViewer, as depicted in Figure 7.1, showing oscilloscope readings from aRdent 1 that illustrate the packet reception and the data transfer process.

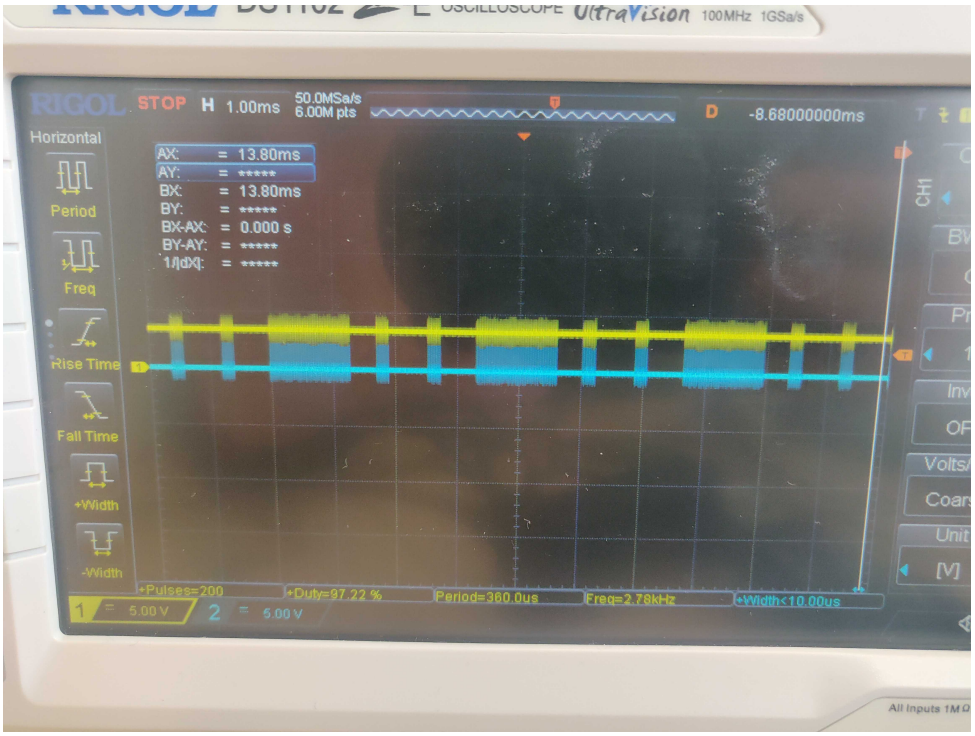


Figure 7.1: Oscilloscope readings from aRdent 1 illustrating the packet reception and the data transfer process.

When applying the changes and optimizations from aRdent 1 to aRdent 2, data losses were noticed. This time, the data losses were not identical to those previously encountered during the BLE optimization of aRdent 1, which occurred at the end of the data stream. Instead, certain packets disappeared in the middle of the data.

First, an explanation of how aRdent 2 uses SPI communication between its MCU and its BLE module to exchange data is needed. When a BLE packet is received at the BLE module, the Interrupt pin is raised. When this interrupt occurs, the MCU asks the BLE module how many bytes it needs to read. The module then responds with the number of bytes, and the MCU reads the data.

In aRdent 1, this process was different as an interrupt mechanism was not used. Instead, a polling mechanism was employed, which checked every millisecond if a packet was available. This mechanism was implemented twice in the code. This process can be seen in the first figure (Figure 7.1), representing an oscilloscope analyzing the SPI communication.

To further investigate, an oscilloscope connected to key SPI interface pins on the `BlueNRG-2` was used, observing both the packet reception interrupts and the signal variations representing the data bits being transmitted. Notably, the aRdent 2 uses an interrupt-driven approach for packet detection, which is a shift from the polling method

used in aRdent 1. The subsequent figure, Figure 7.2, displays oscilloscope readings from aRdent 2, providing a comparative analysis of how packet detection and data transfer have evolved.

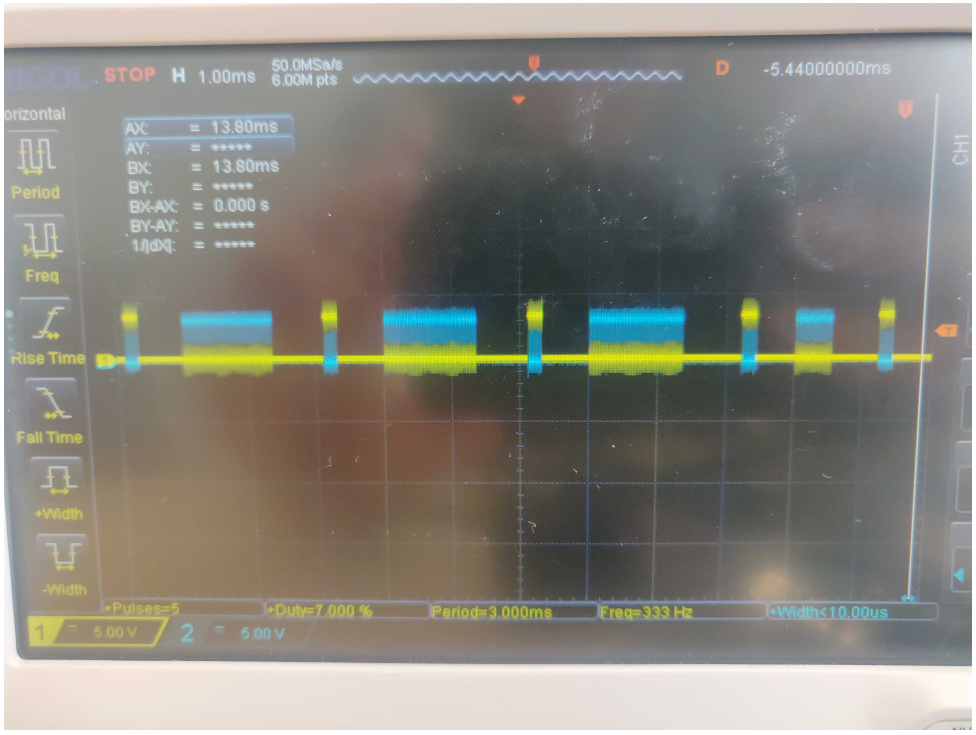


Figure 7.2: Oscilloscope readings from aRdent 2 illustrating the packet reception and the data transfer process.

However, it should be noted that if the queue of the BLE module is too full, it drops the received packet and places a much smaller "packet loss" equivalent in its queue instead. Therefore, if the MCU reads a packet loss, it will know that a BLE packet has been lost. This can be seen in the third image (Figure 7.3), where a packet loss is highlighted in red.

Further analysis revealed regular data packet loss occurring when the MCU failed to read data fast enough from the BLE chip's queue, suggesting an SPI throughput limitation. This was puzzling, given that the SPI bus speed of 1 MBps should be more than adequate to handle the observed data rate of 40 KB/s. Figure 7.3 shows a detail of an anomaly, highlighted in red, that is crucial for understanding the underlying issues in packet transfer.

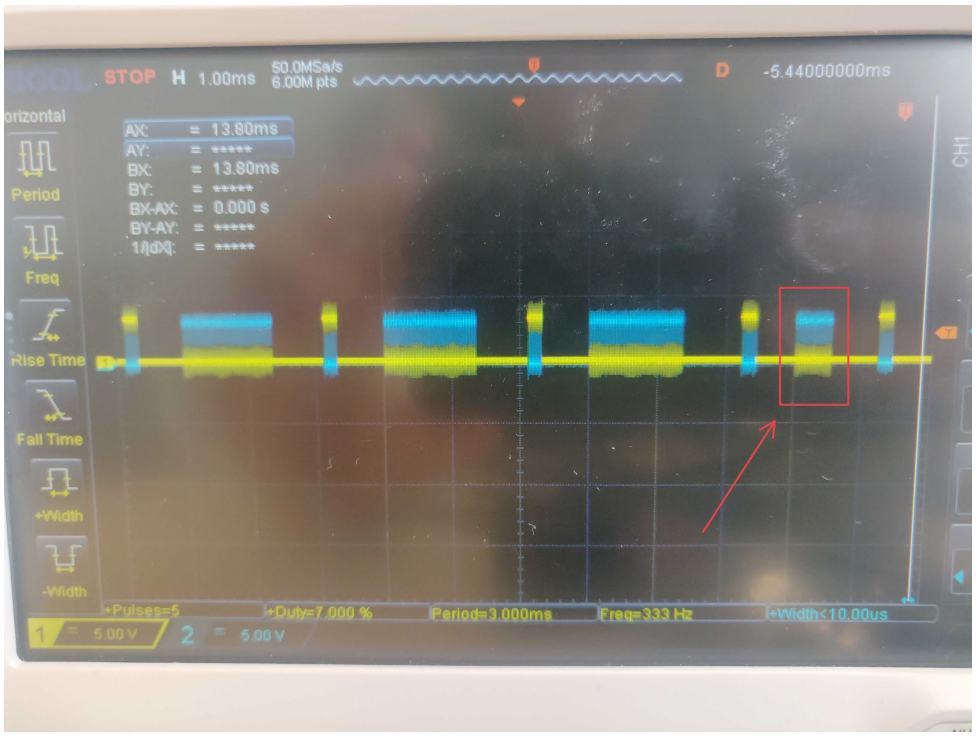


Figure 7.3: Detail of an anomaly indicating packet loss (highlighted in red), crucial for understanding the underlying issues in packet transfer.

The reason for this packet loss is not understood, as the SPI communication on aRdent 2 is faster than on aRdent 1. Additionally, the interrupt mechanism reduces the delay for packet reception detection.

It is unclear why the BLE queue in aRdent 2 is too full, as if the communication to read packets from the BLE module to the MCU were slower than in aRdent 1, but this is not the case. Attempts to resolve and understand the origin of the problem have been made, but no solution has been found so far.

7.5 Concluding Remarks and Future Steps

Efforts to increase the SPI bus speed were constrained by the fixed capabilities of the Renesas driver, highlighting a potential area for future hardware and driver enhancements. The ongoing challenges and our methodical approach to diagnosing and addressing these underscore the complexity of real-world applications of BLE technology in high-performance consumer electronics.

Chapter 8

Interference in BLE Protocol and GYW Auto-Certification

8.1 Introduction to Interference in BLE

Interference in BLE refers to the disruption of the wireless communication signals by external or internal noise sources. These disruptions can degrade the performance of a BLE network, causing reduced data rates, increased error rates, and even complete loss of connectivity. Understanding and mitigating interference is crucial because it directly impacts the reliability and efficiency of BLE devices in complex environments.

8.2 Sources of Interference

The primary sources of interference in BLE are typically categorized into two types: co-channel interference and adjacent channel interference [71].

Co-Channel Interference: Co-channel interference occurs when multiple devices operate on the same frequency channel. In BLE, this is particularly common in places with many wireless devices, such as urban areas, offices, or industrial settings, where different devices might be using the same part of the wireless spectrum.

Adjacent Channel Interference: Adjacent channel interference happens when devices operate on nearby frequency channels. The signal from one channel can bleed into the next, especially if the frequency separation between the channels is not adequate. This type of interference is common in densely populated electronic environments where the spectrum is heavily utilized.

8.3 Engineering and Scientific Tools for Managing Interference

Interference Avoidance Schemes (IAS) [52]: IAS are techniques designed to prevent interference before it can affect communication. This strategy includes dynamic channel selection, which involves constantly scanning the wireless spectrum to find and use channels with the least congestion and interference. By choosing these channels ahead of time, IAS prevents the system from encountering problematic frequencies, avoiding the

degradation of communication quality before it happens. This approach helps maintain clear communication by anticipating and avoiding potential interference sources.

Adaptive Frequency Hopping (AFH): In contrast to the proactive nature of IAS, AFH uses a reactive approach to managing interference in BLE communications. AFH constantly monitors the wireless environment and reacts to changes in real-time. When it detects interference on a currently used channel, AFH quickly switches to another channel that is clearer. This method allows BLE devices to adapt to the changing conditions of the spectrum, ensuring continuous communication without a pre-selected channel plan. The reactive nature of AFH makes it very effective in environments where interference levels can change quickly and unpredictably.

Power Control: Power control is another effective way to manage interference, especially in crowded environments. By adjusting the power level of the transmission, devices can reduce the chance of causing interference to other devices while keeping the signal strong enough for communication. This approach is mostly proactive, as it involves setting transmission power at levels that are less likely to interfere with other nearby devices from the start.

Channel Selection Algorithms (CSA) [26]: CSA 1 [34] and CSA 2 [35] play critical roles in managing the use of communication channels efficiently. CSA 1, the original algorithm, operates with a fixed pseudo-random hopping sequence and does not adapt based on interference data, making it universally compatible with all BLE devices following earlier versions. CSA 2, introduced with Bluetooth 5.0, enhances this by dynamically selecting channels based on real-time interference assessments, thus requiring both communicating devices to support this newer standard for optimal performance. The retrocompatibility of CSA 2 with older devices is limited, as these devices do not support the dynamic channel adjustments offered by the newer algorithm.

8.4 Concluding Thoughts on Interference Management

Effective management of interference in BLE requires a mix of these techniques, suited to the specific environment and application needs. By using both proactive and reactive strategies, engineers and designers can ensure that BLE technology stays reliable and efficient in the growing complexity of global wireless communications. Balancing these approaches helps create strong communication networks that can both anticipate potential disruptions and respond quickly to existing ones.

8.5 Interference Management in aRdent and aRdent 2 Glasses

While both models share many similarities in terms of design and basic functionalities, they differ significantly in their approach to handling interference, primarily due to the differences in the Bluetooth standards and CSA they employ.

The original aRdent glasses, compliant with Bluetooth 4.1, utilize CSA 1 for channel selection. This standard approach, while effective in general usage scenarios, does not dynamically adjust to fluctuating interference patterns. As a result, the aRdent glasses are

somewhat limited in their ability to proactively avoid interference, leading to occasional retransmissions and slightly reduced data transmission efficiency in environments with high electronic activity.

In contrast, the aRdent 2 glasses are equipped with the newer Bluetooth 5.3 certification, which enables the use of CSA 2. This advanced channel selection algorithm offers enhanced interference management by dynamically adapting to the current state of the spectrum. CSA 2 allows aRdent 2 glasses to actively avoid congested channels and minimize the likelihood of interference, resulting in fewer retransmissions and more stable communication performance. The use of CSA 2 marks a significant upgrade in the aRdent 2's ability to maintain robust and reliable connectivity, even in densely populated electronic environments.

8.6 Potential Improvements and Challenges in Interference Management Algorithms

8.6.1 Proposed Enhancements in Interference Management

As wireless communication technologies evolve, particularly within the IoT sector, the robustness of BLE connections becomes increasingly crucial. Recent advancements, as discussed in the academic paper by Bozheng Pang et al., propose significant improvements in the BLE link layer for interference detection and channel selection that could be adapted for future enhancements in devices like the aRdent glasses.

Recent research introduces a novel Interference Avoidance System (IAS) and an Improved Channel Selection Algorithm (CSA) [71] aimed at enhancing connection robustness under various interference conditions. The IAS is designed to detect interference more accurately and swiftly by monitoring metrics such as Signal-to-Noise Ratio (ST) and Packet Loss (PL), which indicate the quality and integrity of the communication channel.

One of the key improvements is the dynamic nature of the proposed CSA, which incorporates real-time environmental data into the channel selection process. This approach allows the device to adapt its channel selection based on the probability of interference, rather than merely reacting to interference after it has impacted the connection. Such a method could significantly reduce the incidence of retransmissions due to interference in the aRdent 2 glasses, which already utilize CSA 2 to great effect.

For the aRdent and aRdent 2 glasses, incorporating an advanced version of IAS in conjunction with CSA 2 could provide a competitive edge by further minimizing interference impacts. This integration could lead to even fewer retransmissions, lower latency, and improved overall communication reliability, particularly in environments with high electronic device density.

8.6.2 Limitations in Current Interference Management Strategies

In the ongoing effort to enhance BLE devices' resistance to interference, significant challenges persist, primarily due to the architectural and access limitations of the BLE hardware and firmware. One of the critical constraints involves the accessibility and modifiability of the Link Layer, where crucial decisions about packet transmission and interference management are made.

The BlueNRG series, which includes BlueNRG_MS and BlueNRG_2 modules, comes equipped with a BLE stack that includes both the host and link layer components. This

stack is compliant with the Bluetooth versions, which means it implements the CSA directly as part of the BLE standard. CSA1 or CSA2 are integrated depending on the product's Bluetooth release version. However, these stacks are not open-source, and modifications at the Link Layer are restricted due to the need to keep the stack compliant with official BLE versions.

The closed nature of the Link Layer means that it is not possible for developers to access or modify the CSA directly. This limitation is significant for those looking to implement custom algorithms or adaptations, such as the improved CSA or Interference Awareness Schemes (IAS) that are proposed in some academic research. Such enhancements could potentially allow devices to better adapt to real-time interference conditions by dynamically selecting channels based on comprehensive environmental data, rather than relying solely on the predefined algorithm.

Given these restrictions, testing and observing the actual impact of interference and the effectiveness of CSAs become challenging. Developers can monitor high-level packet transmissions and HCI (Host Controller Interface) events, but they cannot access detailed Link Layer operations where retransmissions and error handling due to interference occur. This situation complicates efforts to calculate packet error rates or to directly observe the effects of interference on packet loss and retransmission rates.

8.7 Auto-Certification Strategy for aRdent Glasses

8.7.1 Objective of Auto-Certification

The goal of this auto-certification is to ensure that the entire product, not just individual parts, is strong and reliable in places with high potential for interference, such as industrial warehouses. This certification aims to show that the glasses can work well and handle real-world conditions where electronic interference is common.

8.7.2 Planned Testing Procedure

To achieve this certification, GYW plans to conduct a series of thorough EMC [42] tests. These tests are crucial for checking the electromagnetic fields emitted by the glasses and their ability to withstand external electromagnetic interference. The University of Liège has been chosen as the testing location due to its advanced laboratory facilities and its strong reputation for conducting detailed and reliable EMC testing.

The EMC tests will be carried out in a controlled environment within the university's laboratories, designed to simulate various levels of electromagnetic interference. These settings will imitate the conditions typically found in industrial warehouses, where the presence of heavy machinery and electronic equipment can lead to significant electromagnetic disturbances.

The results from these EMC tests are expected to provide valuable insights into the electromagnetic characteristics of the aRdent glasses. Successfully passing these tests will not only affirm the product's compliance with international EMC standards but also enhance its marketability by certifying its performance in demanding environments. Furthermore, this testing will help identify any potential areas of improvement, allowing GYW to optimize the glasses for even better performance and reliability.

8.7.3 Detailed Testing Procedure for Data Integrity and Interference Assessment

The choice to use diverse data types is driven by the need to comprehensively assess the glasses' performance across all potential use cases. By including a broad range of data, the tests can more effectively imitate the multifaceted nature of user interactions, ranging from low-bandwidth text communications to high-bandwidth file transmissions. This approach ensures that the testing is not only robust but also highly relevant to the glasses' intended operational contexts.

Procedure for Data Transmission and Comparison

During the transmission phase, predefined sets of data representing typical user interactions will be sent to the aRdent glasses. This phase will test the glasses' ability to receive data accurately and promptly in the presence of simulated interference.

Following the transmission, the data received and processed by the glasses will be captured and analyzed. This phase is critical as it provides insights into the effectiveness of the glasses' internal processing and error-correction mechanisms under stress conditions.

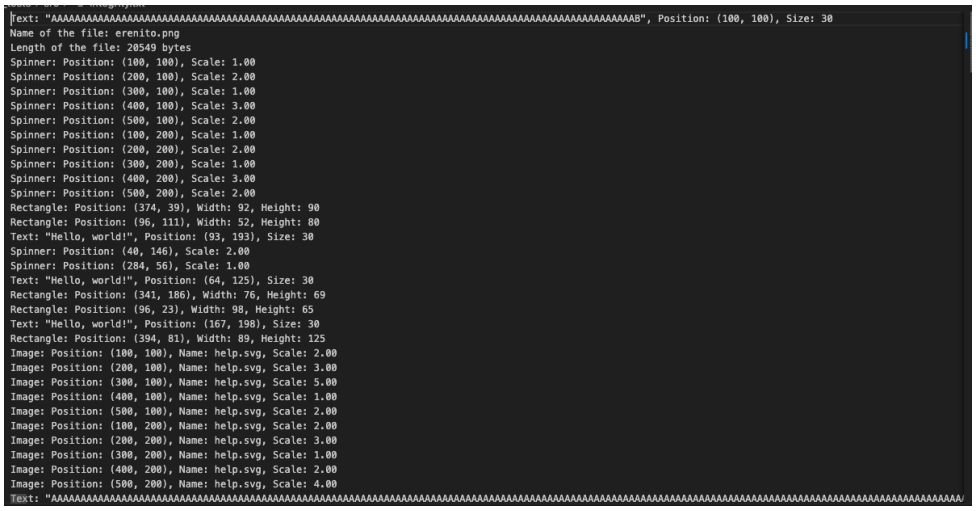
A comparative analysis will be conducted between the data sent to the glasses and the data received and processed by them. This comparison is essential for identifying any discrepancies caused by interference, which may manifest as data corruption, loss, or delay in reception.

The expected outcome of this testing is a detailed report highlighting the aRdent glasses' capabilities to handle different types of data under varying levels of electromagnetic interference. This report will not only validate the product's versions against real-world conditions but also identify potential areas for further enhancement in future iterations of the product.

8.7.4 ECM Testing Procedure During Data Transmission Phase

The data integrity testing phase is designed to rigorously assess the accuracy and completeness of data transmission under electromagnetic interference. This phase involves a series of specialized Python scripts that each transmit a different type of data to the aRdent glasses, including text, files, and various graphical commands such as displaying spinners, rectangles, and icons.

Each test script is responsible for sending a specific type of data and simultaneously recording detailed logs on the computer about the transmitted data. These logs are crucial for later analysis and are stored in text files, which capture the specifics of each data type sent during the tests. Figure 8.1 shows an example of these log files, illustrating the detailed recording of data transmission.


Text: "AAAA

```
Time taken to send data: 2.5048751620051917 seconds
Data throughput: 3.999400909057853 Kilo-bytes/second
Time taken to send data: 2.4888558689999627 seconds
Data throughput: 4.025142686958925 Kilo-bytes/second
Time taken to send data: 2.521012052005972 seconds
Data throughput: 3.973800915401681 Kilo-bytes/second
Time taken to send data: 2.491621958994074 seconds
Data throughput: 4.020674149157243 Kilo-bytes/second
Time taken to send data: 2.4743963590008207 seconds
Data throughput: 4.048664218066237 Kilo-bytes/second
Connection Stability Test: 10 successful attempts out of 10
Latency: 0.023059287006617524 seconds
```

Figure 8.2: Text file recording BLE performance metrics

8.7.5 Representation and Setup of the ECM Testing Procedure

The ECM testing for the aRdent glasses is designed to ensure that both the transmission and reception of data are robust against electromagnetic interference. The following details the test setup and procedure executed in a controlled laboratory environment at the University of Liège.

The test is conducted in a specially equipped laboratory at the University of Liège, designed to simulate high-interference environments typically found in industrial settings. This setup allows for precise control over the electromagnetic conditions, ensuring consistent test results. Figure 8.3 provides a schematic diagram of the ECM testing setup, illustrating how the environment is structured to facilitate these tests.

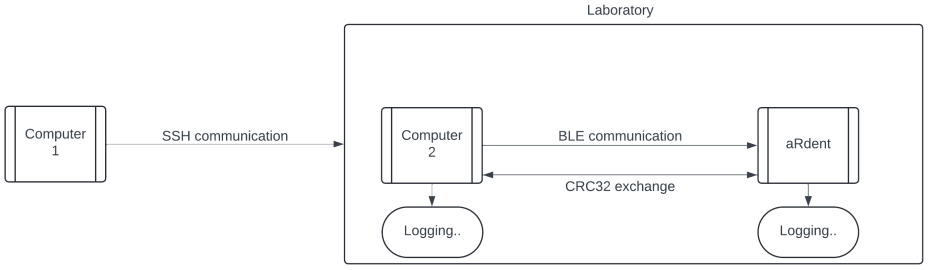


Figure 8.3: Schematic diagram of the CEM testing setup

To facilitate the test, the computer that initiates the data transmission is connected via SSH [79], allowing remote operation and monitoring. This setup enables the test operator to manage the test from a secondary location.

The second computer, which serves as the control unit, initiates the transmission of various types of data including text, files, and graphical commands to the aRdent glasses. During transmission, this computer also logs detailed information about the data sent in a text file.

After receiving the data, the aRdent glasses record similar logs in their internal memory. These logs are crucial for later comparison and analysis of data integrity.

After each data transmission session, the control computer calculates the CRC32 checksum of the data logs. Simultaneously, a command is sent to the aRdent glasses

to calculate the CRC32 checksum of the data they have received. The checksums from both the control computer and the aRdent glasses are compared to verify data integrity.

8.7.6 Impact of Logging on BLE Performance

During the CEM testing of the aRdent glasses, each received data packet is logged into a text file, significantly loading the device's MCU. This logging consumes processing power and memory, which otherwise would be dedicated to managing BLE communications. As a result, the MCU's ability to process incoming BLE packets efficiently is compromised.

The effect of logging on data throughput is substantial. With logging enabled, throughput decreases from 5.4 KB/s to 3.9 KB/s—a 27.8% reduction. This drop reflects the resource-intensive nature of logging operations and their impact on the real-time data handling capabilities of the glasses.

To mitigate these performance impacts, a macro has been introduced in the firmware that allows logging to be enabled or disabled as needed. This feature enables the aRdent glasses to operate without logging during performance-critical tasks, ensuring optimal BLE throughput. Logging can be activated during testing or when detailed diagnostics are necessary, providing flexibility and maintaining device efficiency.

8.7.7 Results

During the tests, several issues occurred. First, short circuit problems were discovered on aRdent 2, which caused a postponement of the tests. Additionally, the firmware was slightly modified before the ECM tests. Specifically, a merge combined work on Bluetooth optimization and integration into the final version of the aRdent glasses. This led to last-minute issues where continuous data transmission over long periods caused the glasses to crash.

As a result, calculating the data rate was complicated and subject to many failures. Integrity tests could not be effectively performed because the text files containing data descriptions were unusable due to consecutive stops and differences between the text file content on the computer sending the data and those on the card.

However, for certain antenna powers used in the University of Liège laboratory, we were able to obtain some measurements. Here are the measurements:

Test	Face	Orientation	Time to Send (s)	Throughput (KB/s)	Frequency (MHz)
3	Front	Horizontal	228.578	1.285	600.0
4	Front	Vertical	261.831	1.295	600.0
5	Right	Vertical	271.691	1.248	600.0
6	Right	Horizontal	276.904	1.224	600.0
7	Right	Horizontal	274.522	1.235	140.0
8	Right	Vertical	276.777	1.225	140.0
9	Back	Vertical	275.321	1.231	140.0
10	Back	Horizontal	274.952	1.233	140.0
12	Back	Vertical	276.112	1.228	600.0
15	Left	Horizontal	283.860	1.194	140.0
16	Left	Vertical	273.401	1.240	140.0
17	Front	Vertical	18.625	1.213	2633.333
18	Front	Vertical	19.247	1.174	5000.0
19	Front	Horizontal	18.857	1.198	5000.0
20	Front	Horizontal	18.975	1.191	2633.333
21	Right	Horizontal	19.886	1.136	2633.333
22	Right	Horizontal	20.017	1.129	5000.0
23	Right	Vertical	19.851	1.138	5000.0
24	Right	Vertical	19.759	1.144	2633.333
25	Back	Vertical	18.732	1.206	2633.333
26	Back	Vertical	23.963	0.943	5000.0
27	Back	Horizontal	18.515	1.221	5000.0
28	Back	Horizontal	18.904	1.196	2633.333
29	Left	Horizontal	18.644	1.212	2633.333
30	Left	Horizontal	18.849	1.199	5000.0
31	Left	Vertical	19.084	1.184	5000.0
32	Left	Vertical	18.975	1.191	2633.333

Table 8.1: Measurement Results of Data Throughput under Different Frequencies

The data from Table 8.1 was analyzed to understand the relationship between the frequency and the throughput.

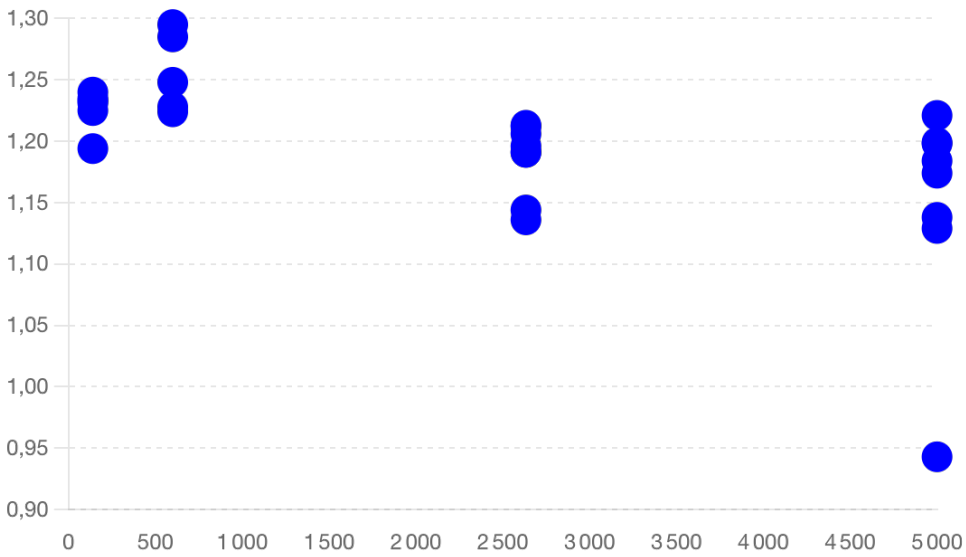


Figure 8.4: Relationship between Frequency and Throughput

The plot in Figure 8.4 shows that throughputs vary depending on the applied frequencies. There appears to be an inverse relationship between frequency and average throughput, where higher frequencies tend to have lower average throughputs.

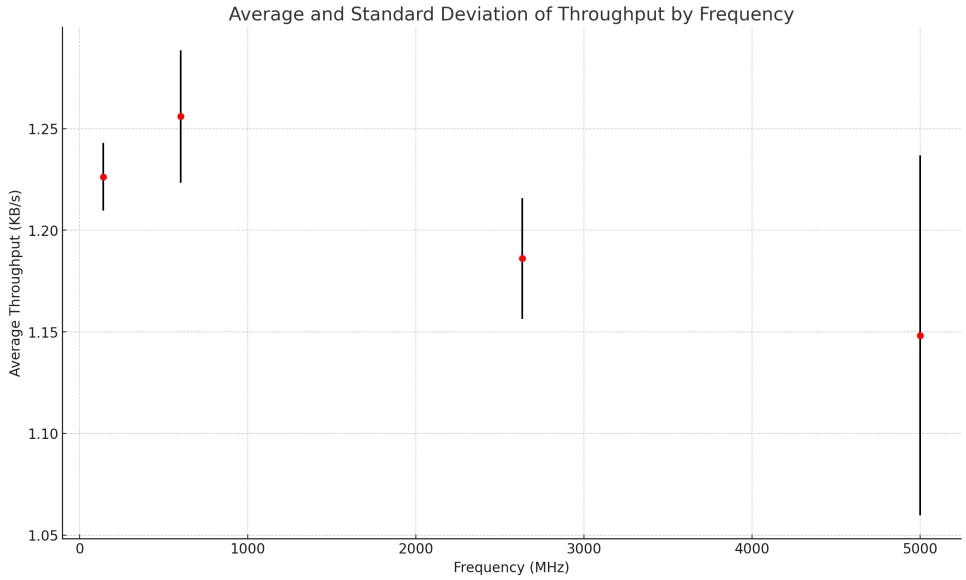


Figure 8.5: Average and Standard Deviation of Throughput by Frequency

Figure 8.5 presents the average and standard deviation of throughput for each frequency group. For example, at 140 MHz, the average throughput is 1.226 KB/s with very low variation, indicating stability. At 600 MHz, the average throughput is slightly higher at 1.256 KB/s, also showing low variation. However, at 2633.33 MHz, the average throughput decreases to 1.186 KB/s, and at 5000 MHz, it is the lowest at 1.148 KB/s with greater variability.

Although these differences are noticeable, the induced interferences during the tests showed that retransmissions indeed slow down the throughput, but not significantly. This reinforces the fact that the aRdent glasses can support high interference environments without significantly affecting their performance.

Chapter 9

Conclusion

The primary objective of this thesis was to enhance the Bluetooth Low Energy (BLE) performance for GYW's aRdent glasses. This involved optimizing various aspects of the embedded software running on FreeRTOS to improve the efficiency and reliability of the device's communication capabilities. The focus was on refining the BLE communication parameters, enhancing data transfer processes, and improving the methods used to receive data within the embedded software.

Through targeted optimizations, the data throughput was significantly increased from 0.3 KB/s to 6 KB/s, representing an increase of 1900%. This enhancement not only demonstrated the effectiveness of the optimizations but also marked a substantial improvement in the device's performance, making it more reliable and efficient in real-world applications. The improvements in data throughput had several positive impacts. Firstly, the speed of data transmission was greatly enhanced, allowing for faster and more efficient data exchanges between devices. This was particularly beneficial for applications requiring real-time data transfer, where the reduced latency and higher throughput contributed to smoother and more responsive interactions.

Secondly, the increased throughput facilitated quicker firmware updates, reducing the time required for maintenance and upgrades. Faster firmware updates are crucial for maintaining device security and performance, as they allow for timely application of patches and new features. This not only enhances the longevity of the device but also ensures that users benefit from the latest improvements without significant downtime. The overall user experience was significantly improved due to faster and more reliable connections. Users could enjoy a more seamless interaction with the device, as the optimizations led to more stable connections and quicker data transfer rates. This improvement in user experience is critical in various scenarios, ranging from everyday consumer use to specialized industrial applications where reliability and efficiency are paramount.

Furthermore, applying these modifications to the aRdent 2 glasses resulted in an even greater data throughput of 40 KB/s. This substantial increase further validates the effectiveness of the applied optimizations and highlights the potential for continued improvements in future iterations of the device. The advancements achieved with the aRdent 2 glasses not only demonstrate the scalability of the optimizations but also suggest a promising pathway for future enhancements that could further elevate the device's performance and user satisfaction.

Preparation for the Electromagnetic Compatibility (CEM) tests equipped GYW with robust tools to continuously assess and compare the performance of their glasses. This setup is crucial for maintaining high standards in product quality and performance across different iterations. This project was an invaluable opportunity to apply the rigorous principles of firmware development and deepen understanding of BLE protocols. It enhanced

problem-solving skills, particularly in diagnosing and addressing issues related to data throughput. Working in a startup environment emphasized the importance of teamwork and adaptability in achieving project goals.

Throughout the course of the thesis, multiple frameworks and various purposes were utilized, enriching software development skills. The initial phase involved learning the initial environment, Python libraries, and modifications to the firmware of the aRdent glasses. Following this, C++ programming was utilized for Arduino, focusing on low-level control and data acquisition tasks, ensuring efficient and precise hardware functioning.

The work then involved modifying an existing Flutter application to meet specific needs during BLE tests and optimizations. Flutter's versatility and cross-platform capabilities allowed for seamless interaction with BLE-enabled devices, ensuring robust testing environments across different platforms.

The core of the project revolved around firmware development for the aRdent glasses, coded in C. This involved deep engagement with embedded systems programming, where performance optimization and efficient resource management were crucial. Writing the firmware required a thorough understanding of the hardware specifics and BLE protocols to ensure reliable and efficient operation.

Additionally, Python scripts were developed to automate the testing process, simulate various data transmission scenarios, and analyze the performance of the BLE communication. Python's simplicity and powerful libraries made it an ideal choice for scripting and testing purposes. Overall, these endeavors culminated in a total of 2420 lines of code, providing a comprehensive understanding of different programming paradigms and their applications. Working across these diverse platforms and languages was both challenging and rewarding, significantly contributing to the successful completion of the thesis.

The advancement of BLE technologies, particularly through the Bluetooth Higher Data Throughput project, suggests promising avenues for further enhancing the product. Future implementations could consider integrating a BLE chip capable of supporting a 2 Mbps LE PHY, potentially doubling the current data transfer rates. Additionally, exploring efficient data compression methods and utilizing open-source BLE tools for testing interference mitigation strategies could further optimize performance.

Bibliography

[1] *A Developer's Guide to Bluetooth.* [Online]. URL: https://www.bluetooth.com/blog/a-developers-guide-to-bluetooth/.

[2] *A Developer's Guide to Bluetooth.* [Online]. URL: https://www.bluetooth.com/blog/a-developers-guide-to-bluetooth/.

[3] *Adaptive Frequency Hopping Spread Spectrum (AFHSS).* [Online]. URL: https://en.wikipedia.org/wiki/Frequency-hopping_spread_spectrum.

[4] *Advanced Encryption Standard (AES-128).* [Online]. URL: https://fr.wikipedia.org/wiki/Advanced_Encryption_Standard.

[5] *API.* [Online]. URL: https://en.wikipedia.org/wiki/API.

[6] *Apple Developer Forum.* URL: https://developer.apple.com/forums/.

[7] EMEA Application. *BlueNRG family MTU_ DLE_ 2Mpbs_ data_ rate_ improvement.* Available online. Oct. 2020.

[8] *Application Layer.* [Online]. URL: https://en.wikipedia.org/wiki/Application_layer.

[9] *Arduino.* [Online]. URL: https://www.arduino.cc/en/Guide/Introduction.

[10] *Assisted Reality.* [Online]. URL: https://blog.amaxperteye.com/what-is-assisted-reality-here-is-what-you-need-to-know#:~:text=Abbreviated%20as%20aR%2C%20assisted%20Reality,without%20blocking%20the%20user's%20vision..

[11] *Attribute Maximum Transmission Unit (ATT MTU).* [Online]. URL: https://punchthrough.com/maximizing-ble-throughput-part-2-use-larger-att-mtu-2/.

[12] *Attribute Protocol (ATT).* [Online]. URL: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/attribute-protocol--att-.html.

[13] *Augmented Reality.* [Online]. URL: https://www.techtarget.com/whatis/definition/augmented-reality-AR#:~:text=Augmented%20reality%20(AR)%20is%20the,overlaid%20on%20top%20of%20it..

[14] *Benchmark.* [Online]. URL: https://www.investopedia.com/terms/b/benchmark.asp.

[15] M. EL-Benday et al. "Power-Efficient of Image Transmission over Bluetooth System Using Randomization Technique". In: *Third International Conference: E-Medical Systems.* Morocco, May 2010.

[16] *Bleak Library.* [Online]. URL: https://bleak.readthedocs.io/en/latest/.

[17] *Bluetooth.* [Online]. URL: https://en.wikipedia.org/wiki/Bluetooth.

[18] *Bluetooth 5 Speed - Maximum Throughput*. Jan. 2024. URL: https://novelbits.io/bluetooth-5-speed-maximum-throughput/#:~:text=If%20you%20know%20that%20the,two%20birds%20with%20one%20stone!.

[19] *Bluetooth 5 Speed Maximum Throughput*. Jan. 2024. URL: https://novelbits.io/bluetooth-5-speed-maximum-throughput/.

[20] *Bluetooth Low Energy (BLE)*. [Online]. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

[21] *BlueZ*. [Online]. URL: https://www.bluez.org/about/.

[22] *Bottleneck (software)*. [Online]. URL: https://en.wikipedia.org/wiki/Bottleneck_(software).

[23] *Buffer Overflow*. [Online]. URL: https://en.wikipedia.org/wiki/Buffer_overflow.

[24] P. Bulic, G. Kojek, and A. Biasizzo. "Data Transmission Efficiency in Bluetooth Low Energy Versions". In: *Sensors* 19.3746 (2019).

[25] *C (programming language)*. [Online]. URL: https://en.wikipedia.org/wiki/C_(programming_language).

[26] *Channel Selection Algorithms (CSA)*. [Online]. URL: https://it.mathworks.com/help/bluetooth/ug/bluetooth-le-channel-selection-algorithms.html#mw_rtc_BLEChannelHoppingExample_M_041FCD2E.

[27] *Clock*. [Online]. URL: https://kb.iu.edu/d/aekt#:~:text=The%20clock%20ensures%20that%20the,used%20to%20measure%20clock%20speed..

[28] *Communication Protocol*. [Online]. URL: https://en.wikipedia.org/wiki/Communication_protocol.

[29] *Communication Protocol - Payload*. [Online]. URL: https://en.wikipedia.org/wiki/Communication_protocol.

[30] *Concepts Across the Sciences: Stability and Change*. [Online]. URL: https://blogs.loc.gov/teachers/2023/03/concepts-across-the-sciences-stability-and-change/#:~:text=Stability%20refers%20to%20the%20tendency,short%20or%20long%20time%20intervals.

[31] *Connection Event*. [Online]. URL: https://punchthrough.com/manage-ble-connection/.

[32] *Connection Interval*. [Online]. URL: https://punchthrough.com/manage-ble-connection/#:~:text=The%20Supervision%20Timeout%20is%20a,scanning%20in%20order%20to%20reconnect..

[33] *CoreBluetooth*. [Online]. URL: https://developer.apple.com/documentation/corebluetooth.

[34] *CSA 1*. [Online]. URL: https://it.mathworks.com/help/bluetooth/ug/bluetooth-le-channel-selection-algorithms.html#mw_rtc_BLEChannelHoppingExample_M_041FCD2E.

[35] *CSA 2*. [Online]. URL: https://it.mathworks.com/help/bluetooth/ug/bluetooth-le-channel-selection-algorithms.html#mw_rtc_BLEChannelHoppingExample_M_041FCD2E.

[36] *Current Consumption Estimation Tool provided by ST Electronics*. [Online]. URL: https://www.st.com/en/embedded-software/stsw-bnrg001.html.

[37] *Cyclic Redundancy Check*. [Online]. URL: https://en.wikipedia.org/wiki/Cyclic_redundancy_check.

[38] *Data Buffer*. [Online]. URL: https://en.wikipedia.org/wiki/Data_buffer.

[39] *Data Integrity*. [Online]. URL: https://en.wikipedia.org/wiki/Data_integrity.

[40] *Data Length Extension (DLE)*. [Online]. URL: https://software-dl.ti.com/lprf/simplelink_cc2640r2_latest/docs/blestack/ble_user_guide/html/ble-stack-3.x/data-length-extensions.html.

[41] *Efficiency*. [Online]. URL: https://en.wikipedia.org/wiki/Efficiency.

[42] *Electromagnetic Compatibility (EMC) Tests*. [Online]. URL: https://www.intertek.com/emc/#:~:text=EMC%20testing%20helps%20identify%20potential,risk%20of%20harm%20to%20users..

[43] *Embedded software*. [Online]. URL: https://en.wikipedia.org/wiki/Embedded_software.

[44] *Empirical research*. [Online]. URL: https://en.wikipedia.org/wiki/Empirical_research.

[45] *Firmware*. [Online]. URL: https://en.wikipedia.org/wiki/Firmware.

[46] *Flutter (software)*. [Online]. URL: https://en.wikipedia.org/wiki/Flutter_(software).

[47] *FreeRTOS*. [Online]. URL: https://en.wikipedia.org/wiki/FreeRTOS.

[48] *Generic Access Profile (GAP)*. [Online]. URL: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-generic-access-profile-gap/.

[49] *Generic Attribute Profile (GATT)*. [Online]. URL: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-gap-gatt/.

[50] C. Gomez, J. Oller, and J. Paradells. "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology". In: *Sensors* 12 (2012), pp. 11734–11753. DOI: 10.3390/s120911734.

[51] *Host Controller Interface (HCI)*. [Online]. URL: https://software-dl.ti.com/lprf/simplelink_cc2640r2_latest/docs/blestack/ble_user_guide/html/ble-stack-3.x/hci.html.

[52] *Interference Awareness Scheme (IAS)*. [Online]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8037550/.

[53] *ISM radio band*. [Online]. URL: https://en.wikipedia.org/wiki/ISM_radio_band.

[54] *JLinkRTTViewer*. [Online]. URL: https://www.segger.com/products/debug-probes/j-link/tools/rtt-viewer/.

[55] R. Katila, T. Nguyen Gia, and T. Westerlund. "Analysis of Mobility Support Approaches for Edge-based IoT Systems Using High Data Rate Bluetooth Low Energy 5". In: *Computer Networks* 209 (2022), Art. no. 108925.

[56] *Latency (Engineering)*. [Online]. URL: https://en.wikipedia.org/wiki/Latency_(engineering).

[57] *LE 1M PHY*. [Online]. URL: https://punchthrough.com/crash-course-in-2m-bluetooth-low-energy-phy/.

[58] *LE 2M PHY*. [Online]. URL: https://punchthrough.com/crash-course-in-2m-bluetooth-low-energy-phy/.

[59] *Link Layer*. [Online]. URL: https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/bluetooth-controller-layer/bluetooth-link-layer/#:~:text=The%20Bluetooth%C2%AE%20Low%20Energy,%2C%20and%20creating%2Fmaintaining%20connections..

[60] *Logical Link Control and Adaptation Protocol (L2CAP)*. [Online]. URL: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/logical-link-control-and-adaptation-protocol-specification.html.

[61] *Maximizing BLE Throughput on iOS and Android*. Dec. 2023. URL: https://punchthrough.com/maximizing-ble-throughput-on-ios-and-android/.

[62] *Maximizing BLE Throughput Part 2: Use Larger ATT MTU*. URL: https://punchthrough.com/maximizing-ble-throughput-part-2-use-larger-att-mtu-2/.

[63] *Maximizing BLE Throughput Part 3: Data Length Extension (DLE)*. URL: https://punchthrough.com/maximizing-ble-throughput-part-3-data-length-extension-dle-2/.

[64] *Maximizing BLE Throughput Part 4: Everything You Need to Know*. URL: https://punchthrough.com/ble-throughput-part-4/.

[65] *Message Authentication Code*. [Online]. URL: https://en.wikipedia.org/wiki/Message_authentication_code.

[66] *Microcontroller*. [Online]. URL: https://en.wikipedia.org/wiki/Microcontroller.

[67] *Nordic nRF52840*. [Online]. URL: https://www.nordicsemi.com/Products/nRF52840.

[68] *Nordic Semiconductor Information Center*. Jan. 2024. URL: https://infocenter.nordicsemi.com/index.jsp?topic=/sds_s140/SDS/s1xx/ble_data_throughput/ble_data_throughput.html.

[69] *Operating System*. [Online]. URL: https://en.wikipedia.org/wiki/Operating_system.

[70] *PacketLogger*. [Online]. URL: https://developer.apple.com/bluetooth/.

[71] B. Pang et al. "Bluetooth Low Energy Interference Awareness Scheme and Improved Channel Selection Algorithm for Connection Robustness". In: *Sensors* 21.2257 (2021).

[72] E. Park et al. "AdaptaBLE: Adaptive Control of Data Rate, Transmission Power, and Connection Interval in Bluetooth Low Energy". In: *Computer Networks* 181 (2020), Art. no. 107520.

[73] *Physical Layer (PHY)*. [Online]. URL: https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/bluetooth-controller-layer/physical/#:~:text=The%20Bluetooth%C2%AE%20Low%20Energy,services%20to%20the%20link%20layer..

[74] *Protocol Stacks Layered Architecture*. [Online]. URL: https://novelbits.io/protocol-stacks-layered-architecture/.

[75] *PyOBJC*. [Online]. URL: https://pyobjc.readthedocs.io/en/latest/.

[76] *Python (programming language).* [Online]. URL: https://en.wikipedia.org/wiki/Python_(programming_language).

[77] *Quality of Service (QoS).* [Online]. URL: https://en.wikipedia.org/wiki/Quality_of_service.

[78] *Race Condition.* [Online]. URL: https://en.wikipedia.org/wiki/Race_condition.

[79] *Secure Shell (SSH).* [Online]. URL: https://en.wikipedia.org/wiki/Secure_Shell.

[80] *Security Manager (SM).* [Online]. URL: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/security-manager-specification.html#UUID-193f95ea-7252-1b51-853b-a1999393dddf.

[81] *Serial Peripheral Interface (SPI).* [Online]. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[82] C. Shao and S. Nirjon. "Demo Abstract: Image Storage and broadcast via Bluetooth Low Energy Beacons". In: *Proceedings of the 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation.* Pittsburgh, PA, USA, Apr. 2017, pp. 1–2. DOI: 10.475/123_4.

[83] *ST Electronics Community Forum.* Dec. 2023. URL: https://community.st.com/.

[84] *Static Random Access Memory (SRAM).* [Online]. URL: https://fr.wikipedia.org/wiki/Static_Random_Access_Memory.

[85] STMicroelectronics. *BlueNRG BlueNRG-MS Stacks Programming Guidelines.* Programming Manual, DocID027104 Rev 7. Available online: www.st.com. Nov. 2018.

[86] STMicroelectronics. *Slot Allocation and Multiple Connection Timing Strategy for BlueNRG, BlueNRG-MS, BlueNRG-1, and BlueNRG-2.* Design Tip DT0107, Rev 2. Available online: www.st.com. Nov. 2018.

[87] A. K. Sultania, C. Delgado, and J. Famaey. "Enabling Low-Latency Bluetooth Low Energy on Energy Harvesting Batteryless Devices Using Wake-Up Radios". In: *Sensors* 20.5196 (2020).

[88] *Supervision Timeout.* [Online]. URL: https://punchthrough.com/manage-ble-connection/.

[89] *The Inter Frame Space (IFS).* [Online]. URL: https://novelbits.io/bluetooth-5-speed-maximum-throughput/.

[90] K. Townsend et al. *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking.* 1st ed. O'Reilly Media, Incorporated, 2014. ISBN: 978-1491949511.

[91] *Universally Unique Identifier (UUIDs).* [Online]. URL: https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/bluetooth-host-layer/bluetooth-generic-attribute-profile-gatt/UUIDs/#:~:text=A%20Universally%20Unique%20Identifier%20(UUID,for%20shortened%2016%2Dbit%20UUIDs..

[92] *UTF-8.* [Online]. URL: https://en.wikipedia.org/wiki/UTF-8.

[93] *Wave interference.* [Online]. URL: https://en.wikipedia.org/wiki/Wave_interference.

[94] *Wi-Fi.* [Online]. URL: https://en.wikipedia.org/wiki/Wi-Fi.

[95] *WireShark.* [Online]. URL: https://www.wireshark.org/about.html.