Enhancing Bluetooth Communication Security and Location Accuracy with RTK Technology in IoT Systems

Guanyi Heng

Department of Electrical & Computer Engineering
McGill University
Montréal, Québec, Canada

August 30, 2024

A thesis presented for the degree of Masters of Electrical Engineering

Abstract

The rapid expansion of the Internet of Things (IoT) has dramatically increased the demand for secure and efficient communication protocols, particularly for Bluetooth technology. This thesis presents a comprehensive approach to enhancing Bluetooth communication security and incorporating RTK technology to improve location-based services.

This research begins by implementing a security protocol for Bluetooth connection utilizing the security co-processor. This protocol ensures robust authentication and data integrity, mitigating common security vulnerabilities. Detailed profiling of the connection establishment process highlights the protocol's efficiency and reliability.

The study aims to enhance location accuracy by integrating RTK technology with Bluetooth communication. It involves detailed protocol setup and rigorous session management. The results show significant improvements in location accuracy, achieving an accuracy of 0.445 meters in latitude, 0.17 meters in longitude, and 0.38 meters in altitude, even with the nearest RTK base station 10 kilometres away.

Moreover, the thesis explores the system's effectiveness in concurrently managing multiple applications utilizing diverse communication protocols. It gauges the performance in transmitting data to multiple applications simultaneously, maintaining high throughput and efficient data transmission. The analysis also explores the impact of increasing baud rates on data throughput and transmission efficiency, revealing that while higher baud rates can enhance throughput, they may also introduce transmission errors.

Furthermore, a location shift functionality is implemented, allowing users to apply predefined shifts to latitude, longitude, and altitude. This feature is tested and validated, demonstrating the system's ability to handle accurately and display shifted geospatial data. The thesis also includes a robust error-handling mechanism, employing an Independent Watchdog (IWDG) timer to ensure continuous data transmission and system stability.

In conclusion, the developed security protocol and the integration of RTK technology significantly enhance the security of Bluetooth communications in IoT applications and hold potential for real-world applications. The findings contribute to advancing secure and efficient IoT systems.

Abrégé

L'expansion rapide de l'Internet des objets (IoT) a considérablement augmenté la demande de protocoles de communication sécurisés et efficaces, en particulier pour la technologie Bluetooth. Cette thèse présente une approche complète pour améliorer la sécurité des communications Bluetooth et intégrer la technologie RTK afin d'améliorer les services basés sur la localisation.

Cette recherche commence par la mise en œuvre d'un protocole de sécurité pour la connexion Bluetooth en utilisant le coprocesseur de sécurité. Ce protocole assure une authentification robuste et l'intégrité des données, atténuant ainsi les vulnérabilités de sécurité courantes. Un profil détaillé du processus d'établissement de la connexion met en évidence l'efficacité et la fiabilité du protocole.

L'étude vise à améliorer la précision de la localisation en intégrant la technologie RTK avec la communication Bluetooth. Elle implique une configuration détaillée du protocole et une gestion rigoureuse des sessions. Les résultats montrent des améliorations significatives de la précision de la localisation, atteignant une précision de 0,445 mètres en latitude, 0,17 mètres en longitude et 0,38 mètres en altitude, même avec la station de base RTK la plus proche à 10 kilomètres.

De plus, la thèse explore l'efficacité du système dans la gestion simultanée de plusieurs applications utilisant divers protocoles de communication. Elle évalue les performances de transmission des données vers plusieurs applications simultanément, en maintenant un débit élevé et une transmission de données efficace. L'analyse explore également l'impact de l'augmentation des débits en bauds sur le débit de données et l'efficacité de la transmission, révélant que bien que des débits en bauds plus élevés puissent améliorer le débit, ils peuvent également introduire des erreurs de transmission.

En outre, une fonctionnalité de déplacement de la localisation est implémentée, permettant aux utilisateurs d'appliquer des décalages prédéfinis à la latitude, la longitude et l'altitude. Cette fonctionnalité est testée et validée, démontrant la capacité du système à gérer et afficher avec précision les données géospatiales décalées. La thèse comprend également un mécanisme robuste de gestion des erreurs, utilisant un timer de chien de garde indépendant (IWDG) pour assurer une transmission continue des données et la stabilité du système.

En conclusion, le protocole de sécurité développé et l'intégration de la technologie RTK améliorent considérablement la sécurité des communications Bluetooth dans les applications IoT et présentent un potentiel pour des applications réelles. Les résultats contribuent à l'avancement de systèmes IoT sécurisés et efficaces.

Acknowledgements

I am deeply thankful to my supervisor, Prof. Zeljko Zilic, whose unwavering guidance and endless patience have formed the cornerstone of this thesis. His insightful wisdom and inspiring mentorship have greatly advanced my academic journey, making this achievement possible.

I would like to extend my thanks to Prof. Zilic for his kind and patient support and for developing the mobile device application that displays the GNSS data as a read-only protocol, which allowed me to test my system effectively.

I am especially grateful to Yuxiang Ma, who developed the mobile device application to send commands and display the GNSS data that supports the read-and-write protocol. His contribution greatly helped me to test and accomplish my system.

I want to extend my deep appreciation to all the lab members for their invaluable support and moral encouragement, which have been crucial in the completion of my thesis. In particular, I am grateful to Shaluo Wu for creating an additional application that allowed me to test and evaluate my accessory for supporting multiple protocols with data transmission.

I also want to thank Zice Tang for his support and assistance with debugging.

Special thanks go to Wenxi and my parents for their unwavering support, which helped me navigate through the Master's program. Wenxi has been a constant source of encouragement and emotional support.

Contents

List of Figures

List of Tables

List of Acronyms

AES	Advanced Encryption Standard.
ATT	Attribute Protocol.
BLE	Bluetooth Low Energy.
BR/EDR	Basic Rate/Enhanced Data Rate.
CRC	Cyclic Redundancy Check.
CRL	Certificate Revocation List.
DMA	Direct Memory Access.
FKP	Flächen-Korrektur Parameter.
FSM	Finite State Machine.
GAP	Generic Access Profile.
GATT	Generic Attribute Profile.
GNSS	Global Navigation Satellite System.
GPIO	General-Purpose Input/Output.
HCI	Host Controller Interface.

HMAC	Hash-based Message Authentication Code.
HTTP	Hypertext Transfer Protocol.
I2C	Inter-Integrated Circuit.
iMAX	Individualized Master-Auxiliary Corrections.
IoT	Internet of Things.
IP	Internet Protocol.
ISM	Industrial, Scientific, and Medical.
IWDG	Independent Watchdog.
L2CAP	Logical Link Control and Adaptation Protocol.
MAC	Media Access Control.
NMEA	National Marine Electronics Association.
NTRIP	Networked Transport of RTCM via Internet Protocol.
OCSP	Online Certificate Status Protocol.
OS	Operating System.
PKI	Public Key Infrastructure.
PPP	Precise Point Positioning.
PRS	Pseudo Reference Station.
PSK	Pre-Shared Key.
RF	Radio Frequency.
RFCOMM	Radio Frequency Communication.

RTCM	Radio Technical Commission for Maritime Services.
RTK	Real-Time Kinematic.
SDP	Service Discovery Protocol.
SHA	Secure Hash Algorithm.
SMP	Security Manager Protocol.
SPP	Serial Port Profile.
TLS	Transport Layer Security.
UART	Universal Asynchronous Receiver-Transmitter.
UHF	Ultra High Frequency.
USB	Universal Serial Bus.
VRS	Virtual Reference Station.

Chapter 1

Introduction

This chapter explains the importance of secure Bluetooth communication in the expanding Internet of Things (IoT) industry. With Bluetooth being used in various IoT applications such as healthcare and smart homes, maintaining strong security measures is crucial to prevent data breaches. Additionally, the chapter introduces Real-Time Kinematic (RTK) technology, which enhances location accuracy for autonomous vehicles and precision agriculture applications. This thesis focuses on developing a secure Bluetooth protocol, improving location accuracy with RTK, and effectively managing multiple applications. Finally, the chapter provides an overview of the thesis structure, outlining the key topics covered in the subsequent chapters.

1.1 Motivation

Bluetooth technology has become a critical part of our lives. It enables seamless communication between devices over short distances. As we move closer into the Internet of Things (IoT) era, the demand for secure, efficient, and reliable Bluetooth communication increases. The rise of IoT devices has propelled Bluetooth into the spotlight due to its low power consumption and high efficiency. Bluetooth is now a backbone for various applications, ranging from smart healthcare systems and wearable devices to industrial automation and smart homes [6].

For instance, IoT devices are crucial in monitoring patient health and transmitting sensitive data in the healthcare sector, indicating the importance of robust security measures to protect against cyber threats. A security breach in healthcare data can have unpredictable consequences [7]. As the number of connected devices grows, the potential for vulnerabilities can be raised. Advanced security protocols are needed.

Beyond security, integrating Real-Time Kinematic (RTK) technology with Bluetooth significantly enhances location accuracy. RTK provides centimetre-level precision. It is essential for autonomous vehicles, precision agriculture, and land surveying applications. This level of accuracy ensures that location-based services are reliable and highly accurate, meeting the demands of modern location-based applications.

The ever-increasing deployment of IoT devices requires the development of secure and efficient Bluetooth communication protocols. Integrating RTK technology further enhances the accuracy and reliability of these systems. This thesis aims to address these needs by developing a secure Bluetooth communication protocol and integrating RTK technology to improve location accuracy, thus contributing to the security of IoT communications.

1.2 Contribution To Knowledge

This thesis aims to advance the field of secure wireless communication by developing a robust security protocol for Bluetooth connections. Then, the integrated RTK technology will improve location accuracy. The key contributions of this work include:

- **Implementation of Secure Bluetooth Communication**: Using HMAC-SHA256 and RFC5208 protocol for robust authentication, certificate and data integrity, this research ensures the security of Bluetooth communication against potential attacks.
- **Evaluation of RTK Performance**: By integrating RTK technology, the study demonstrates significant improvements in location accuracy despite practical challenges.
- **Development of Multi-Application Concurrent Connections**: This work showcases the system's capability to handle multiple protocols and optimize data throughput. It can enhance the overall performance and reliability of Bluetooth communication.

- **Analysis of Location Shift Functionality**: The ability of the system to accurately manage and display shifted geospatial data is validated, emphasizing the importance of precise data handling in geospatial applications.
- **Implementation of a Reconnection Protocol for Error Handling**: Ensuring system stability and reliability, the reconnection protocol addresses transmission failures and signal loss, maintaining continuous and secure data transmission.

1.3 Thesis Organization

This thesis is organized as follows:

- **Chapter 2: Background and Literature Review** - Provides a comprehensive review of Bluetooth technology, RTK positioning, and security protocols, establishing a solid foundation for the research.
- **Chapter 3: Secure Bluetooth Connection Establishment** - Details the implementation of the secure Bluetooth connection, including the setup of the security protocol, authentication and identification processes, and packet encapsulation.
- **Chapter 4: Location Application Enhanced by RTK** - Discusses the integration of RTK technology into the location application, covering the protocol setup, session

establishment, and handling of RTK data for improved accuracy. In addition, the location shift function is carefully explained, which can expand the application's use cases.

- **Chapter 5: Results and Discussion** - Presents the experimental results and analysis, including the performance of the security protocol, RTK accuracy, multi-application concurrent connections, and location shift functionality.
- **Chapter 6: Conclusions and Future Work** - Summarizes the research findings, discusses the implications of the results, and suggests potential directions for future work in this field.

By addressing these key areas, this thesis aims to advance the understanding and implementation of secure Bluetooth communications and high-accuracy location services, providing a foundation for future innovations in IoT and related technologies.

Chapter 2

Background and Literature Review

This chapter provides an overview of the foundational concepts and technologies relevant to the research. It begins with a detailed discussion of Bluetooth technology, focusing on its two main types: Bluetooth Classic (BR/EDR) and Bluetooth Low Energy (BLE). The chapter compares their protocol stacks, functions, and use cases. It then explores Real-Time Kinematic (RTK) positioning, explaining its types, advantages, and applications in improving positional accuracy in IoT systems. The chapter also covers various authentication methods for IoT, including RFC 5280 with X.509 certificates and HMAC-SHA256, which ensure secure and reliable communication. This background sets the stage for understanding subsequent chapters' security and accuracy enhancements.

2.1 Bluetooth

Bluetooth is a wireless communication technology that utilizes short-range radio transmissions to exchange data between fixed and mobile devices. It operates using Ultra High Frequency (UHF) radio waves within the unlicensed Industrial, Scientific, and Medical (ISM) frequency band, ranging from 2.4 to 2.48 GHz. Bluetooth technology can be divided into Basic Rate/Enhanced Data Rate (BR/EDR) and Bluetooth Low Energy (BLE). Both types use the same stack structure: Application, Host, and Controller [1].

The stack's highest-level application layer represents the user's interface. The host stack manages high-level Bluetooth protocols and profiles, while the controller stack handles low-level Bluetooth operations and physical RF communication. Figure 2.1 illustrates Bluetooth Classic, BLE, and dual-mode protocol structure.

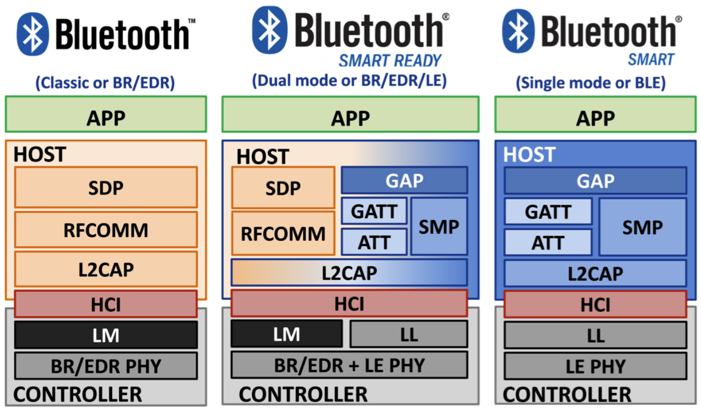


Figure 2.1: Protocol Structure of BR/EDR, BLE, and Dual Mode [1]

The following subsections will discuss Bluetooth Classic and BLE separately from the perspective of their Bluetooth host stack structures. Since our program mainly focuses on implementing the Host stack, we will also present a comparison between the two.

2.1.1 BR/EDR

In this section, we explain the functions of BR/EDR, with each paragraph headed by the respective acronym of a function, such as SDP, RFCOMM and L2CAP.

SDP The Service Discovery Protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services. The service discovery mechanism provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service. Fig. 2.2 indicates the SDP client-server interaction.

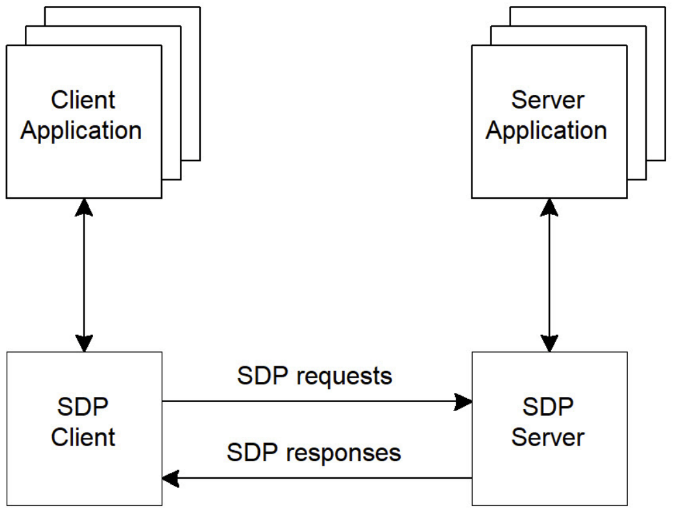


Figure 2.2: Client-Server Interaction for SDP [2]

RFCOMM RFCOMM (Radio Frequency Communication) is a Bluetooth protocol that emulates RS-232 serial ports over the Logical Link Control and Adaptation Protocol (L2CAP). Based on the ETSI GSM 07.10 standard, RFCOMM facilitates serial port communication by enabling data and control signal transfer between Bluetooth devices. It supports up to 60 simultaneous connections, allowing multiple emulated serial ports to operate concurrently between two devices. This built-in null modem emulation allows direct communication without physical adapters, making it ideal for applications needing serial communication, such as connecting computers, printers, and modems via Bluetooth [8].

L2CAP The Logical Link Control and Adaptation Protocol (L2CAP) is a key component of the Bluetooth protocol stack that facilitates data communication between devices. It operates over the Baseband layer and interfaces with the Host Controller Interface (HCI) to provide reliable data transmission. L2CAP supports the segmentation and reassembly of packets, allowing large data packets to be transmitted by breaking them into smaller segments [9].

2.1.2 BLE

Generic Access Profile (GAP) The Generic Access Profile (GAP) has three main purposes:

1. To define standards and common requirements for modes and access procedures used by transport and application profiles, including generic procedures for discovering Bluetooth devices (idle mode procedures) and link management for connecting to Bluetooth devices (connecting mode procedures).

2. To ensure devices behave correctly in standby and connecting states, focusing on discovery, link establishment, and security to prevent connection issues and support multi-profile operations, including procedures for using different security levels.

3. To specify user interface requirements, including coding schemes, procedure names, and common format requirements for parameters accessible on the user interface level,

ensuring a good user experience [10].

GATT The Generic Attribute Profile(GATT) establishes how data will be organized and transferred over a Bluetooth Low Energy(BLE) connection once devices have a dedicated connection. GATT uses the Attribute Protocol(ATT) as a transport mechanism [11].

ATT The Attribute Protocol is a protocol for discovering, reading, and writing attributes on a peer device. It allows a device to act as the server to broadcast a set of attributes and their corresponding values to a client. The client can access these attributes by using the ATT [12].

SMP The Security Manager(SM) defines the protocol and behaviour for managing pairing, authentication, and encryption between LE-only or dual-mode (BR/EDR/LE) devices. The Bluetooth Classic device cannot utilize this protocol.

The Security Manager handles key generation, distribution, and storage, ensuring that the encryption keys used in communication are robust and securely managed. By managing these keys and ensuring they are exchanged securely, the Security Manager helps protect against unauthorized access and eavesdropping [13].

2.1.3 Comparison Between Bluetooth Versions

Bluetooth Classic and Bluetooth Low Energy (BLE) are communication protocols under the Bluetooth standard but designed for different use cases and applications. Both Bluetooth Classic and BLE operate within the 2.4 GHz ISM band. Bluetooth Classic uses 79 channels, each 1 MHz wide, while BLE uses 40 channels, each 2 MHz wide. This difference in channel utilization impacts their performance and application areas [14].

The power consumption is the most significant difference between Bluetooth Classic and BLE. BLE is designed for ultra-low power consumption. that require long battery life, such as fitness trackers and medical devices. BLE achieves efficiency by remaining in sleep mode most of the time and only waking up to transmit data. In contrast, Bluetooth Classic consumes more power due to its continuous data transmission capabilities, making it suitable for applications like audio streaming where a stable and high-quality connection is required.

Bluetooth Classic supports higher data transfer rates, reaching 1-3 Mbps, making it better for applications requiring continuous, high-bandwidth data transfer, such as wireless headphones and speakers. BLE, with a maximum data rate of around 1 Mbps, is optimized for applications that need periodic, sparse data transmissions. BLE offers significantly lower latency (approximately 6 ms) compared to Bluetooth Classic's latency of around 100 ms, making BLE more responsive for real-time applications [14].

Due to its efficient power usage, BLE typically provides a longer range (up to 50 or 150 meters in open areas). In contrast, Bluetooth Classic offers higher throughput while having a shorter range (less than 30 meters). The topologies supported by Bluetooth Classic and BLE also differ. Bluetooth Classic primarily supports a peer-to-peer (1:1) topology, while BLE supports multiple topologies, including peer-to-peer (1:1), star (many:1), broadcast (1:many), and mesh (many: many) [15].

Bluetooth Classic requires device pairing, which can be a drawback for applications requiring quick and seamless connectivity. However, if the device is already paired, it can establish automatic reconnection and prevent unauthorized access. Conversely, BLE does not require device pairing, making it more suitable for applications needing rapid and transient connections. Both protocols support robust security measures, but BLE uses 128-bit AES encryption, providing strong security suitable for modern applications.

The following table. 2.1 summarizes the differences.

	Bluetooth Classic	**Bluetooth Low Energy**
Frequency Band	2.4 GHz ISM Band	2.4 GHz ISM Band
No. of Channels	79 one MHz Channel	40 two MHz Channel
Power Consumption	Low	Lower
Data Rate	1-3 Mbps	1 Mbps
Latency	Approx. 100 ms	Approx. 6 ms
Range	<30 m	50 m (150 m in open area)

Topology	Peer-to-peer	Peer-to-peer, Star, Broadcast, Mesh
Device pairing	Required	Not Required
Nodes/Active Slaves	7	Unlimited
Security	64b/128 bit, user-defined application layer	128 bits AES, user-defined application layer
Use Cases	Streaming applications like audio streaming, file transfer, and headsets	Smart home applications, medical devices, industrial monitoring, fitness trackers

Table 2.1: Comparison between Bluetooth Classic and Bluetooth Low Energy, the table is adjusted from [5]

2.2 Real-Time Kinematic (RTK) Positioning

Real-time kinematic (RTK) positioning is a technique used to enhance the precision of position data derived from satellite-based positioning systems such as GPS, GLONASS, Galileo, and BeiDou. RTK provides real-time corrections to the satellite signals, significantly improving the accuracy of position data to centimetre-level precision. RTK uses a fixed base station and one or more mobile receivers, known as rovers.

The following subsections discuss the concepts of RTK and introduce the different types of it. Network RTK is the type we utilize for the system; its calculation is discussed. Then, comparisons among these types are provided. Then, we show the engineering applications that utilize the different types of RTK to achieve high accuracy.

2.2.1 Understanding of RTK

RTK compares the signals received from GNSS satellites at a base station and a rover. The base station, located at a precisely known position, computes the errors in the satellite signals and transmits these corrections to the rover. The rover applies these corrections to its satellite data, enabling it to accurately determine its position. Correcting the rover's position based on calculated errors at the base station is called differential corrections.

2.2.2 Types of RTK

Single-Baseline RTK

Single-baseline RTK involves a single base station that provides corrections to a rover, which means the rover can only access the base station locally.

Advantages:

- High accuracy within a limited range (typically 10-20 km).
- Simpler setup compared to network RTK.

Limitations:

- Accuracy decreases with distance from the base station.
- Within the range, it has to ensure no obstacles will block the signal from the base station, as the atmospheric error can be unpredictable.

Network RTK (NRTK)

To reduce distance-dependent errors, Network RTK uses networked base stations to provide corrections over a larger area. Data collected from the base stations is transmitted to a control center(caster) in real-time instead of directly transferring to the rover. Also, with the known coordinates of reference stations, the control center can estimate the errors based on these coordinates to mitigate the atmospheric delays. Then, the caster broadcasts all the interpolated corrections, and the rover determines how to access them [16].

Advantages:

- Extended coverage area.
- Higher accuracy and reliability over larger distances.

Limitations:

- More complex and costly infrastructure.
- Dependent on a reliable communication network.

Calculation Formula for NRTK

Network RTK calculations involve generating corrections based on data from multiple base stations and interpolating these corrections for the rover's position. The main steps are [16]:

1. **Pseudorange Correction (PRC)**:

$$\text{PRC}_s^A(t) = q_s^A(t) + c \cdot dT_A(t) - c \cdot dt_s(t) - R_s^A(t)$$

   where $q_s^A(t)$ is the geometric range between satellite s and reference station A, c is the speed of light, $dT_A(t)$ is the clock error of the reference station, $dt_s(t)$ is the satellite clock error, and $R_s^A(t)$ is the residual error.

2. **Carrier Phase Correction (CPC)**:

$$\text{CPC}_s(t) = \phi_s^A(t) + c \cdot dT_A(t) - c \cdot dt_s(t) + N_s \cdot \lambda$$

   where $\phi_s^A(t)$ is the carrier phase measurement, N_s is the integer ambiguity, and λ is the wavelength of the carrier signal.

3. **Interpolation for Virtual Reference Station (VRS) [17]**:

$$\text{PRC}_s^{VRS}(t) = \text{PRC}_s^A(t) + \frac{\partial \text{PRC}}{\partial x}\Delta x + \frac{\partial \text{PRC}}{\partial y}\Delta y + \frac{\partial \text{PRC}}{\partial z}\Delta z$$

Where the partial derivatives represent the gradients of pseudo-range corrections, and Δx, Δy, and Δz are the differences in coordinates between the base station and the VRS, there are many other correction methods can also be used, including Flächen-Korrektur Parameter (FKP), Master-Auxiliary Concept (MAC), Individualized Master- Auxiliary Corrections (iMAX), and Pseudo Reference Station (PRS).

4. **Corrected Observations at the Rover**:

$$\rho_s^B(t)_{\text{corrected}} = \rho_s^B(t) + \text{PRC}_s^{VRS}(t)$$

$$\phi_s^B(t)_{\text{corrected}} = \phi_s^B(t) + \text{CPC}_s^{VRS}(t)$$

These equations correct the pseudorange and carrier phase measurements at the rover.

Fig. 2.3 shows the difference between a single-baseline RTK and the Network RTK.

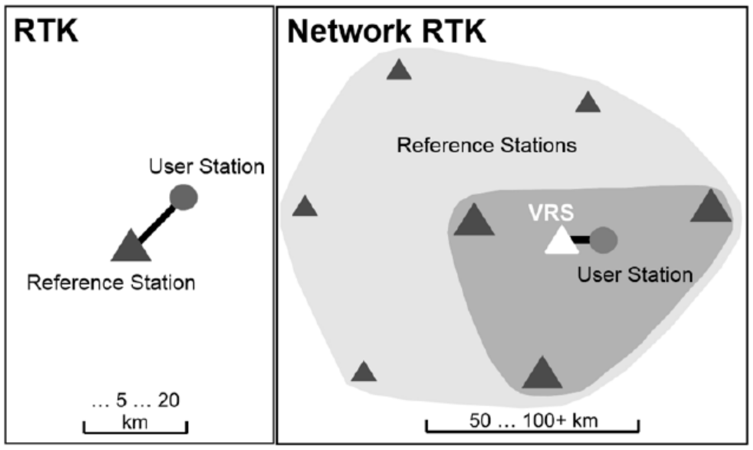


Figure 2.3: Principles of RTK and Network RTK [3]

Precise Point Positioning (PPP) with RTK

Description: Combines RTK corrections with Precise Point Positioning (PPP), allowing for highly accurate positioning without the need for a local base station [18].

Advantages:

- Global coverage.
- High accuracy without the need for nearby base stations.

Limitations:

- Requires a longer convergence time compared to traditional RTK.
- More complex data processing.

2.2.3 Comparison of RTK Types

Accuracy: All types of RTK provide high accuracy, but the distance to the base station limits single-baseline RTK, while NRTK offers more consistent accuracy over larger areas. PPP with RTK offers global accuracy but requires more processing time.

Range: Single-baseline RTK is effective within 10-20 km, NRTK covers a broader area with multiple base stations, and PPP with RTK provides global coverage.

Complexity: Single-baseline RTK is simpler to set up but limited in range. NRTK is more complex and requires a network of base stations. PPP with RTK is the most complex, requiring precise satellite orbit and clock data.

Cost: Single-baseline RTK is generally less expensive due to its simpler setup. NRTK requires more infrastructure and maintenance, making it expensive. PPP with RTK avoids the cost of local base stations. However, it involves significant computational resources and high-quality data subscriptions.

2.3 Authentication

2.3.1 Authentication Methods for IoT

Authentication in the Internet of Things (IoT) is essential for ensuring the integrity, trustworthiness, and privacy of devices and users within the ecosystem. As IoT devices proliferate across domains such as smart homes, healthcare, and industrial automation, robust authentication mechanisms are essential to protect data and prevent unauthorized access. Authentication in IoT confirms the identities of devices and users. It can preserve data reliability and security and reduce risks like data breaches, device compromise, and unauthorized access.

Several methods are employed to authenticate IoT devices, each with its consideration of security needs and resource constraints. Mehic et al. [19] proposed the Pre-shared key (PSK) methods to distribute a shared secret key among devices, requiring this key for network access. However, PSK lacks a well-defined sharing procedure in large-scale environments. Certificate-based authentication uses digital certificates signed by a trusted certificate authority. The authenticating device can achieve validity using the certification path validation algorithm specified in RFC 5280 [20]. Biometric-based authentication leverages unique biometric features like fingerprints, facial recognition, or iris scans, providing high security due to the uniqueness of biometric data and often integrating with blockchain platforms for mutual authentication.

One-time passwords (OTPs) generate unique passwords for each session, commonly used in online banking and financial transactions to prevent password reuse. Jiang et al. [21] proposed Token-based authentication that issues tokens for each session to validate identities, enhancing security through cloud-edge collaboration. Public key infrastructure (PKI) employs a public-private key pair for authentication, ensuring secure communication and adaptability in various IoT environments without constant reliance on trusted third parties [22]. Each method offers specific advantages and addresses unique challenges within the IoT ecosystem, ensuring robust and scalable security solutions.

2.3.2 RFC 5280 and X.509 Certificates for Authentication

RFC 5280 [23] defines the standards and protocols for using X.509 certificates and certificate revocation lists (CRLs) within the Internet Public Key Infrastructure (PKI). The process of creating an X.509 certificate begins with the generation of a public and private key pair. The public key is included in the certificate. However, the private key remains secured by the certificate holder. The Certification Authority (CA) then signs the certificate by adding its name and encrypting specific information using its private key. Essential fields within the certificate include the version, serial number, signature algorithm, issuer, validity period, subject, and subject public key information.

Critical parameters within the certificate, such as the issuer and the signature, are fundamental for establishing trust. Other important parameters include the certificate owner, expiration date, certificate policies, and revocation status. During the validation process, these parameters are checked to ensure they meet the necessary criteria. If the certificate has expired or the owner's information does not match expectations, the certificate is rejected. If all parameters are correct, they must still be verified by the issuer's signature. The issuer's signature is encrypted with the issuer's private key, providing extra security.

The certification path validation algorithm specified by RFC 5280 is crucial for validating certificates. This algorithm indicates the requirements that a certificate must meet to be considered valid. The chain of trust is a recursive validation process where each certificate in the chain is verified until a trusted root certificate is reached. This process verifies each certificate's signature using the issuer's public key, ensuring that each certificate in the chain is valid and trusted.

Certificate revocation is managed using Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP). These mechanisms provide information on whether a certificate has been revoked for reasons such as key compromise, cessation of operation, or change of affiliation. Trust in the certification system is anchored by root certificates issued by trusted CAs, which are widely distributed and trusted across systems. Intermediary certificates are also used to establish a chain of trust, ensuring that each certificate in the chain can be validated back to a trusted root certificate.

2.3.3 HMAC-SHA256 for IoT Authentication

HMAC (Hash-based Message Authentication Code) [24] is a widely used mechanism for ensuring data integrity and authenticity by combining cryptographic hash functions with a secret key. The hash function is denoted as H, and the secret key is K. The underlying hash function H can be used without modification to its code. HMAC with SHA-256 hash function is used by the secured Bluetooth connection presented in the thesis for authentication purposes.

HMAC employs a specific process that involves key padding, inner and outer padding with predefined constants, and two stages of hashing. Its security is based on the properties of the underlying hash function, such as SHA-256. The process ensures that even if the hash function has specific weaknesses, the overall HMAC construction remains secure.

The detailed construction for HMAC-SHA256 is as follows:

1. **Key Padding**: The secret key K is first processed to fit the block size of the hash function (64 bytes for SHA-256). If the key is longer than the block size, it is hashed to produce a 32-byte key. If the key is shorter, it is padded with zeros to make it 64 bytes. Longer keys are strongly recommended, as shorter keys decrease the security strength of the function.

2. **Inner and Outer Padding**: Two predefined constants, the inner pad (ipad) and the outer pad (opad), are used. The inner pad is a block-sized string consisting of the byte 0x36 repeated 64 times, while the outer pad is a block-sized string consisting of the byte 0x5c repeated 64 times.

3. **Key XOR with Pads**: The padded key is XORed with the inner pad to create the inner key and separately XORed with the outer pad to create the outer key. This procedure can help reduce the likelihood of hash collisions. The different values of the inner and outer pads can reduce the possibilities of differential attacks, where attackers

can exploit similar patterns in the key's usage.

4. **Inner Hash**: The inner key is concatenated with the message and hashed using SHA-256 to produce the inner hash.

5. **Outer Hash**: The outer key is concatenated with the inner hash and hashed again using SHA-256 to produce the final HMAC-SHA256 value. Using an inner and outer hash with separate key transformations helps mitigate length extension attacks. In these attacks, attackers can extend the message and compute a valid hash even without knowing the key. This process ensures that the final hash computation is unique and cannot be extended predictably.

The complete HMAC-SHA256 function can be expressed as:

$$\text{HMAC-SHA256}(K, \text{msg}) = H\left((K' \oplus \text{opad}) \parallel H((K' \oplus \text{ipad}) \parallel \text{msg})\right)$$

Where:

- H is the SHA-256 hash function.
- K' is the key after being hashed or padded.
- $\parallel$ denotes concatenation.
- $\oplus$ denotes bitwise exclusive-OR (XOR).

The construction of HMAC-SHA256 ensures robust message authentication, significantly enhancing the security properties of SHA-256. This scheme makes brute-force attacks practically impossible due to the extensive time required to succeed. Additionally, HMAC reduces the likelihood of collision attacks, further improving the security features of SHA-256. Since HMAC operates independently of the underlying hashing function, it offers flexibility for future upgrades, allowing the integration of stronger hashing functions as they become available.

2.3.4 Challenges and Proposed Solutions

Current Bluetooth connections, particularly within IoT environments, often lack robust security measures, rendering them susceptible to eavesdropping, unauthorized access, and data manipulation. The absence of proper authentication mechanisms and traffic control among devices introduces vulnerabilities in communication and data transmission.

To address these vulnerabilities, this thesis presents a comprehensive security protocol designed for Bluetooth communication. The protocol aims to authenticate connections and secure data packets exchanged between devices to safeguard integrity and confidentiality. By integrating X.509 certificates for certification and HMAC-SHA256 for message authentication, the solution effectively enhances the overall security of Bluetooth networks. Additionally, the protocol manages traffic between devices and applications to create a local, secure network.

In addition to securing Bluetooth communication, it's essential to expand this secure network to support high-precision location applications, which are crucial in IoT environments. Real-time kinematic (RTK) positioning technology offers centimetre-level precision, delivering significant advantages when integrated with secure communication networks. This thesis utilizes Network RTK (NRTK) to expand the functionality of secure Bluetooth connections. NRTK is chosen for its ability to provide high-precision positioning over a network, enhancing both security and application potential. Integrating NRTK with a secure Bluetooth network offers additional layers of security by ensuring that only authenticated and correctly positioned devices can communicate within the system. Furthermore, the high precision and network features of RTK open up new possibilities, such as integrating with blockchain technologies for location-based authentication and verification processes, providing even more security for sensitive applications.

Chapter 3

Secure Bluetooth Connection Establishment

This chapter provides a comprehensive analysis of the process involved in establishing a secure connection between mobile devices and accessories. It begins by explaining the module components and setup required to establish non-secured RFCOMM connections. This initial step is crucial as it lays the foundation for creating an essential Bluetooth connection. It enables the Bluetooth module to bridge between the mobile device and the microprocessor, facilitating the subsequent data exchange necessary for establishing a secure connection.

The chapter then introduces the security protocol, which enables the mobile device to authenticate the accessory and manage the state effectively. While it enhances the secure data exchange, it does not rely solely on encryption from the Bluetooth module. Instead, the protocol uses self-identified headers and checksums to ensure the integrity and authenticity of the data exchanged.

Detailed attention is given to the authentication and identification procedures, which are critical components of the security protocol. These procedures ensure that both the mobile device and the accessory are verified and can communicate securely, with mechanisms in place to protect the integrity of the exchanged data. The identification process also provides the mobile device with essential information about the accessory, such as supported data types and protocols, preparing it for subsequent interactions as described in Chapter 4.

The final section addresses the methods for establishing a reconnection to manage signal loss and system resets. This part of the chapter outlines the strategies to ensure a stable and continuous connection, even in the face of potential disruptions. By covering these topics in detail, the chapter aims to provide a thorough understanding of the complexities and technicalities involved in establishing and maintaining a secured Bluetooth connection, focusing on data integrity and authentication.

The following diagram serves as a visual guide, outlining the step-by-step procedure for establishing a secure Bluetooth connection between the mobile device and the accessory.

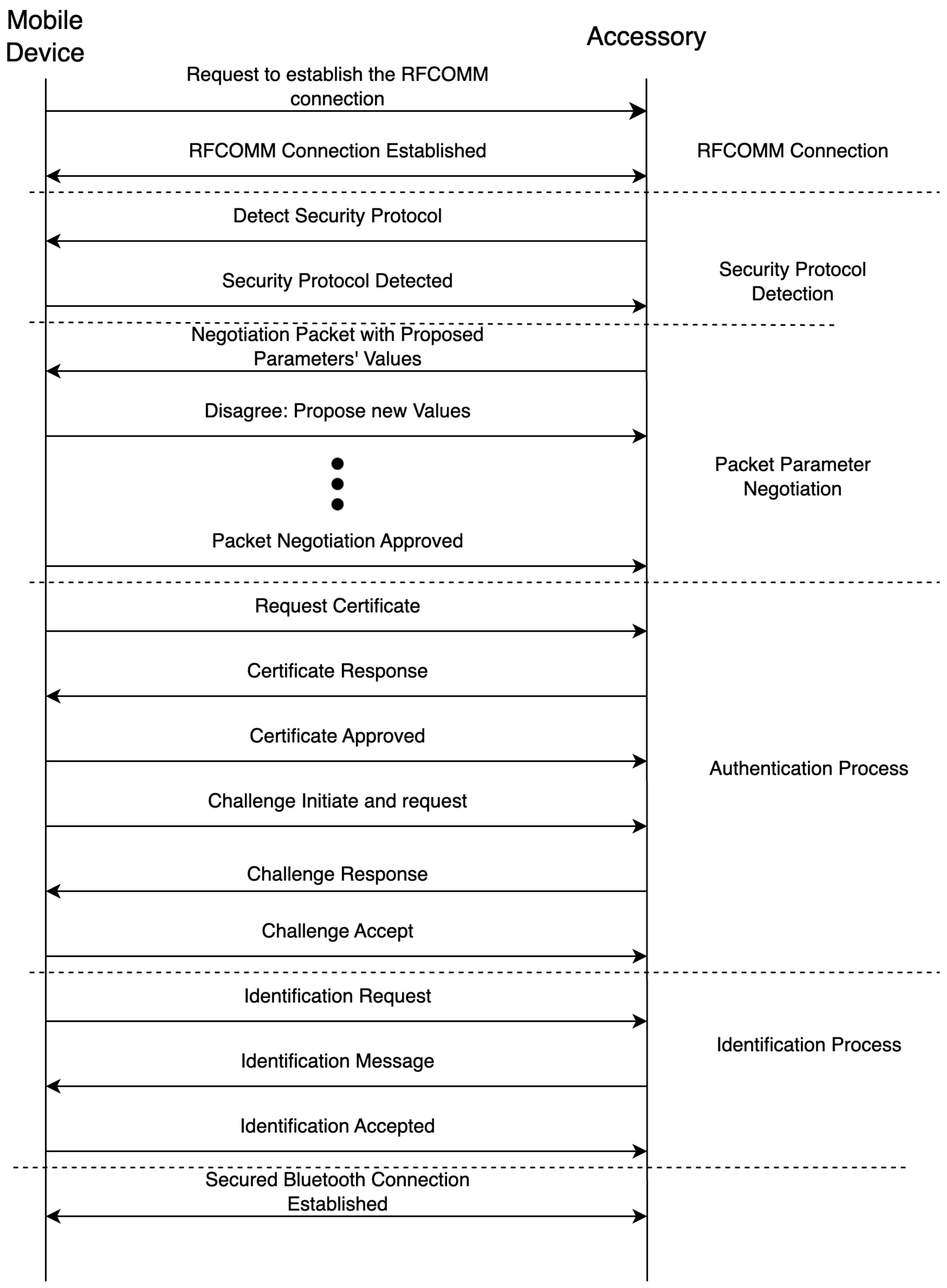

Mobile Device
Accessory
Request to establish the RFCOMM connection
RFCOMM Connection Established
RFCOMM Connection
Detect Security Protocol
Security Protocol Detected
Security Protocol Detection
Negotiation Packet with Proposed Parameters' Values
Disagree: Propose new Values
Packet Parameter Negotiation
Packet Negotiation Approved
Request Certificate
Certificate Response
Certificate Approved
Challenge Initiate and request
Authentication Process
Challenge Response
Challenge Accept
Identification Request
Identification Message
Identification Process
Identification Accepted
Secured Bluetooth Connection Established

Figure 3.1: Overview of the Secured Bluetooth Connection

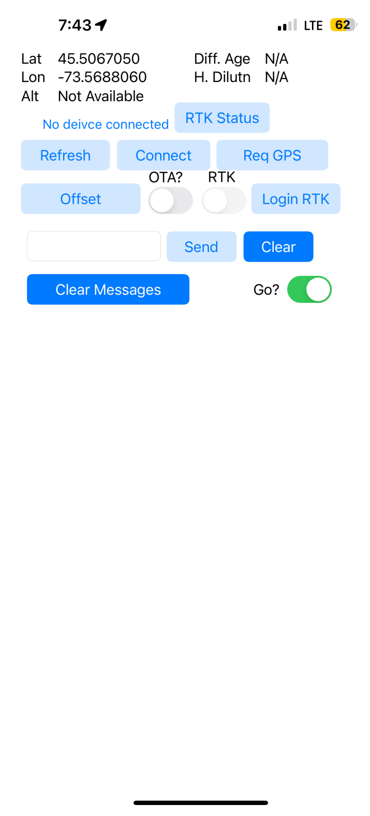
3.1 Module explanation and RFCOMM connections

The system is primarily composed of four key components: a mobile device, a security coprocessor, a Bluetooth module, and the STM32 microprocessor serving as the central processing unit. Each unit has a specific role and setup, detailed in the following sub-sections. The final subsection will indicate how to enable the accessory to open the RFCOMM port, allowing the mobile device to establish a Bluetooth connection.

3.1.1 Mobile device

Our system is designed to support iOS and Android operating system-based devices. Corresponding iOS and Android applications are being built to establish secured connections. The mobile device shares the same secret private key with the security coprocessor to perform the same irreversible hash-based message authentication code (HMAC) procedure based on the SHA256 hashing algorithm as the security coprocessor for authentication purposes.

The diagram below illustrates the appearance of our application. Users can search for desired accessories using the explore tab. If the device is not displayed, the user can press the ”refresh” button to check for new devices. Once the desired device is found, they should press the ”Connect” button to establish a secure connection. If the connection is successful, the ”Connect” button will change from blue to grey. If the connection fails, the user must attempt to establish a secure connection again, or the accessory may not support our security protocol.



Lat 45.5067050
Lon -73.5688060
Alt Not Available
Diff. Age N/A
H. Dilutn N/A
No deivce connected

Figure 3.2: Appearance of the Application

3.1.2 Security coprocessor

Multiple security coprocessors are available that can facilitate the authentication process through certificate and challenge/response exchange mechanisms. The following figure illustrates the structure utilized by the current system. Similar products performing the same authentication processes can also be considered for integration.

The security coprocessor connects to the microprocessor via the I2C interface, as shown in Figure 3.3. The authentication process comprises two key steps: the certificate exchange and the challenge/response exchange. The X.509v3 certificate is stored in the ROM of the coprocessor [23]. During the authentication process, a random 64-bit challenge is generated by the authenticating party and written into the RAM of the coprocessor.

The Logic Control Unit (LCU) reads the challenge from the RAM and processes it using the 256-bit secret key stored in the ROM and the HMAC-SHA256 core. The resulting response is then generated and stored back in the RAM [4].

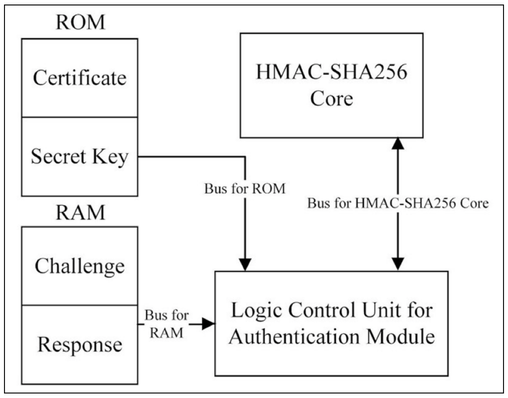


Figure 3.3: Security Coprocessor Structure [4]

3.1.3 Bluetooth module

The primary role of the Bluetooth module is to make our accessory discoverable and connectable, initiating communication between the mobile device and the accessory. We utilize a Bluetooth module developed by Silicon Labs, which is programmable using BGScript.

To set up the Bluetooth module for an RFCOMM connection, the following configurations are necessary:

- Upon system reset and initialization, assign a user-friendly local name to the Bluetooth module.
- Initiate an RFCOMM server with an associated SDP file to activate peripheral mode.
- Enable bonding mode to facilitate the acceptance of new pairings.

Once the RFCOMM connection is established, it is crucial to transmit a connection message to the microprocessor through UART. This message should include a flag indicating a successful connection and the current software module's version number. Subsequently, data received from the RFCOMM port should be directed to the microprocessor. In contrast, data from the microprocessor must be efficiently transmitted to the mobile device via Bluetooth, serving as a bridge for data transportation between the microprocessor and mobile devices.

3.1.4 Microprocessor

The microprocessor serves as the device's central processing unit. It requires support for at least two UART connections: one for the Bluetooth module with flow control enabled and the other for an external sensor, such as the GNSS receiver. Additionally, it should feature at least one I2C connection to facilitate communication with the security coprocessor. The interrupt receive function is utilized to receive messages from the Bluetooth module, while the DMA receive function is employed to obtain messages from the external sensor. The STM32 F and U series processors have undergone rigorous testing and are recommended for integration into the system. Therefore, processors with similar capabilities from other reputable companies, such as Raspberry Pi, may also be considered.

The diagram illustrates how the microprocessor functions as the central unit, facilitating connections with all components mentioned in the previous sections.

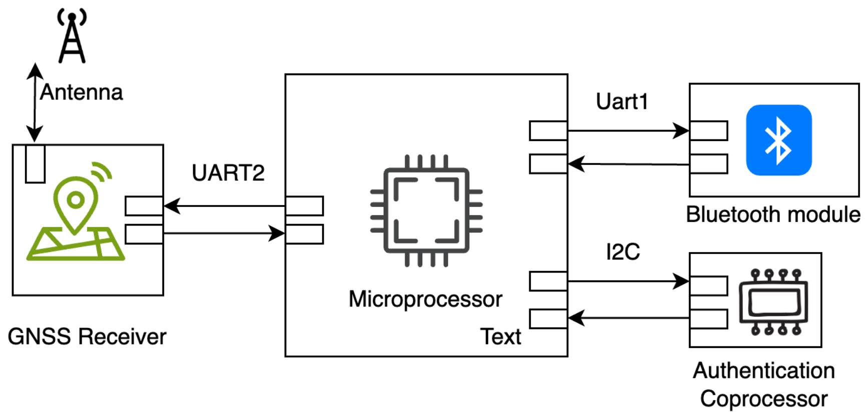


Figure 3.4: The Components of the Accessory

3.1.5 RFCOMM connection

The following diagram illustrates how to establish the RFCOMM connections.

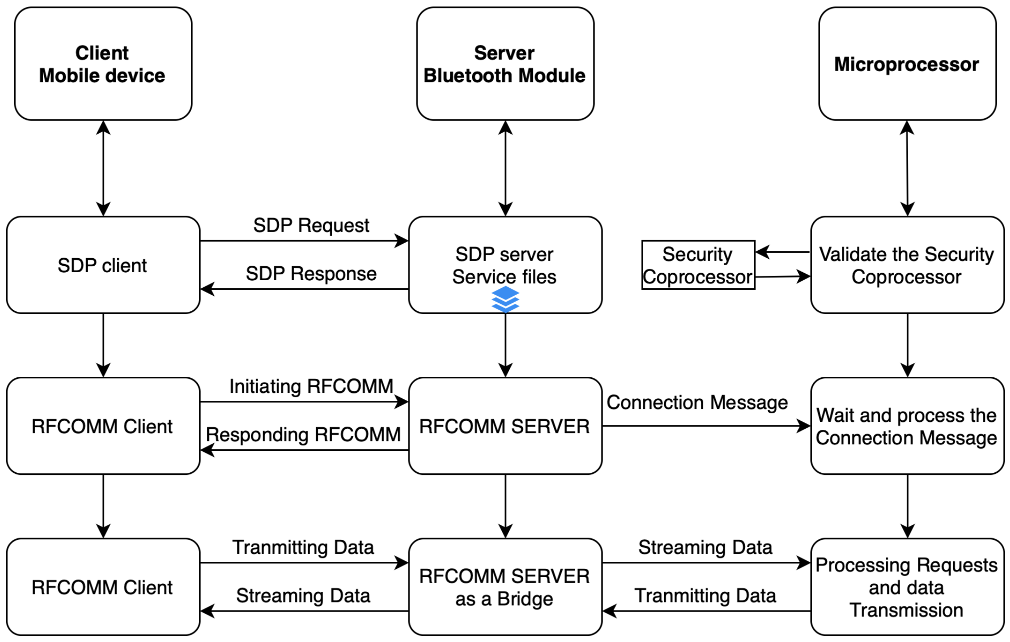


Figure 3.5: The The establishment of the RFCOMM connection

The Bluetooth module should only start after the microprocessor has been initialized and is running. The microprocessor will first check the value stored in the read-only Certificate length register of the security coprocessor. If the microprocessor is unable to retrieve the value, it means there is a connection issue with the coprocessor, and the program will be aborted. In such a case, the user must fix the connection problem and restart the program. If the security coprocessor returns a non-zero length value, the program will wait until the connection message is received, and the version number will be saved for later use. This way, the microprocessor will always be kept running.

At this stage, the mobile device and the accessory can start communicating with each other and exchange authentication messages to establish a secure connection.

3.2 Security Protocol

As illustrated in Figure 3.1, after establishing the RFCOMM connection, the security protocol comprises two main components:

1. **Authentication and Identification**: This component is essential for establishing a secure connection. Detailed explanations of the authentication and identification processes are provided in subsequent sections.

2. **Packet Encapsulation**: This component ensures data integrity and security using an adjusted header and checksum to encapsulate packets. The properties of the packet header aid in state management.

This section focuses on the packet encapsulation aspect of the security protocol. There are two types of packets utilized in this protocol:

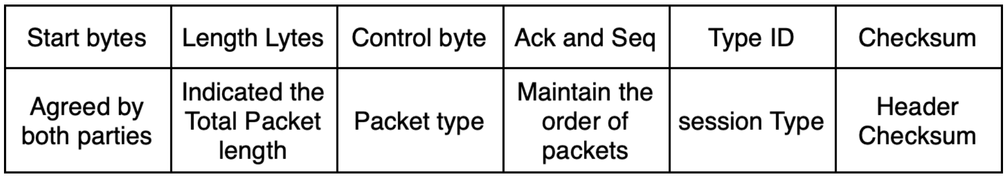
- **Command Packets**: These packets control operations and configurations for the system.

- **Data Packets**: These packets transmit messages between the application and the accessory.

State management within the security protocol employs a finite state machine to manage communication. Each state represents a stage in the process, controlled by command packets with unique identifiers. These packets guide the system through authentication, data transmission, or resetting connections, ensuring synchronization and efficiency.

3.2.1 Security Protocol Header

All packets transmitted between the mobile device and the accessory must begin with the security protocol header bytes. Figure 3.6 illustrates the structure of the header packet. The structure of a security protocol header begins with start bytes. Upon seeing these start bytes, the accessory and the mobile device recognize each other. If the start bytes are not detected, the packet is dropped. The start bytes are followed by bytes that define the total message length. The packet also includes bytes for acknowledgment and sequence numbers, which help ensure that packet transmission is always in order. If a packet is received out of order or is missing, the error can be easily detected.



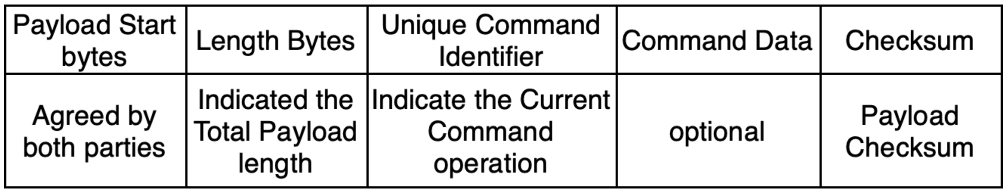
Start bytes	Length Lytes	Control byte	Ack and Seq	Type ID	Checksum
Agreed by both parties	Indicated the Total Packet length	Packet type	Maintain the order of packets	session Type	Header Checksum

Figure 3.6: Structure of the Security Protocol Header

The security protocol header, without any payload data, acts as the acknowledgement packet.

3.2.2 Command Packets

Command Packets include the security protocol header and the command payload. The security protocol header ensures the packet is recognized and properly formatted, while the command payload contains specific instructions or configurations for the system operations. The most crucial part of the command packet includes a unique identifier that specifies the current command. The command data part is optional; for instance, some commands, such as a certificate response, must include the certificate data in the command payload. Figure 3.7 shows the structure of the command packet payload.

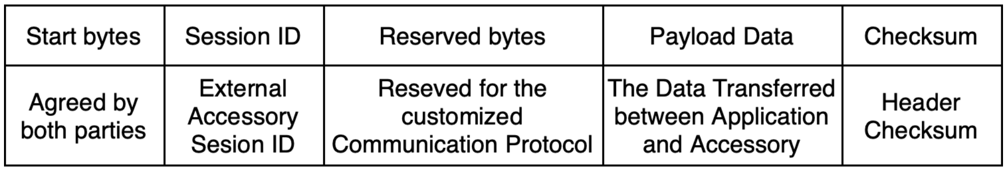


Payload Start bytes	Length Bytes	Unique Command Identifier	Command Data	Checksum
Agreed by both parties	Indicated the Total Payload length	Indicate the Current Command operation	optional	Payload Checksum

Figure 3.7: Structure of the Command Payload

3.2.3 Data Packet

Data Packets include the security protocol header and the data packet payload. They are exchanged only after the session has been established for the accessory (the session for External Accessory, detailed explanation can be found in Chapter 4, Section 1). The data packet is used to transmit data between the application and the accessory. In our program, it is the packet to exchange the data between the application and the sensor. Figure 3.8 illustrates the detailed structure of the data packet.



Start bytes	Session ID	Reserved bytes	Payload Data	Checksum
Agreed by both parties	External Accessory Sesion ID	Reseved for the customized Communication Protocol	The Data Transferred between Application and Accessory	Header Checksum

Figure 3.8: Structure of the Data Packet Payload

3.2.4 State Management

The security protocol manages communication between the mobile device and the accessory using a series of states structured as a finite state machine. Each state represents a specific stage in the communication process, ensuring synchronization and efficient operation handling.

State transitions are controlled by command packets, each containing a unique identifier that specifies the command. These commands guide the system through steps like initiating authentication, switching to data transmission, or resetting the connection. This ensures that operations are synchronized and handled efficiently.

Command packets include the security protocol header and a payload with specific instructions or configurations. The unique identifier in each packet specifies the command, guiding the system through different states.

Data packets are used during active data exchange states. Once the session is established, data packets carry the payload from sensors or other sources, ensuring accurate and efficient data transmission.

The following diagram illustrates the basic structure of our system's state management mechanism. The left side represents the processes of establishing a secure Bluetooth connection (Chapter 3), while the right represents the location application enhanced by RTK (Chapter 4). Each block may contain multiple sub-states, discussed in detail in the relevant chapters.

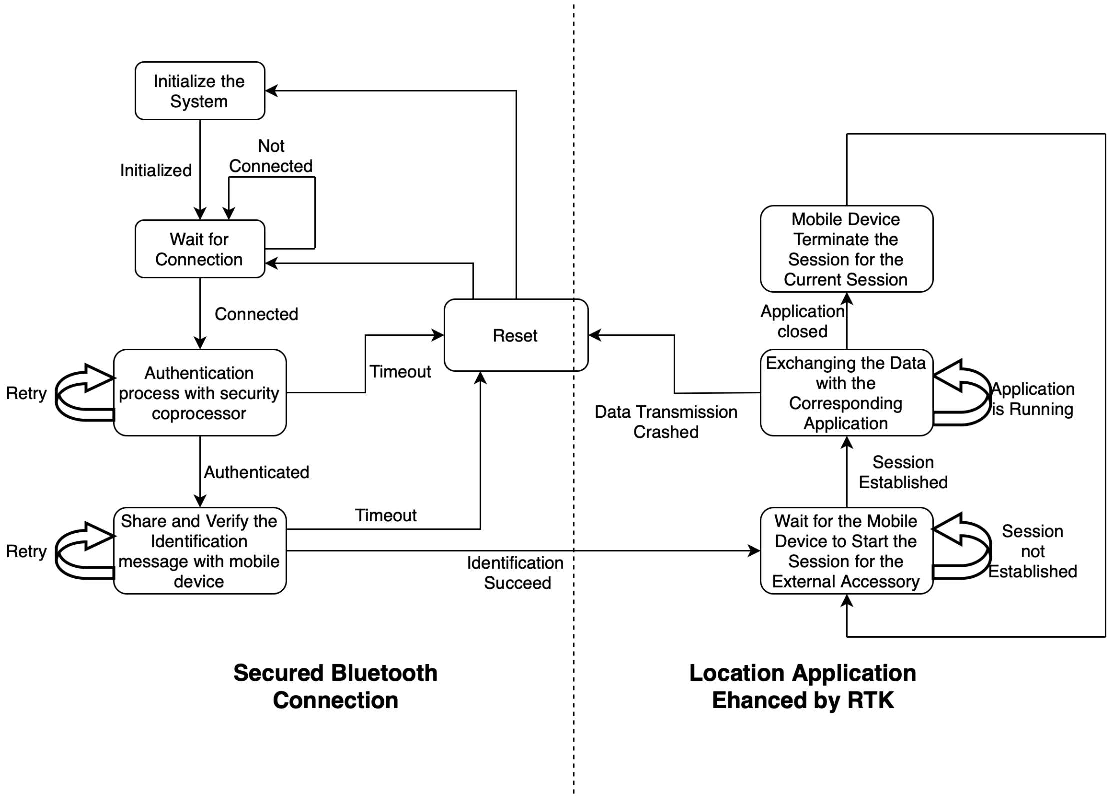

Initialize the System
Initialized
Not Connected
Wait for Connection
Connected
Reset
Retry
Authentication process with security coprocessor
Timeout
Authenticated
Retry
Share and Verify the Identification message with mobile device
Timeout
Identification Succeed
Data Transmission Crashed
Mobile Device Terminate the Session for the Current Session
Application closed
Exchanging the Data with the Corresponding Application
Application is Running
Session Established
Wait for the Mobile Device to Start the Session for the External Accessory
Session not Established
Secured Bluetooth Connection
Location Application Ehanced by RTK

Figure 3.9: State Management Mechanism of the System

By utilizing the security protocol, the system transitions through states using command packets to control the flow and data packets for information exchange. Each state has specific operations, and by completing these, the accessory and mobile device exchange state information and move to the next state. This ensures reliable and orderly communication, which is essential for maintaining a secure and efficient connection.

3.3 Security protocol detection

Once the Bluetooth connection is established, the accessory must query the mobile device to check whether it can support the security protocol to process the following authentication procedure.

The query message, called the security protocol check sequence, is a six-byte array. When the desired mobile device receives the sequence, it sends the exact six-byte check sequence message to the accessory, indicating its support for the security protocol. If the mobile device does not support the security protocol, the accessory and mobile device will only exchange plain text messages without any protection.

3.4 Security Protocol Preparation

3.4.1 Security protocol detection

Once the Bluetooth connection is established, the accessory must query the mobile device to check whether it can support the security protocol to process the following authentication procedure, as shown in Fig. 3.1.

The query message, called the security protocol check sequence, is a six-byte array. When the desired mobile device receives the sequence, it sends the exact six-byte check sequence message to the accessory, indicating its support for the security protocol. If the mobile device does not support the security protocol, the accessory and mobile device will only exchange plain text messages without any protection.

3.4.2 Packet Receiving and transmission

The program utilizes interrupt receive to gather data from the Bluetooth module byte by byte. The microprocessor assembles the bytes into a complete packet for further processing.

In the interrupt handler, since all the packets contain a security protocol header packet(see Fig. 3.6), the program scans for two consecutive start bytes to identify the beginning of the packet. Once the start bytes are located, the program continuously stores the third and fourth bytes to extract the packet length information. Upon obtaining the packet's length, the program receives the remaining bytes until the counter reaches the packet length value.

When transmitting the packet, the security protocol header should be sent first, followed by the payload data and checksum.

3.4.3 Packet parameter negotiation

The accessory uses a negotiation packet to negotiate crucial parameters with the mobile device, such as the maximum packet length and timeout value throughout the entire session.

The structure of the negotiation packet is as follows:

Once the connected mobile device receives the proposed negotiation packet from the accessory, the values of these proposed parameters will be checked. If the mobile device agrees, it will send the same link packet back to the accessory; otherwise, it will suggest different values of specific parameters and send the new link packet to the accessory. This process repeats until the mobile device and accessory agree on all these negotiable parameters, or the packet negotiation process fails, and the whole process terminates; the process is shown as Fig. 3.1.

3.5 Authentication

The authentication process involves two main phases: certificate and challenge-response. This section will provide an overview of the authentication process before digging into the specifics of each phase.

Given that the program is designed to receive messages using interrupts, a flag indicates the successful receipt of a complete packet. In the main function, when the flag is set to one, the program proceeds to a switch-case statement to verify if there is a corresponding label for the received packet. Hence, the microprocessor can switch to the matched state.

To identify the current state, two bytes, located in the 13th and 14th bytes of the packet, are reserved as the identifiers. The value of the identifiers must be agreed upon on mobile devices and accessories.

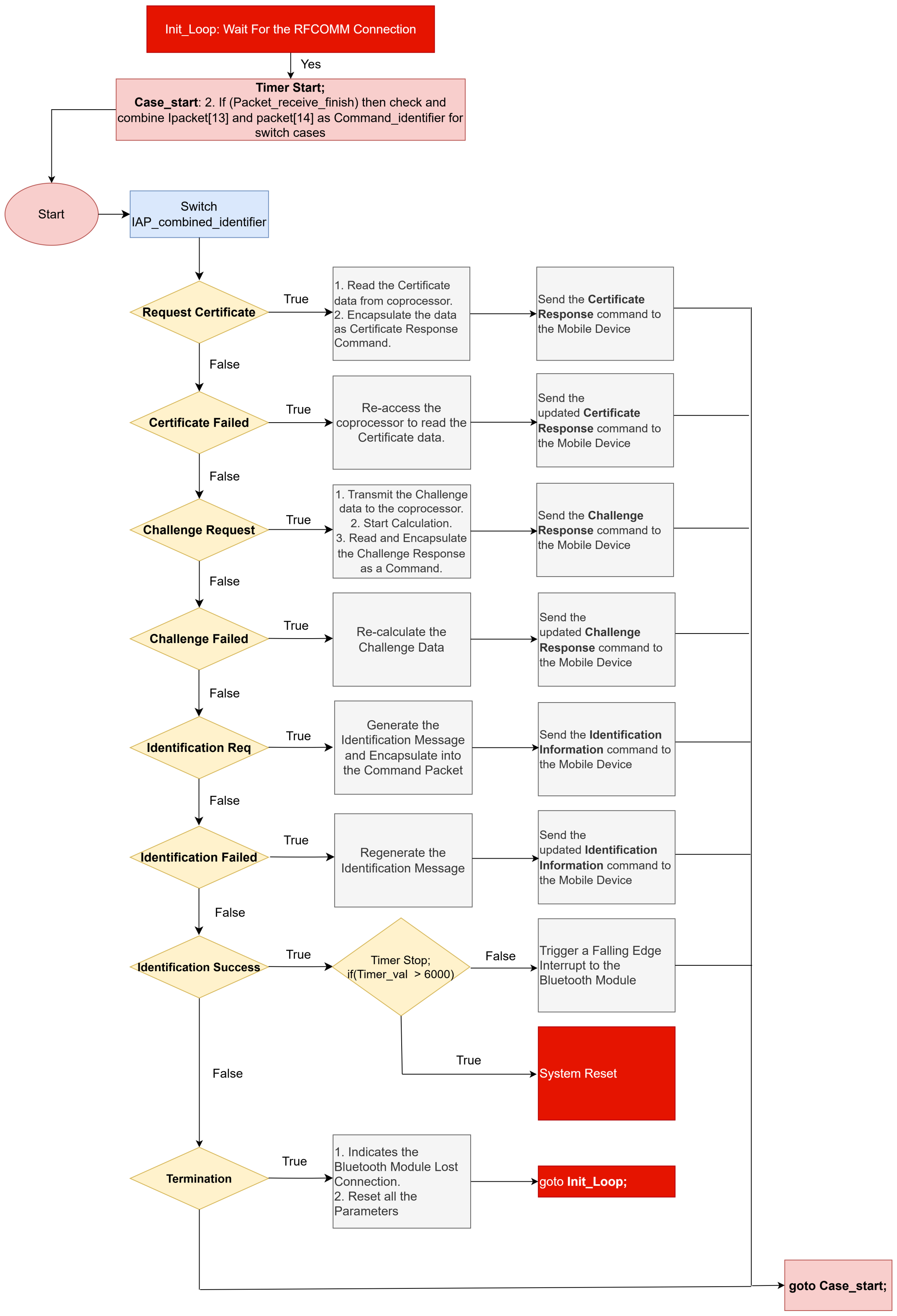

Init_Loop: Wait For the RFCOMM Connection
Yes
Timer Start;
Case_start: 2. If (Packet_receive_finish) then check and combine Ipacket[13] and packet[14] as Command_identifier for switch cases
Start
Switch IAP_combined_identifier
Request Certificate
True
1. Read the Certificate data from coprocessor.
2. Encapsulate the data as Certificate Response Command.
Send the Certificate Response command to the Mobile Device
False
Certificate Failed
True
Re-access the coprocessor to read the Certificate data.
Send the updated Certificate Response command to the Mobile Device
False
Challenge Request
True
1. Transmit the Challenge data to the coprocessor.
2. Start Calculation.
3. Read and Encapsulate the Challenge Response as a Command.
Send the Challenge Response command to the Mobile Device
False
Challenge Failed
True
Re-calculate the Challenge Data
Send the updated Challenge Response command to the Mobile Device
False
Identification Req
True
Generate the Identification Message and Encapsulate into the Command Packet
Send the Identification Information command to the Mobile Device
False
Identification Failed
True
Regenerate the Identification Message
Send the updated Identification Information command to the Mobile Device
False
Identification Success
True
Timer Stop; if(Timer_val > 6000)
False
Trigger a Falling Edge Interrupt to the Bluetooth Module
True
System Reset
False
Termination
True
1. Indicates the Bluetooth Module Lost Connection.
2. Reset all the Parameters
goto Init_Loop;
goto Case_start;

Figure 3.10: Detailed Structure of Authentication and Identification Process.

Figure 3.10 details the authentication procedure. For comprehensive understanding, the identification and error-handling procedures are also included and will be discussed in subsequent sections.

3.5.1 Certificate

The certificate information is securely stored in the security coprocessor's read-only memory, with a fixed length for the certificate data. This data is transmitted to the designated mobile device for thorough verification and validation. The authenticating device employs the certification path validation algorithm specified in RFC 5280 to ensure its validity.[22]. This algorithm mandates that a certificate must be signed by authorities of higher trust, which are then trusted by both parties.

The mobile device initiates the certificate process by sending a certificate request to the accessory. Upon receiving the request, the accessory's microprocessor fetches the certificate data length from the coprocessor to accurately determine the amount of data to be read. The microprocessor then efficiently reads the certificate data from its corresponding register, handling data more significant than 128 bytes by reading 128 bytes at a time until the total amount is reached.

Moving on to the next step, the accessory proactively sends the stored certificate data to the mobile device for verification. Upon verification, if the mobile device trusts the accessory, it sends a certificate success message promptly. In case of distrust, an authentication failed message is sent to the accessory, prompting it to resend the certificate data until the process is completed, fostering a secure and reliable connection. If the process cannot be completed, the program will stop at this phase, and the connection process needs to be aborted.

3.5.2 Challenge-Response

The mobile device initiates the challenge process, which generates random 64-bit long challenge data and identifies itself with the challenge initiate identifier. Upon receiving the challenge data, the accessory stores it and then sends it to the coprocessor along with its corresponding register address. The coprocessor then begins the calculation using the same secret key shared with the mobile device. The coprocessor and the mobile device calculate using HMAS-SHA256, which produces irreversible messages, thereby preventing ”man in the middle” attacks and safeguarding the secret key.

Once the coprocessor calculates the message, the microprocessor retrieves it and sends it back to the mobile device as the challenge response. The mobile device then compares the received challenge response with its calculation to verify if they are the same. If they match, an authentication success message is sent to the accessory, and it is ready to process the next request. An authentication failed message is sent in the event of a mismatch, prompting the program to redo the challenge process. If the accessory fails to provide the correct result, the program will be blocked at this phase, and the connection will eventually be aborted.

3.6 Identification

The identification process can only proceed after the accessory has successfully completed the Accessory Authentication, marking the end of the security coprocessor's role. This process involves three commands: Request Identification, Identification Message, and Identification Rejected. Each command has a unique identifier, with packet structures referenced in Figures 3.6 and 3.7.

The Identification Message is the command containing a command payload. This payload informs the mobile device of details about the microprocessor, such as the firmware version and accessory name. It also specifies the communication protocols supported by the accessory, ensuring the mobile device can establish a session and communicate effectively with the accessory. The structure of the Identification Message is illustrated in the following figure:

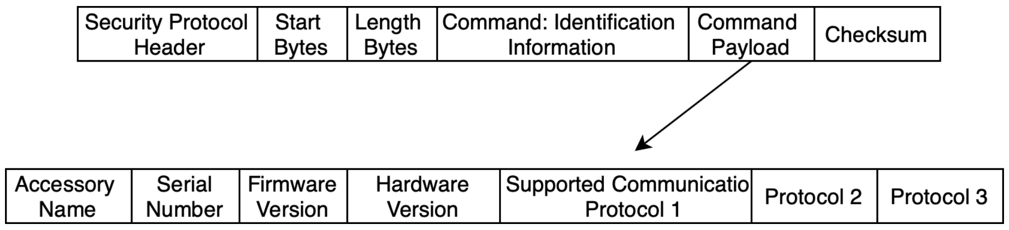


Figure 3.11: Structure of the Identification Message

The Identification Message can include many more fields than indicated in the figure. However, our program only considers the essential fields necessary for establishing communication.

The mobile device sends an Identification Rejected command if the Identification Message is rejected. The accessory must then modify and resend the Identification Message, as shown in Fig. 3.10. Continuous rejection of the message will result in the termination of the current connection.

3.7 Reconnection for Error Handling

There are two situations where reconnection is necessary. The first scenario occurs when the Bluetooth connection between the mobile device and the accessory is disconnected and requires a new connection to be established. The second scenario arises when the accessory cannot receive incoming data from the mobile device. The upcoming subsections will provide detailed explanations of these two issues and their corresponding solutions.

3.7.1 Bluetooth module Lost Connection

When the user turns off the mobile device's Bluetooth or the mobile device moves out of the Bluetooth range of the accessory, the Bluetooth module will lose its connection. Upon disconnection, the Bluetooth module triggers an interrupt for the closing event. This event sends a Terminate Connection command to the microprocessor, as shown in Fig. 3.10. The microprocessor then:

- Resets all parameters.

- Restarts from the beginning of the main function.

Simultaneously, the Bluetooth module starts two timers:

- The first timer restarts the RFCOMM server to accept new connections.
- The second timer attempts to reopen the RFCOMM channel with the last paired mobile device.

These timers ensure the previous connection is entirely disconnected and prevent command conflicts.

When the mobile device re-enters the Bluetooth range, and the Bluetooth setting is enabled, the accessory automatically attempts to reconnect. This process ensures a seamless and uninterrupted connection, similar to how AirPods reconnect when the user approaches the paired accessory.

After establishing the initial connection with a mobile device, the Bluetooth module stores the device's MAC(Media Access Control) address in flash memory. This process allows the Bluetooth module to:

- Retain the last connected device's information, even if the accessory is reset by being powered off and on.
- Act as a central module to attempt to reconnect with the stored MAC address when within range. If the device is within range, it will reconnect. If not, the Bluetooth

module will open the RFCOMM server to act as a slave and wait for a new connection, as controlled by the two timers.

3.7.2 Microprocessor Crash Detection

Maintaining a reliable connection in Bluetooth-enabled systems is crucial for ensuring seamless data transmission and user experience. During the secure authentication and identification stage, the microprocessor can crash, leading to potential disruptions. This subsection details the mechanisms to detect such crashes and ensure timely reconnection.

Timers are employed on both the microprocessor and Bluetooth module sides to detect failures during the secure authentication and identification stage.

Microprocessor Timer: A timer starts when the RFCOMM connection is established and ends when the identification process is finished. On average, the secured connection should be established within 3 seconds after establishing the RFCOMM connection. To account for variations, a timer is set for 6 seconds. If this timer exceeds 6 seconds, it indicates that the secured Bluetooth connection has crashed, data transmission is stuck, and the microprocessor is not responding. Consequently, the entire system needs to be reset, as shown in Fig. 3.10.

Bluetooth Module Timer: Similarly, the Bluetooth module sets a timer of 8 seconds, starting when the RFCOMM connection is established. If this timer reaches 8 seconds, the Bluetooth module will reset the connection.

If the microprocessor completes the authentication and identification process within 6 seconds, it triggers an interrupt to the Bluetooth module. Upon receiving this interrupt, the Bluetooth module disables its 8-second timer and continues the process. This dual-timer mechanism helps detect microprocessor failures and ensures that the system can attempt to reconnect and restart the program if a crash occurs.

The error handling mechanism is designed to reset and recover the system efficiently. When a crash is detected:

- The microprocessor resets all parameters and restarts from the beginning of the main function.
- The Bluetooth module attempts to reopen the RFCOMM channel with the last paired mobile device, ensuring minimal disruption to the user.

Chapter 4

Location Application Enhanced by RTK

Once the secure Bluetooth connection is established, the system can enable real-time data communication between the accessory and the mobile device, allowing for the introduction of the Location application.

Real-time kinematic (RTK) technology offers highly accurate location-based services with centimetre-level precision, making it an ideal choice for various applications. This chapter is focused on developing a location application enhanced by RTK technology. It will begin by exploring the protocol setup and session establishment necessary to establish the connection between the application and the accessory.

The chapter will then focus on the format of the National Marine Electronics Association

(NMEA) message received from the GNSS receiver and how to parse and process these messages for the microprocessor. It will then discuss the utilization of RTK technology to correct the current location.

Moreover, we'll introduce a practical function for the application, the 'location shift.' This feature significantly enhances the application's versatility, offering a wide range of potential use cases in various scenarios.

4.1 Establishing the Connection between the Application and the Accessory

4.1.1 Overview of the Communication Protocols

After successfully authenticating the Bluetooth connection between the mobile device and the accessory, it is crucial to establish a communication protocol for transmitting data from the accessory to the compatible application. This system defines two distinct communication protocols and allows for the simultaneous establishment of these two communication protocols.

- **Read-only Communication Protocol**: This communication protocol enables the mobile device to receive data from the accessory exclusively. Any data sent from the mobile device to the accessory will be disregarded. This protocol is commonly utilized to monitor the accessory. It is identified by the name ”com.bluegiga.read,” and the protocol name can be customized.

- **Read-and-Write Communication Protocol**: This protocol allows the application to receive data from the accessory and send data to the accessory. In this protocol, every message exchanged between the accessory and the mobile device application must have an additional four bytes at the beginning of the payload. As shown in Fig. 3.8, these four bytes are reserved in the data packet for the current communication protocol. The first two bytes are always 0xAB and 0xCD, serving as the unique current communication protocol identifier. The third byte indicates the length of the payload, excluding these four bytes. The fourth byte is reserved for the function identifier to distinguish between commands. The protocol is named ”com.bluegiga.readnwrite”.

0xAB	0xCD	0x60	0x10
Identifier for the read-and-write protocol		Length of the payload	Reserved byte as function identifier

Figure 4.1: Extra Header bytes for Read-and-Write Communication Protocol

It is important to declare all communication protocols in the identification message to ensure the accessory can support them. The identification message also assigns an index to each protocol, providing a convenient way for the application to utilize the desired protocol.

4.1.2 Session Management

Session management involves initiating, maintaining, and terminating communication sessions between a mobile device and an accessory. This process ensures that multiple instances of the same communication protocol can be handled concurrently without data loss or interference. This is crucial for applications requiring continuous data streams, such as GNSS data transmission.

To manage communication sessions, two primary commands are used:

- **Start Accessory Session**: Initiates a communication session for the accessory.
- **Stop Accessory Session**: Terminates a communication session for the accessory.

Each command has a unique identifier and follows the packet structure referenced in Figure 3.7. The command payload must contain:

- A unique identifier to indicate the message type.
- The communication protocol index that the application intends to use.
- The current two-byte session ID.

Whenever the mobile device application needs to open a connection with the accessory, it must establish a new communication session using a unique two-byte session ID. This session ID remains associated with the specific connection until it is terminated. The following figure illustrates the process of session establishment:

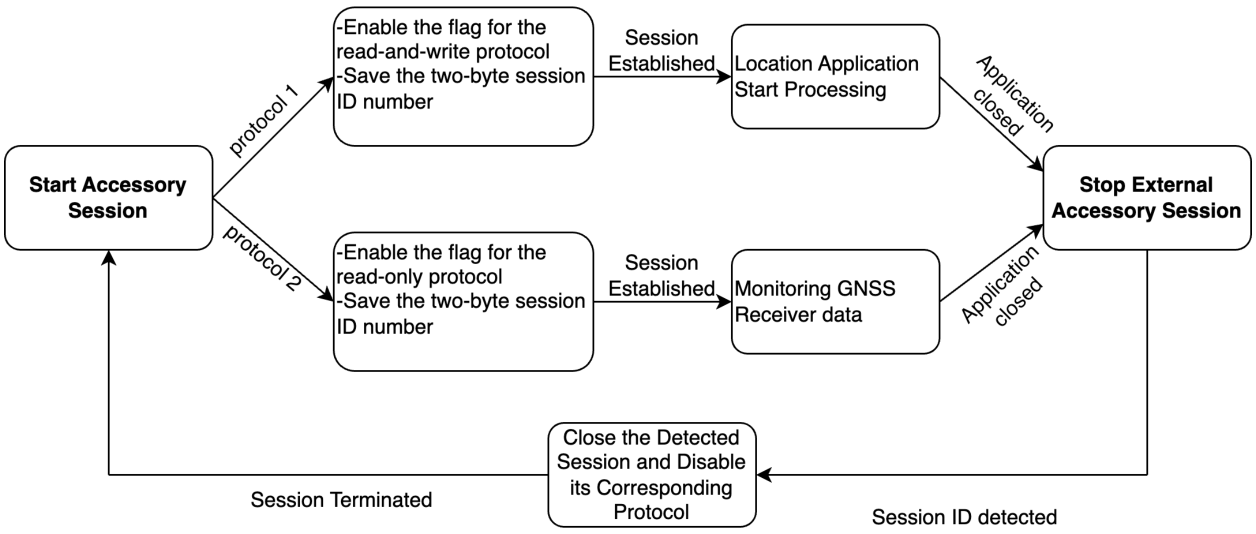

Start Accessory Session
protocol 1
protocol 2
-Enable the flag for the read-and-write protocol -Save the two-byte session ID number
-Enable the flag for the read-only protocol -Save the two-byte session ID number
Session Established
Session Established
Location Application Start Processing
Monitoring GNSS Receiver data
Application closed
Application closed
Stop External Accessory Session
Session ID detected
Close the Detected Session and Disable its Corresponding Protocol
Session Terminated

Figure 4.2: Session Establishment for the Accessory

Once the session is properly opened, the accessory can begin communicating with the application. Data packets can start to transmit. The two-byte session ID needs to be set in the data packet payload, as shown in Figure 3.8, allowing the microprocessor to identify the session to which each message belongs.

Since the session is linked to a specific protocol during the initiation stage, the microprocessor processes the message using the appropriate protocol. The processed message is then returned to the application, maintaining the same session ID in the header to ensure it is correctly delivered to its corresponding application. This systematic approach ensures efficient and accurate communication between the accessory and the mobile application.

4.2 Processing the GNSS Receiver Message on the Microprocessor

4.2.1 GNSS Receiver Integration

The GNSS receiver has two main components: the antenna and the processing unit. In our system, the processing unit connects to the microprocessor via UART, as shown in Fig. 3.4. Depending on its configuration, the GNSS receiver continuously outputs multiple NMEA(National Marine Electronics Association) sentences every second. The NMEA is a standard protocol used in marine electronics and GPS systems to communicate between devices like GPS receivers and navigation instruments.

For our application, we specifically require the latitude, longitude, and altitude data, as well as the age of the correction data, to monitor the RTK correction age. Therefore, the GNSS receiver should at least output the sentences "$GPGGA" and "$GPRMC" for position information. These are two sentences within the NMEA standard that provide the essential GPS fix data and basic GPS data separately. While other NMEA sentences, such as "$GPGSV", can also be transmitted, they are used solely for display purposes within the application and not for obtaining position data.

4.2.2 Receiving and Filtering the NMEA Sentences

The microprocessor uses GPDMA (General Purpose Direct Memory Access) to receive data from the GNSS receiver. This is crucial because the GNSS receiver can send hundreds of bytes per second. Relying on polling or interrupt-based methods would heavily tax the microprocessor's CPU, reducing efficiency. DMA helps alleviate this burden, allowing the program to run more efficiently even on less powerful microprocessors.

In our implementation, we use the GPDMA in circular mode. The following flowchart illustrates the receiving and filtering process.

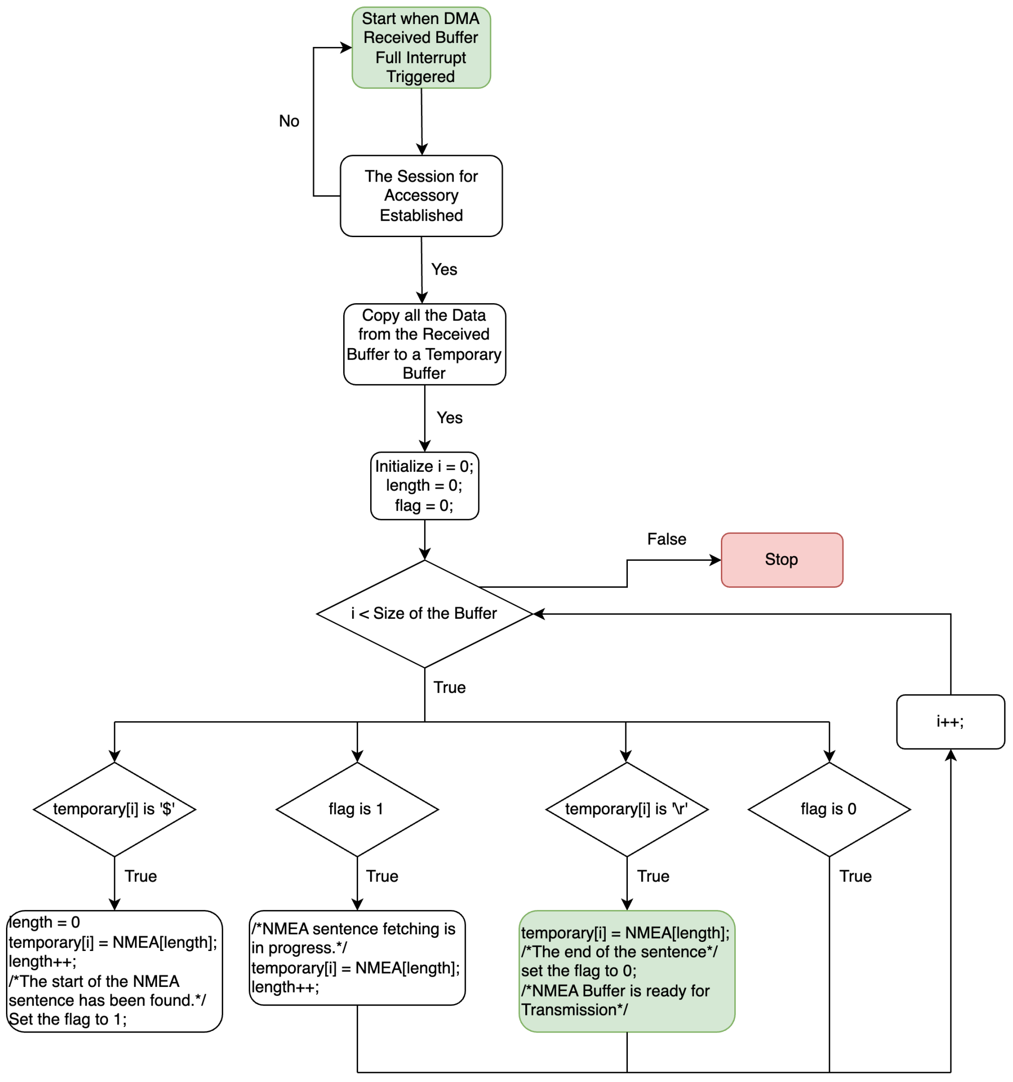

Start when DMA Received Buffer Full Interrupt Triggered
No
The Session for Accessory Established
Yes
Copy all the Data from the Received Buffer to a Temporary Buffer
Yes
Initialize i = 0; length = 0; flag = 0;
False
Stop
i < Size of the Buffer
True
i++;
temporary[i] is '$'
flag is 1
temporary[i] is '\r'
flag is 0
True
True
True
True
length = 0 temporary[i] = NMEA[length]; length++; /*The start of the NMEA sentence has been found.*/ Set the flag to 1;
/*NMEA sentence fetching is in progress.*/ temporary[i] = NMEA[length]; length++;
temporary[i] = NMEA[length]; /*The end of the sentence*/ set the flag to 0; /*NMEA Buffer is ready for Transmission*/

Figure 4.3: Filtering the NMEA message from the GNSS receiver

Once the DMA receive buffer is full, an interrupt is triggered to process the received data. The message is copied to a temporary buffer, where the program searches for the ”\$” character, indicating the start of an NMEA sentence. From this point, characters are stored in the NMEA sentence buffer until the ”\r” character is encountered, marking the end of the sentence. The length of each sentence is also recorded.

The message is now set for further processing or transmission to the application and is located in the green statement box at the bottom of the figure above. For each prepared message, if the characters ”GA” or ”MC” are found at the fourth and fifth byte in the message, indicating ”$GPGGA” or ”$GPRMC” sentence, additional configuration may be necessary based on the application’s request. The following section will provide a detailed discussion of this situation.

4.2.3 Transmitting the Filtered Message

The accessory transmits NMEA sentences in two primary scenarios:

1. **Active Accessory Session:**

    - When the accessory session is open with the application, the accessory transmits NMEA sentences using the designated communication protocol.
    - **Read-Only Protocol:** Every incoming NMEA sentence is encapsulated into the ”payload data” field of the data packet. The ”Reserved Bytes” field is set to empty, as shown in Figure 3.8. The accessory then sends out the resulting data packet.
    - **Read-and-Write Protocol:** In this protocol, the ”Reserved Bytes” field of the data packet payload is set to a four-byte sequence {0xab, 0xcd, length, 0x10} (see

Figure 4.1). The byte 0x10 indicates a standard NMEA sentence transmission. Once an NMEA sentence is received from the sensor, the microprocessor sends out the resulting data packet.

2. **User-Requested GPS Data:**

    - The following actions do not affect the data transmission in the first scenario.
    - **Request Location:** By default, the application displays the system location. When the user presses the "Req GPS" button (see Figure 3.2), the application sends a command to the accessory to request the accessory's location.
    - **Location Information:** Upon receiving this command, the accessory searches for either the GPGGA or GPRMC sentence. Once found, it transmits the Location Information command with a unique identifier to indicate that it contains the position data. The command data is the GPGGA or the GPRMC sentence. This allows the mobile device to update the location displayed in the application to the accessory's location instead of the system location. This action does not interrupt the standard NMEA transmission.
    - **Terminate Location:** To stop sending these location information commands, the accessory waits for the mobile device to send the Terminate Location command.

Fig. 4.4 illustrates the User-Requested GPS process together with an active session:

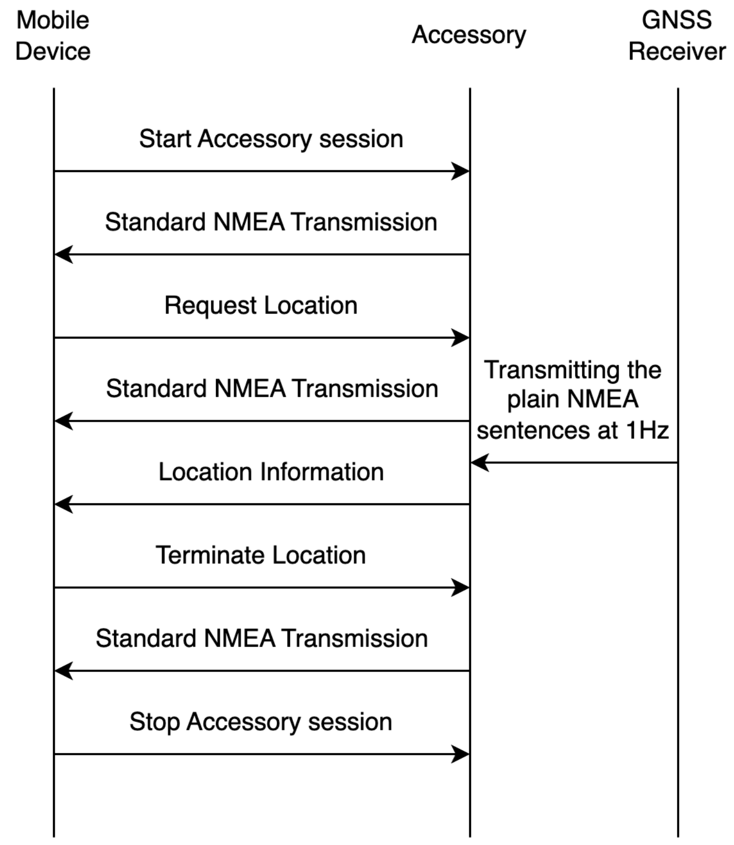


Figure 4.4: User-Requested GPS Process under an active session

4.3 RTK-Based Corrections for Improved GNSS Accuracy

In this section, we discuss the application of RTK-based corrections to enhance GNSS accuracy using the GPGGA and GPRMC NMEA sentences, which pertain to Global Positioning System (GPS) data. Consequently, the technology is referred to as GPS-RTK. We utilized Network-based RTK technology, specifically the NTRIP protocol and RTCM

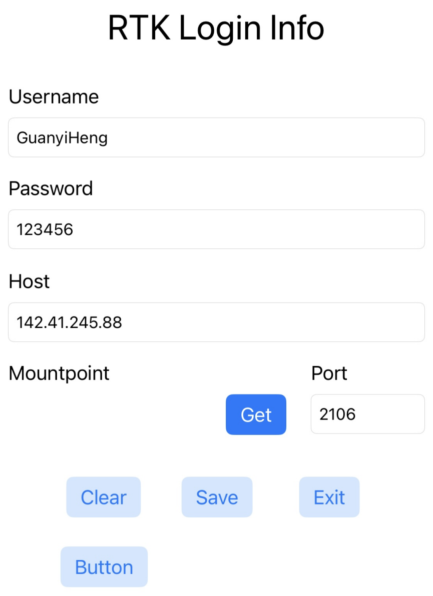
3.2 message format. To apply RTK corrections to the current location, the following are required:

1. Access to an RTK base station within 10 km.
2. A GNSS receiver that supports RTK correction features.
3. The ability to route the RTK correction message to the GNSS receiver, allowing it to process the RTK data and calculate the corrected position.

4.3.1 Receiving RTK Correction Data

The left part of Fig. 4.6 indicates the process of receiving the RTK data from the server. The NTRIP (Networked Transport of RTCM via Internet Protocol) server collects GNSS data from multiple reference stations. This data is then transmitted to an NTRIP caster, which acts as a central hub. The caster manages the data stream and makes it available to NTRIP clients, such as the rover GNSS receivers.

The mobile device's application interface and accessory act as an NTRIP client, allowing users to initiate RTK corrections. A new window appears upon clicking the "Login RTK" button, as shown in Figure 4.5. Users can enter their credentials and the server information, including IP address, mount point, and port. This enables the application to connect to the NTRIP caster. The correction data, in RTCM 3.2 format, is received at approximately 500 bytes per second. By toggling the "RTK" button to "on," the application sends a command to the GNSS receiver to activate RTK mode. This command is encapsulated in a data packet with reserved bytes {0xAB, 0xCD, length, 0x20}, where 0x20 indicates the RTK mode, notifying the accessory to process the RTK request. The RTK correction data is then encapsulated as payload data within packets and transmitted to the client.


RTK Login Info

Username: GuanyiHeng

Password: 123456

Host: 142.41.245.88

Mountpoint

Port: 2106

Figure 4.5: RTK Login Window

4.3.2 Handling RTK Data in the Accessory

Upon receiving correction data packets, the microprocessor begins processing the RTK data, as illustrated in the right part of Fig. 4.6. It first checks whether the packet is a data packet. If it is the data packet, the microprocessor will check whether the function byte is 0x20. If it is, it will start to process the data packet as the RTK-related message. Then, the microprocessor must remove the security protocol header, checksum, and parameters, isolating the payload data. This data is then routed to the GNSS receiver through UART, which directs the command and the correction data to the GNSS receiver. It allows the GNSS receiver to activate RTK mode, perform the necessary calculations as detailed in Chapter 2, and output the updated corrected NMEA sentences.

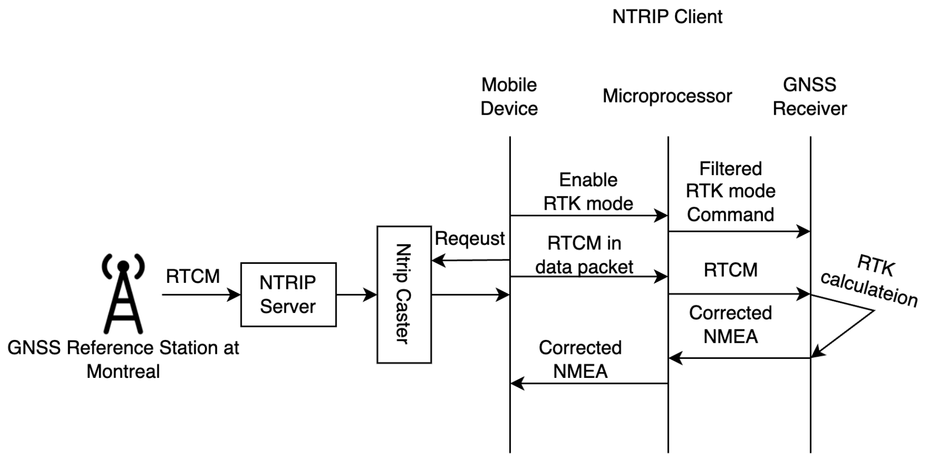


Figure 4.6: RTK Correction Structure

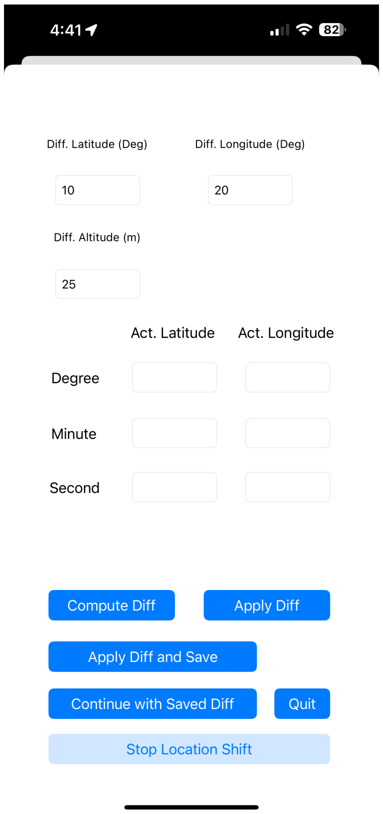
4.4 Location shift

The "location shift" feature empowers users to apply an offset to latitude, longitude, and altitude within the application. The application then transmits the calculated offset as a message. Upon receiving the offset, the accessory's microprocessor applies it to the current position message (GPGGA and GPRMC sentences) and executes the necessary calculations to generate the new shifted position message. Subsequently, the application can be updated to display the adjusted location.

In the following sections, we will focus on applying the location shift on the application side and efficiently managing the requests on the accessory side. Additionally, we will explore this application's potential use case.

4.4.1 Location Shift in the Application

In the application (see Figure 3.2), pressing the "Offset" button opens a window where users can enter offset values for latitude, longitude, and altitude, as shown in Figure 4.7.



4:41

Diff. Latitude (Deg): 10
Diff. Longitude (Deg): 20
Diff. Altitude (m): 25

	Act. Latitude	Act. Longitude
Degree		
Minute		
Second		

Figure 4.7: Appearance of the Offset Feature in the Application

After inputting the offset values, pressing the ”Compute Diff” button calculates the current offset value. The user can then choose from the following functions:

Apply Diff: This function encapsulates the calculated offset values into the ”payload data” part of the data packet, with the reserved bytes set to 0xAB, 0xCD, length, 0xB4. The byte 0xB4 indicates that the offset values apply to the GPGGA and GPRMC sentences. The packet is then sent to the accessory, requesting it to apply the offset. These offset elements are separated by commas to facilitate microprocessor parsing.

Apply Diff and Save: Similar to ”Apply Diff,” this function encapsulates the offset values into the data packet with the reserved bytes field set to 0xAB, 0xCD, length, 0xB5. In addition to applying the offsets, this function instructs the microprocessor to save the offset values to flash memory for future use.

Continue with Saved Diff: This function does not require user input for offset values. It sends a data packet without payload data, with the reserved bytes field set to 0xAB, 0xCD, length, 0xB6. The byte 0xB6 indicates a request for the accessory to apply the offsets stored in flash memory. If the flash memory is empty, no offset is applied.

The following diagram illustrates the structure of the data packet used for these requests:

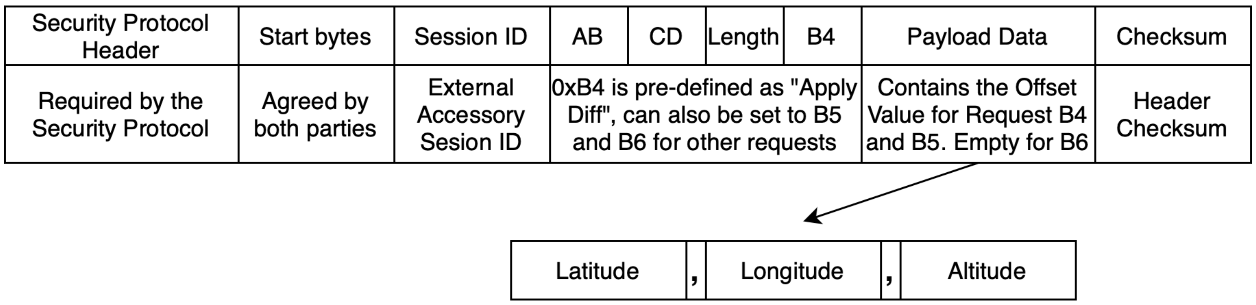

Security Protocol Header
Start bytes
Session ID
AB
CD
Length
B4
Payload Data
Checksum
Required by the Security Protocol
Agreed by both parties
External Accessory Sesion ID
0xB4 is pre-defined as "Apply Diff", can also be set to B5 and B6 for other requests
Contains the Offset Value for Request B4 and B5. Empty for B6
Header Checksum
Latitude
,
Longitude
,
Altitude

Figure 4.8: Structure of the Data Packet for Offset Requests

4.4.2 Handling Location Shift Requests on the Accessory

When the accessory receives a location shift request, it processes the request according to the specified parameters. The accessory identifies the request type using the identifier and performs the corresponding action, primarily applying location shifts to NMEA sentences. The following steps outline this process.

Processing Requests:

- The accessory uses the identifier from the request to determine the specific action needed, focusing on applying the location shifts.

Applying Location Shifts:

1. **Parsing Location Shifts:**

   The device effectively processes the GPGGA and GPRMC sentences by accurately splitting the fields using commas. It efficiently retrieves the payload data from the received packet received from the application. Following this, the payload data is separated into individual fields, allowing for the storage of the corresponding offset latitude, longitude, and altitude values.

2. **Adjusting Location Values:**

   Latitude: First, extract the original latitude value from the NMEA sentence. Then, adjust the latitude by adding the user-defined offset value. If the offset value is positive, it indicates a shift toward the North; if negative, it indicates a shift toward the South. Finally, format the adjusted latitude back into the NMEA sentence, making sure to include the correct hemisphere indicator (N or S).

   Longitude: The process for adjusting the longitude is analogous to that of latitude, with positive offset values denoting a shift toward the East and negative offset values indicating a shift toward the West.

   Altitude: Extract the original altitude value from the NMEA sentence and proceed to adjust the altitude by incorporating the user-defined offset value. Then, format the adjusted altitude back into the NMEA sentence.

3. **Transmitting Adjusted NMEA Sentences:** Applying the location shift will not affect the data transmission flow. The accessory continuously transmits the adjusted NMEA sentences, replacing the original data packets that contain the GPGGA and GPRMC messages with the adjusted location data.

Following these steps, the accessory can accurately apply location shifts to the NMEA sentences and transmit the adjusted data to the mobile application. This process ensures that the location data displayed in the application reflects the user-defined offsets, maintaining accurate and relevant position information. The concise pseudocode for the location shift algorithm is shown in Algorithm 1.

4.4.3 Potential Use Cases

The location shift function can be integrated with an external IMU sensor. This integration would be beneficial in scenarios where reliable satellite data is unavailable, such as indoor locations or tunnels. The IMU unit can calculate the movement and estimate the position of the accessory within indoor spaces. This movement data can then be applied to the location shift application to update the location, achieving dead reckoning in areas where GPS data is unavailable. Additionally, other sensors, such as laser distance detectors, can be utilized to enhance this function. Using a laser, users can determine the accurate location of an object pointed to by the laser. These applications can all be advanced in the future to improve the accuracy and reliability of location-based services.

4.5 Reconnection for Error Handling

During the data packet transmission stage, the GNSS receiver transmits NMEA sentences to the microprocessor at a frequency of one Hertz. Consequently, once the accessory is

Algorithm 1 Location Shift Algorithm

```
Input: originalLatitude, originalLongitude, originalAltitude, latitudeOffset,
    longitudeOffset, altitudeOffset
Output: adjustedLatitude, adjustedLongitude, adjustedAltitude
 1: function AdjustLocation(originalValue, offset, isLatitude)
 2:     Parse originalValue into degrees and minutes
 3:     Convert degrees and minutes to decimal format
 4:     Adjust value by adding offset
 5:     if isLatitude then
 6:         if offset is positive then
 7:             Indicate North
 8:         else
 9:             Indicate South
10:         Convert adjusted value back to degrees and minutes format
11:         Format adjusted value with correct hemisphere (N or S)
12:     else
13:         if offset is positive then
14:             Indicate East
15:         else
16:             Indicate West
17:         Convert adjusted value back to degrees and minutes format
18:         Format adjusted value with correct hemisphere (E or W)
19:     return adjusted value
20: function LocationShift(originalLatitude, originalLongitude, originalAltitude,
    latitudeOffset, longitudeOffset, altitudeOffset)
21:     adjustedLatitude = AdjustLocation(originalLatitude, latitudeOffset, true)
22:     adjustedLongitude = AdjustLocation(originalLongitude, longitudeOffset, false)
23:     adjustedAltitude = originalAltitude + altitudeOffset
24:     // Transmit adjusted NMEA sentences
25:     Encapsulate and transmit adjusted NMEA sentences
```

activated, data packet transmission must occur every second. An Independent Watchdog (IWDG) timer is implemented to ensure no delays in data packet transmission. The system could potentially crash if no data packet is transmitted for over one second, indicating a transmission failure.

The IWDG is a timer with a down counter, configured based on the microprocessor's clock settings. We have set the IWDG to reset the system every two seconds, allowing a grace period for data transmission. A system reset is automatically triggered if the IWDG counts down to zero without the program refreshing the timer. Thus, if a transmission fails and no data is transmitted within two seconds, the system will automatically reset.

Upon system reset, the microprocessor sends a falling edge interrupt to the Bluetooth module, notifying it of the system crash and the need for a reset. The Bluetooth module then closes the current RFCOMM port and attempts to reconnect with the mobile device to restart the entire process.

If the user closes the application, data transmission for the active accessory session will stop. The IWDG will be automatically refreshed in this scenario to prevent an undesired reset. Once the accessory session is reopened, the automatic refresh will be disabled, allowing the IWDG to function normally.

Chapter 5

Results and Discussion

This chapter presents the results obtained after executing the entire program, structured into three primary sections: the security protocol, RTK performance, and the location shift function. Initially, we examine the performance of the secured Bluetooth connection based on the implemented security protocol. The performance includes successfully establishing a secure connection and commencing data transmission from the GNSS receiver. A detailed time profiling is taken for each state. It includes processing times on the microprocessor and transmission delays. Additionally, we provide results on multi-application concurrent connections, compare secured connections with non-secured plain text transmissions, and assess the system's maximum throughput.

Subsequently, we discuss the RTK performance, focusing on its accuracy and reliability. The final section evaluates the location shift function by applying shift values and analyzing the resulting performance. Throughout these sections, we identify areas for improvement and discuss the current constraints of the implementation. This leads to a consideration of potential future enhancements and ongoing development needs.

The test kit used for the following tests includes a BT122 Bluetooth module from Silicon Labs and an STM32U585 microprocessor from STMicroelectronics. The security coprocessor for authentication is synthesized using SIMC 65nm technology, with details in [4]. The mobile device can be either an iOS or Android device. The GNSS receiver is the GNSS development kit from Hemisphere GNSS. The connections of each component are illustrated in Fig. 3.4.

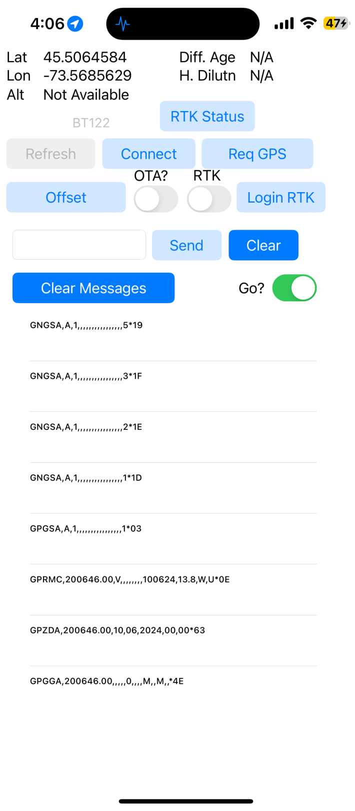
In the end, the future work subsection details the future directions based on the current application, such as showing the potential use cases.

5.1 Security protocol Performance

5.1.1 Single Application Experiment

This section presents the results of testing the system with an accessory and an iOS mobile device running the application (Fig. 3.2). The system successfully passed authentication and identification, establishing a secure connection between the application and the accessory. Fig. 5.1 shows the application after establishing the secure connection, displaying active GNSS data, indicating successful data transmission. Over a month of continuous testing confirmed stable and reliable data transmission, with standard NMEA sentences from the

GNSS receiver updating every second.



```
4:06
Lat   45.5064584          Diff. Age   N/A
Lon   -73.5685629         H. Dilutn   N/A
Alt   Not Available
BT122                     RTK Status
Refresh   Connect   Req GPS
          OTA?   RTK
Offset                Login RTK
          Send   Clear
Clear Messages            Go?

GNGSA,A,1,,,,,,,,,,,,,,,,5*19
GNGSA,A,1,,,,,,,,,,,,,,,,3*1F
GNGSA,A,1,,,,,,,,,,,,,,,,2*1E
GNGSA,A,1,,,,,,,,,,,,,,,,1*1D
GPGSA,A,1,,,,,,,,,,,,,,,,1*03
GPRMC,200646.00,V,,,,,,,100624,13.8,W,U*0E
GPZDA,200646.00,10,06,2024,00,00*63
GPGGA,200646.00,,,,,0,,,,M,,M,,*4E
```

Figure 5.1: Application after establishing a secure connection

As seen in Fig. 5.1, the GPGGA and GPRMC messages, representing position data, are empty due to poor sensor signal and the Request Location command not being sent. The displayed latitude and longitude are from the mobile device's system location. This result demonstrates the data transmission process and prepares for comparisons in subsequent sections.

The profiling process analyzing the time to complete each stage on the mobile device is shown in Fig. 5.2. "Accessory session" represents the time to establish a session with the accessory, and ”Location Service” represents the time to request location data. The time measured is from sending the request to receiving the corresponding response. Due to hardware limitations, the measured time fluctuates slightly for each trial; the following profiling results are the averages of 10 trials.

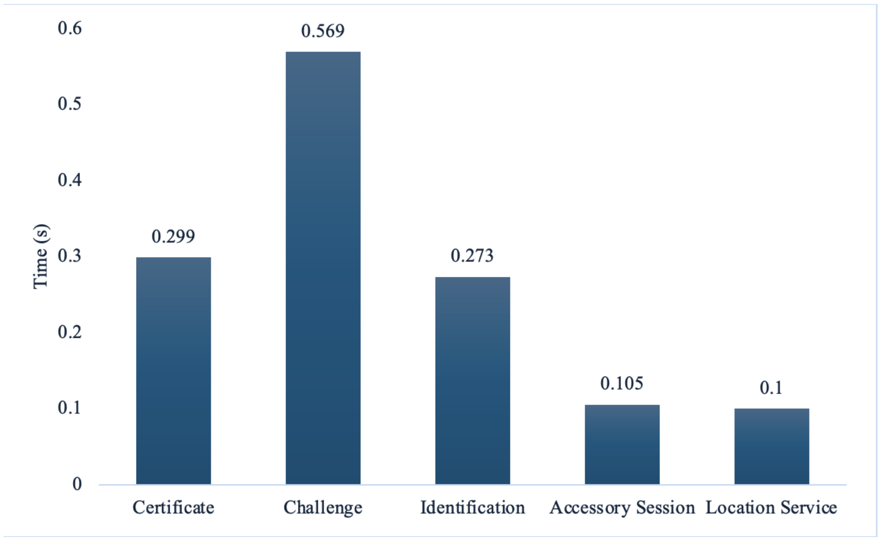


Figure 5.2: Average Time taken for each stage on the phone

For accessory performance profiling, the built-in cycle counter of the microprocessor was used, offering higher resolution and simplicity compared to traditional timers. Table 5.1 shows the cycle counts and corresponding times for each stage. The certificate and challenge processes take longer due to communication with the security coprocessor. Processing the challenge data using HMAC-SHA256 is relatively costly, significantly increasing the time required. The identification process triggers a Bluetooth module interrupt, slightly increasing its duration. Establishing an accessory session involves triggering a watchdog timer, while the location service only sets flags, making the former slightly longer.

Process	**Cycle Count**	**Time (seconds)**
Certificate	13,857,980	0.0866
Challenge	66,438,945	0.4152
Identification	439,733	0.00275
Accessory Session	155,416	0.00097
Location Service	125,800	0.00079

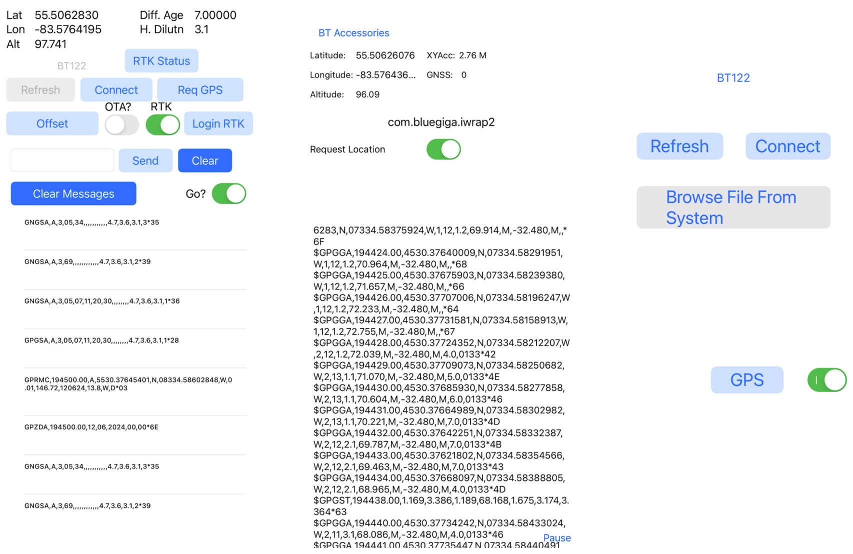
Table 5.1: Cycle counts and corresponding times for processing each stage on the accessory

Comparing Fig. 5.2 and Table 5.1, the time taken on the phone is significantly longer than on the microprocessor, especially for the Accessory Session and Location Service stages. This indicates the microprocessor's efficiency in processing requests, with most delays attributed to data transmission. For example, the Accessory Session involves sending a request from the mobile device and receiving confirmation from the accessory, with each transmission round taking around 0.012 seconds after subtracting the accessory's processing time at the baud rate of 115200bps. The transmission delay mainly comprises two parts: (1) the time consumed by the wired UART transmission from the microprocessor to the Bluetooth module and (2) the Bluetooth Classic transmission delay. Increasing the UART baud rate to four times 115200bps reduced the transmission delay marginally, as the amount of data to transmit is not demanding. However, increasing the baud rate introduced packet loss errors. Thus, increasing the baud rate did not significantly improve performance. Bluetooth Classic inherently has a latency of approximately 30 to 100 milliseconds, depending on the conditions such as distance and obstacles.

As a result, the delay caused by accessing the certificate and challenge information from the security coprocessor does not significantly affect the program. Since authentication is required only once at the beginning, there is no continuous data traffic. Given the current low data transmission requirements, the transmission delays during wired and wireless transmission are tolerable.

5.1.2 Multi-Application Concurrent Connections

Figure 5.3 demonstrates the simultaneous operation of three applications using two protocols, correctly directing messages to their corresponding applications. The left application is based on a read-and-write protocol, while the other two are read-only. These applications can receive data from the accessory, with the middle one displaying the data and the third one processing it in the background without display. This setup confirms the system can support multiple applications with different communication protocols, enabling concurrent connections and data transmissions.



```
Lat   55.5062830      Diff. Age  7.00000
Lon   -83.5764195     H. Dilutn  3.1
Alt   97.741

GNGSA,A,3,05,34,,,,,,,,,,,4.7,3.6,3.1,3*35
GNGSA,A,3,69,,,,,,,,,,,,4.7,3.6,3.1,2*39
GNGSA,A,3,05,07,11,20,30,,,,,,,,4.7,3.6,3.1,1*36
GPGSA,A,3,05,07,11,20,30,,,,,,,,4.7,3.6,3.1,1*28
GPRMC,194500.00,A,5530.37645401,N,08334.58602848,W,0
.01,146.72,120624,13.8,W,D*03
GPZDA,194500.00,12,06,2024,00,00*6E
GNGSA,A,3,05,34,,,,,,,,,,,4.7,3.6,3.1,3*35
GNGSA,A,3,69,,,,,,,,,,,,4.7,3.6,3.1,2*39
```

```
Latitude:   55.50626076   XYAcc:  2.76 M
Longitude:  -83.576436...  GNSS:  0
Altitude:   96.09

com.bluegiga.iwrap2
Request Location

6283,N,07334.58375924,W,1,12,1.2,69.914,M,-32.480,M,,*
6F
$GPGGA,194424.00,4530.37640009,N,07334.58291951,
W,1,12,1.2,70.964,M,-32.480,M,,*68
$GPGGA,194425.00,4530.37675903,N,07334.58239380,
W,1,12,1.2,71.657,M,-32.480,M,,*66
$GPGGA,194426.00,4530.37707006,N,07334.58196247,W
,1,12,1.2,72.233,M,-32.480,M,,*64
$GPGGA,194427.00,4530.37731581,N,07334.58158913,W,
1,12,1.2,72.755,M,-32.480,M,,*67
$GPGGA,194428.00,4530.37724352,N,07334.58212207,W
,2,12,1.2,72.039,M,-32.480,M,4.0,0133*42
$GPGGA,194429.00,4530.37709073,N,07334.58250682,
W,2,13,1.1,71.070,M,-32.480,M,5.0,0133*4E
$GPGGA,194430.00,4530.37685930,N,07334.58277858,
W,2,13,1.1,70.604,M,-32.480,M,6.0,0133*46
$GPGGA,194431.00,4530.37664989,N,07334.58302982,
W,2,13,1.1,70.221,M,-32.480,M,7.0,0133*4D
$GPGGA,194432.00,4530.37642251,N,07334.58332387,
W,2,12,2.1,69.787,M,-32.480,M,7.0,0133*4B
$GPGGA,194433.00,4530.37621802,N,07334.58354566,
W,2,12,2.1,69.463,M,-32.480,M,7.0,0133*43
$GPGGA,194434.00,4530.37668097,N,07334.58388805,
W,2,12,2.1,68.965,M,-32.480,M,4.0,0133*4D
$GPGST,194438.00,1.169,3.386,1.189,68.168,1.675,3.174,3.
364*63
$GPGGA,194440.00,4530.37734242,N,07334.58433024,
W,2,11,3.1,68.086,M,-32.480,M,4.0,0133*46
$GPGGA,194441.00,4530.37735447,N,07334.58440491
```

Figure 5.3: Multiple applications running concurrently on two protocols

For performance evaluation, the accessory transmitted up to 68 packets containing NMEA sentences per second, each ranging from 50 to 90 bytes, resulting in a maximum throughput of 6152. Compared to the theoretical maximum of 14400 bytes per second achievable through wired UART transmission at 115200 bps, the actual throughput is approximately 42.72% of the maximum potential. This discrepancy can be attributed to two main factors: 1. **Baud Rate Limitation**: As data transmission volume increases, the current baud rate becomes a limiting factor. Increasing the baud rate could enhance throughput but also introduce more errors and inefficiencies. 2. **Transmission Protocol Overhead**: To maintain data integrity and sequence, each packet transmission waits for an acknowledgment before sending the next packet, exacerbating the effects of Bluetooth Classic transmission delay.

Further testing aimed at maximizing throughput involves increasing the baud rate.

Doubling the baud rate increased throughput, but the rate of actual data transmission efficiency decreased, with more frequent transmission errors observed. When the baud rate has been increased to 230400, the throughput reaches 7693, while the actual throughput falls to approximately 26.7% of the maximum potential at 28800 bytes per second.

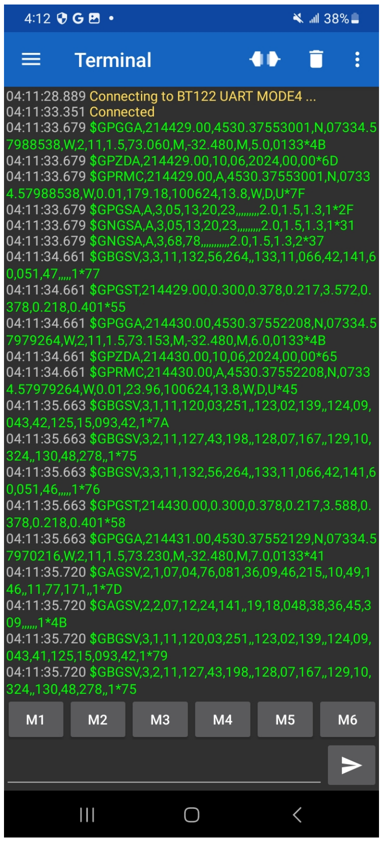
To mitigate Bluetooth delay, we tested the system's maximum throughput capabilities at the current baud rate of 115200 bps. The maximum packet size was set to 250 bytes to minimize Bluetooth latency. The microprocessor transmitted packets as rapidly as possible to the three applications, each containing up to 250 bytes of data. Thirty-one packets were transmitted, achieving a throughput of 7590 bytes per second, representing 52.7% of the theoretical maximum.

Transmission Type	**Data Transmitted (bytes/sec)**	**Percentage of Potential Maximum**
Normal Transmission	6152	42.7%
Adjusted Packets	7590	52.7%

Table 5.2: Comparison of Data Transmission at Baud Rate 115200

Increasing the baud rate can enhance throughput but may reduce actual data transmission efficiency and introduce more errors. Effective packet management can significantly improve throughput efficiency.

In scenarios where the security protocol is disabled, allowing plain text transmission over the Serial Port Profile (SPP), any mobile device can initiate data transmission without authentication. This setup permits unauthorized access to the transmitted data. Figure 5.4 illustrates a non-secure connection scenario using a random serial port application from an app store, demonstrating how easily an unauthorized mobile device can access the data.



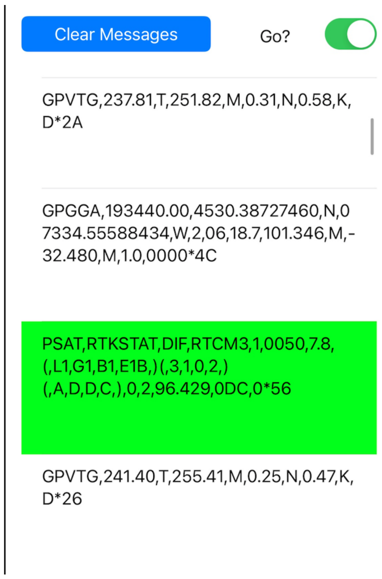
```
04:11:28.889 Connecting to BT122 UART MODE4 ...
04:11:33.351 Connected
04:11:33.679 $GPGGA,214429.00,4530.37553001,N,07334.5
7988538,W,2,11,1.5,73.060,M,-32.480,M,5.0,0133*4B
04:11:33.679 $GPZDA,214429.00,10,06,2024,00,00*6D
04:11:33.679 $GPRMC,214429.00,A,4530.37553001,N,0733
4.57988538,W,0.01,179.18,100624,13.8,W,D,U*7F
04:11:33.679 $GPGSA,A,3,05,13,20,23,,,,,,,,,2.0,1.5,1.3,1*2F
04:11:33.679 $GNGSA,A,3,05,13,20,23,,,,,,,,,2.0,1.5,1.3,1*31
04:11:33.679 $GNGSA,A,3,68,78,,,,,,,,,,,2.0,1.5,1.3,2*37
04:11:34.661 $GBGSV,3,3,11,132,56,264,,133,11,066,42,141,6
0,051,47,,,,,1*77
04:11:34.661 $GPGST,214429.00,0.300,0.378,0.217,3.572,0.
378,0.218,0.401*55
04:11:34.661 $GPGGA,214430.00,4530.37552208,N,07334.5
7979264,W,2,11,1.5,73.153,M,-32.480,M,6.0,0133*4B
04:11:34.661 $GPZDA,214430.00,10,06,2024,00,00*65
04:11:34.661 $GPRMC,214430.00,A,4530.37552208,N,0733
4.57979264,W,0.01,23.96,100624,13.8,W,D,U*45
04:11:35.663 $GBGSV,3,1,11,120,03,251,,123,02,139,,124,09,
043,42,125,15,093,42,1*7A
04:11:35.663 $GBGSV,3,2,11,127,43,198,,128,07,167,,129,10,
324,,130,48,278,,1*75
04:11:35.663 $GBGSV,3,3,11,132,56,264,,133,11,066,42,141,6
0,051,46,,,,,1*76
04:11:35.663 $GPGST,214430.00,0.300,0.378,0.217,3.588,0.
378,0.218,0.401*58
04:11:35.663 $GPGGA,214431.00,4530.37552129,N,07334.5
7970216,W,2,11,1.5,73.230,M,-32.480,M,7.0,0133*41
04:11:35.720 $GAGSV,2,1,07,04,76,081,36,09,46,215,,10,49,1
46,,11,77,171,,1*7D
04:11:35.720 $GAGSV,2,2,07,12,24,141,,19,18,048,38,36,45,3
09,,,,,,1*4B
04:11:35.720 $GBGSV,3,1,11,120,03,251,,123,02,139,,124,09,
043,41,125,15,093,42,1*79
04:11:35.720 $GBGSV,3,2,11,127,43,198,,128,07,167,,129,10,
324,,130,48,278,,1*75
```

Figure 5.4: Non-secure connection scenario

This analysis emphasizes optimizing baud rate and packet management for efficient data transmission while maintaining data integrity and security in multi-application environments.

5.1.3 RTK Correction

The left figure of Fig. 5.5 displays the application showing the received response from the GNSS receiver. The message highlighted in green specifically indicates the current RTK status, showing that the GNSS receiver is currently receiving RTCM3 messages, which are used for RTK correction data. The differential age field is set to one, indicating that the correction occurred one second ago.

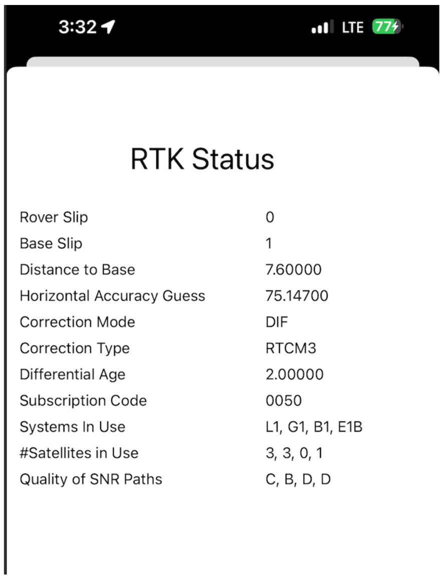


```
GPVTG,237.81,T,251.82,M,0.31,N,0.58,K,
D*2A

GPGGA,193440.00,4530.38727460,N,0
7334.55588434,W,2,06,18.7,101.346,M,-
32.480,M,1.0,0000*4C

PSAT,RTKSTAT,DIF,RTCM3,1,0050,7.8,
(,L1,G1,B1,E1B,)(,3,1,0,2,)
(,A,D,D,C,),0,2,96.429,0DC,0*56

GPVTG,241.40,T,255.41,M,0.25,N,0.47,K,
D*26
```

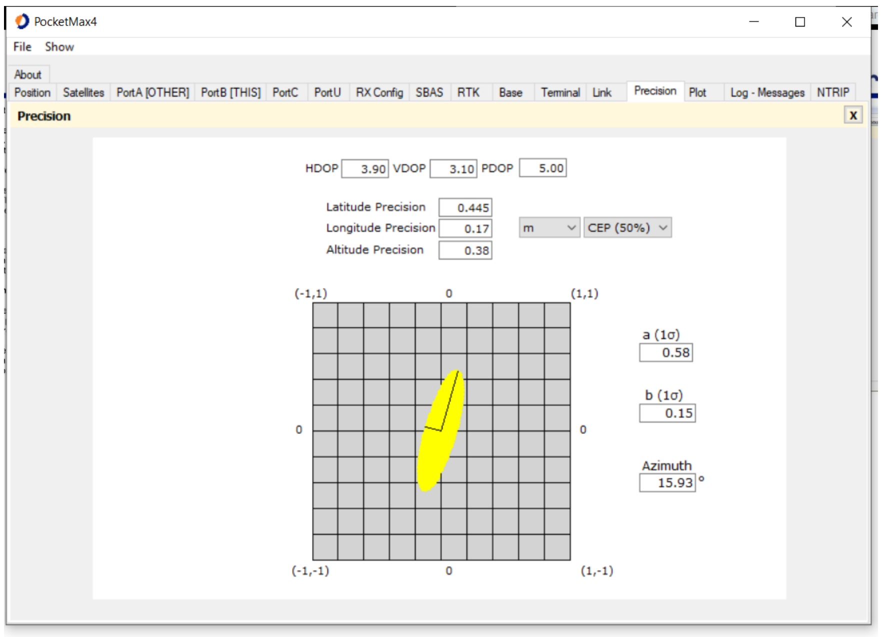


RTK Status

Rover Slip	0
Base Slip	1
Distance to Base	7.60000
Horizontal Accuracy Guess	75.14700
Correction Mode	DIF
Correction Type	RTCM3
Differential Age	2.00000
Subscription Code	0050
Systems In Use	L1, G1, B1, E1B
#Satellites in Use	3, 3, 0, 1
Quality of SNR Paths	C, B, D, D

Figure 5.5: RTK Status Display on the Application

Figure 5.6 presents the RTK accuracy statistics from the Hemisphere PocketMax application. This application provides an estimate of the accuracy of current GPS data. It indicates that the current location data precision is approximately 0.445 meters in latitude, 0.17 meters in longitude, and 0.38 meters in altitude. Although theoretical RTK correction can achieve centimetre-level accuracy, our access to the RTK station is limited, with the closest station being approximately 10 kilometres from the rover. This distance contributes to the degradation of accuracy. Nevertheless, the corrected data represents a significant improvement compared to the standard GPS accuracy of around 10 meters. Improving signal quality and gaining better access to the base station could further enhance the accuracy of the current system.


PocketMax4
File Show
About
Position Satellites PortA [OTHER] PortB [THIS] PortC PortU RX Config SBAS RTK Base Terminal Link Precision Plot Log - Messages NTRIP
Precision
(-1,1) 0 (1,1)
0 0
(-1,-1) 0 (1,-1)

HDOP 3.90 VDOP 3.10 PDOP 5.00

Latitude Precision 0.445
Longitude Precision 0.17
Altitude Precision 0.38

m CEP (50%)

a (1σ) 0.58

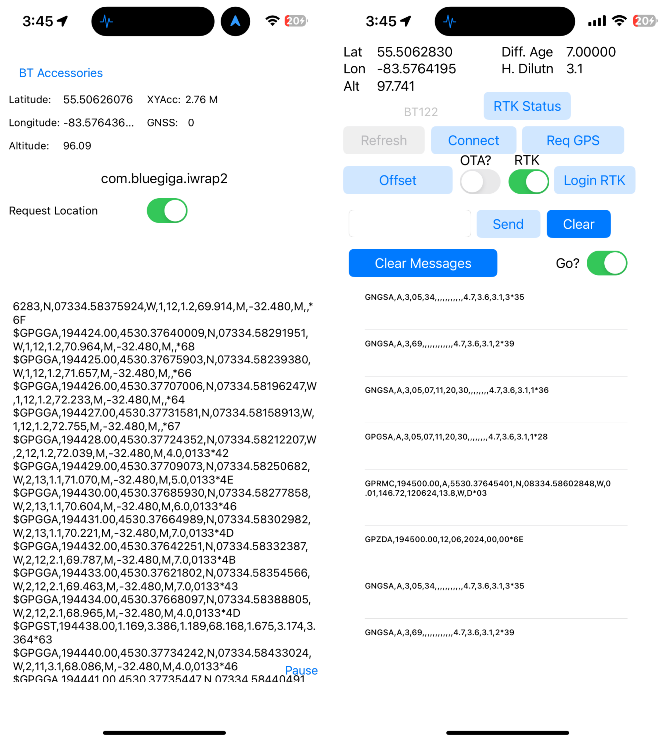
b (1σ) 0.15

Azimuth 15.93 °

Figure 5.6: Accuracy Analysis from PocketMax Application

5.1.4 Location Shift Feature

In this experiment, we applied a location shift to the original coordinates. The shift parameters were set to +10 degrees in latitude, -10 degrees in longitude, and +20 meters in altitude. After applying these shifts, the resulting location is shown in Fig. 5.7. Both applications displayed the shifted latitude, longitude, and altitude, replacing the original system location data as illustrated in Fig. 5.1. The new coordinates place the location in the middle of the sea, which is a significant location change.



Latitude: 55.50626076 XYAcc: 2.76 M
Longitude: -83.576436... GNSS: 0
Altitude: 96.09

Request Location

```
6283,N,07334.58375924,W,1,12,1.2,69.914,M,-32.480,M,,*
6F
$GPGGA,194424.00,4530.37640009,N,07334.58291951,
W,1,12,1.2,70.964,M,-32.480,M,,*68
$GPGGA,194425.00,4530.37675903,N,07334.58239380,
W,1,12,1.2,71.657,M,-32.480,M,,*66
$GPGGA,194426.00,4530.37707006,N,07334.58196247,W
,1,12,1.2,72.233,M,-32.480,M,,*64
$GPGGA,194427.00,4530.37731581,N,07334.58158913,W,
1,12,1.2,72.755,M,-32.480,M,,*67
$GPGGA,194428.00,4530.37724352,N,07334.58212207,W
,2,12,1.2,72.039,M,-32.480,M,4.0,0133*42
$GPGGA,194429.00,4530.37709073,N,07334.58250682,
W,2,13,1.1,71.070,M,-32.480,M,5.0,0133*4E
$GPGGA,194430.00,4530.37685930,N,07334.58277858,
W,2,13,1.1,70.604,M,-32.480,M,6.0,0133*46
$GPGGA,194431.00,4530.37664989,N,07334.58302982,
W,2,13,1.1,70.221,M,-32.480,M,7.0,0133*4D
$GPGGA,194432.00,4530.37642251,N,07334.58332387,
W,2,12,2.1,69.787,M,-32.480,M,7.0,0133*4B
$GPGGA,194433.00,4530.37621802,N,07334.58354566,
W,2,12,2.1,69.463,M,-32.480,M,7.0,0133*43
$GPGGA,194434.00,4530.37668097,N,07334.58388805,
W,2,12,2.1,68.965,M,-32.480,M,4.0,0133*4D
$GPGST,194438.00,1.169,3.386,1.189,68.168,1.675,3.174,3.
364*63
$GPGGA,194440.00,4530.37734242,N,07334.58433024,
W,2,11,3.1,68.086,M,-32.480,M,4.0,0133*46
$GPGGA,194441.00,4530.37735447,N,07334.58440491
```

Lat 55.5062830 Diff. Age 7.00000
Lon -83.5764195 H. Dilutn 3.1
Alt 97.741

Go?

```
GNGSA,A,3,05,34,,,,,,,,,,,4.7,3.6,3.1,3*35

GNGSA,A,3,69,,,,,,,,,,,,4.7,3.6,3.1,2*39

GNGSA,A,3,05,07,11,20,30,,,,,,,,4.7,3.6,3.1,1*36

GPGSA,A,3,05,07,11,20,30,,,,,,,,4.7,3.6,3.1,1*28

GPRMC,194500.00,A,5530.37645401,N,08334.58602848,W,0
.01,146.72,120624,13.8,W,D*03

GPZDA,194500.00,12,06,2024,00,00*6E

GNGSA,A,3,05,34,,,,,,,,,,,4.7,3.6,3.1,3*35

GNGSA,A,3,69,,,,,,,,,,,,4.7,3.6,3.1,2*39
```

Figure 5.7: Results after Applying the Location Shift

Similar experiments have been conducted and tested for over a month, with each experiment consistently yielding the correct results.

This illustrates the capability of the program to accurately and reliably parse the NMEA sentence and extract the necessary fields to apply the offset from the mobile device. The accuracy of the adjusted values is contingent on the precision of the GNSS receiver provided. Therefore, the adjusted value can match the accuracy level attainable by the GNSS receiver.

This feature enables highly precise applications. Moreover, upon issuing the command to stop the location shift, the location information will revert to its original value, ensuring that the original information is preserved for the user to freely access both old and new locations. Additionally, the program's ability to store the last location shift value on the accessory allows the mobile application to consistently refer back to previous results, preventing information loss and facilitating seamless interaction.

This experiment demonstrates the system's capability to handle and accurately display shifted geospatial data. It clearly illustrates the impact of manual coordinate adjustments on the resulting geolocation, emphasizing the importance of precise data handling in Location applications.

Chapter 6

Conclusions and Future Work

This thesis has presented an in-depth analysis of secure Bluetooth connections integrated with RTK (Real-Time Kinematic) technology for enhanced location-based services. The implementation of the HMAC-SHA256 for authentication has demonstrated robust security. It makes brute-force attacks impractical and reduces the likelihood of collision findings. The independence of HMAC from the hashing function allows future upgrades to more robust hashing algorithms without significant overhauls of the system.

The system's performance was evaluated through various experiments. The secure Bluetooth connection successfully established and maintained data transmission between the mobile device and the accessory. Extensive testing confirmed the stability and reliability of the secure connection.

The RTK performance evaluation highlighted the system's ability to significantly improve location accuracy. Although theoretical RTK corrections can provide centimetre-level precision, practical limitations, such as the distance to the nearest RTK base station (approximately 10 kilometres), resulted in a location precision of roughly 0.445 meters in latitude, 0.17 meters in longitude and 0.38 meters in altitude. Despite these limitations, the corrected data represented a considerable improvement over the standard GPS accuracy of around 10 meters.

The study also explored the system's capacity to support multiple applications with different communication protocols running concurrently. The accessory could transmit up to 68 packets per second, achieving a maximum throughput of 6152 bytes per second, which is about 42.72% of the theoretical maximum achievable through wired UART transmission. Further optimization of baud rates and packet management strategies demonstrated that while increasing the baud rate could improve throughput, it also introduced more errors and inefficiencies.

The location shift functionality was tested by applying predefined offsets to latitude, longitude, and altitude. The system successfully handled and displayed the shifted geospatial data, proving the efficacy of the location shift feature.

In addition, the reconnection protocol for error handling was rigorously tested. The Independent Watchdog (IWDG) timer ensured system stability by resetting the system if no data packet was transmitted within the designated time frame. This feature was crucial in maintaining the reliability of the data transmission process, especially in scenarios where the application was closed by the user, necessitating an automatic refresh to prevent undesired system resets.

This thesis has demonstrated the feasibility and effectiveness of integrating secure Bluetooth connections with RTK technology for precise location-based services. The findings highlight the potential for further development and optimization to enhance the system's performance and reliability. This work builds a strong foundation for future advancements in secure IoT communications and high-accuracy geospatial applications by addressing the current constraints.

Future work

Due to the limitation of Bluetooth Classic technology, the current system can only support peer-to-peer connections. Future work can focus on developing and expanding the current system to point-to-multi-point connections, such as utilizing overlaid multiplexing technique [25] to establish a secured and robust local Bluetooth connection network. This can further expand the range and capabilities of the current system.

Additionally, the current location application could be further implemented with blockchain systems, which has the potential to greatly enhance blockchain systems based on the proof-of-location consensus [26]. By functioning as a highly precise node within the network, the RTK technology can dramatically enhance the reliability of location data stored on the blockchain. This is achieved through the application's ability to provide centimetre-level accurate geographic coordinates. Additionally, the network-based nature of RTK introduces a secondary layer of authentication by continuously verifying position data against a network of base stations. This approach, combining RTK accuracy and blockchain security, not only ensures the precision and authenticity of recorded locations but also mitigates the risk of location spoofing, thereby enhancing the overall trustworthiness of the system.

The extended security and location accuracy can be also applied to a variety of human activity tracking projects, including the life-saving ones, such as the tracking for closed-loop control of the diabetes [27].

Bibliography

[1] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, "Performance evaluation of bluetooth low energy: A systematic review," *Sensors*, vol. 17, no. 12, p. 2898, 2017. [Online]. Available: https://doi.org/10.3390/s17122898

[2] B. SIG, "Bluetooth core specification: Service discovery protocol (sdp) specification," n.d., accessed: June 9, 2024. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/service-discovery-protocol--sdp--specification.html

[3] B. Eissfeller, D. Dötterböck, S. Junker, C. Stöber, and Munich, "Online gnss data processing – status and future developments," 09 2011.

[4] J. Wang, K. Han, A. Alexandridis, Z. Zilic, Y. Pang, and J. Lin, "An asic implementation of security scheme for body area networks," 05 2018, pp. 1–5.

[5] MOKOSmart, "Bluetooth vs. bluetooth low energy: A detailed comparison," 2024. [Online]. Available: https://www.mokosmart.com/bluetooth-vs-bluetooth-low-energy-a-detailed-comparison/

[6] Bluetooth, "Bluetooth devices are poised to rapidly scale in 2023," https://www.bluetooth.com/blog/bluetooth-devices-are-poised-to-rapidly-scale-in-2023/, 2023, accessed: 2023-06-13.

[7] M. Zubair, D. Unal, A. Al-Ali, and A. Shikfa, "Exploiting bluetooth vulnerabilities in e-health iot devices," in *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems (ICFNDS '19)*, July 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3341325.3342000

[8] B. SIG, "RFCOMM WITH TS 07.10 Serial Port Emulation," Prepared by N.B. Barb, barb-main@bluetooth.org, November 2012, approved Adopted, Revision V12, Document No RFCOMM_SPEC.

[9] ——, “Logical link control and adaptation protocol (l2cap) specification,” n.d. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/logical-link-control-and-adaptation-protocol-specification.html

[10] ——, “Generic access profile (gap) specification,” n.d., accessed: June 9, 2024. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/generic-access-profile.html

[11] M. T. Inc., “Overview of the bluetooth generic attribute profile (gatt),” 2023, accessed: June 9, 2024. [Online]. Available: https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/bluetooth-host-layer/bluetooth-generic-attribute-profile-gatt/Overview/#:~:text=The%20Generic%20Attribute%20Profile%20(GATT,etc.

[12] B. SIG, “Attribute protocol (att) specification,” n.d., accessed: June 9, 2024. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/attribute-protocol--att-.html

[13] ——, “Security manager specification,” n.d., accessed: June 9, 2024. [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/security-manager-specification.html

[14] Bluetooth, “Technology overview,” 2024, accessed: 2024-06-09. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[15] Nordic Semiconductor, “The difference between classic bluetooth and bluetooth low energy,” 2024, accessed: 2024-06-09. [Online]. Available: https://blog.nordicsemi.com/getconnected/the-difference-between-classic-bluetooth-and-bluetooth-low-energy

[16] A. El-Mowafy, “Precise real-time positioning using network rtk,” in *Global Navigation Satellite Systems: Signal, Theory and Applications*, S. Jin, Ed. InTech Publishing, 2012, pp. 161–188.

[17] G. Wübbena, A. Bagge, and M. Schmitz, “Network based techniques for rtk applications,” in *Proceedings of GPS Symposium 2001*. Tokyo, Japan: Japan Institute of Navigation, November 14–16 2001, pp. 53–65.

[18] R. M. Alkan, S. Erol, V. İlçi, and M. Ozulu, “Comparative analysis of real-time kinematic and ppp techniques in dynamic environment,” *Measurement*, vol. 163, p. 107995, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224120305339

[19] M. Mehic *et al.*, “Quantum cryptography in 5g networks: A comprehensive overview,” *IEEE Communications Surveys and Tutorials*, pp. 1–1, 2023.

[20] A. Khurshid and S. Raza, “Autocert: Automated toctou-secure digital certification for iot with combined authentication and assurance,” *Computers and Security*, vol. 124, p. 102952, 2023.

[21] X. Jiang, R. Dou, Q. He, X. Zhang, and W. Dou, “Edgeauth: An intelligent token-based collaborative authentication scheme,” *Software - Practice and Experience*, April 2023.

[22] S. Yang, X. Zheng, G. Liu, and X. Wang, “Iba: A secure and efficient device-to-device interaction-based authentication scheme for internet of things,” *Computer Communications*, vol. 200, no. September 2022, pp. 171–181, 2023.

[23] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5280

[24] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for Message Authentication,” Network Working Group, Request for Comments 2104, February 1997, category: Informational.

[25] A. Abuzneid, S. Patel, V. U. Mohammed, and V. K. Godula, “Multiplexing overlays on bluetooth,” in *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics*, T. Sobh, K. Elleithy, A. Mahmood, and M. A. Karim, Eds. Dordrecht: Springer Netherlands, 2008, pp. 375–383.

[26] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli, “Blockchain-based proof of location,” in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion.* Parma, Italy: Distributed Systems Group, University of Parma, 2018.

[27] K. Radecka and Z. Zilic, “Energy and food consumption tracking for weight and blood glucose control,” *US Patent Application 14/570,070*, 2016.