CHAPTER 1.
INTRODUCTION

1.1 Background

Radio Frequency Identification (RFID) is a wireless automatic identification technology that uses radio waves to transfer data between a data carrier device (such as a chip on a tag) and a reader device. One of the most commonly used types of RFID is Ultra-High Frequency (UHF) RFID [1]. Compared to other types of low-frequency RFID, UHF RFID has a broader reading range and higher data transfer speed. Because of its ability to automatically identify objects, which can reduce human error and increase operational efficiency, UHF RFID technology has been widely implemented in various types of applications and sectors, ranging from logistics [2], manufacturing [3], retail [4], and sports [5], [6]. Along with the increasing implementation of UHF RFID technology in various applications and sectors, various research is also being carried out to improve performance and overcome the limitations of this technology. It not only focuses on the development of the hardware or the system [7], [8], [9], [10], [11] but also on developing optimal and efficient data processing algorithms to overcome problems such as data collision [12], [13], reading inaccuracy/uncertainty [14], [15], [16], interference, and data redundancy [17]. Conversely, the emergence of edge computing technology enables data processing on devices proximate to the data source, hence diminishing latency and bandwidth demands for transmitting data to a central server [18]. Implementing edge computing in UHF RFID systems enhances data processing efficiency by transferring some processing tasks to edge or proximate devices, diminishing data communication burdens on the central server and decreasing process latency[19].

One of the main challenges of implementing an RFID-based system in a dynamic environment with high data volume is data redundancy, which occurs when the reader device repeatedly reads the same tag within a specific time interval. This redundancy can be spatial, where the same tag is read by multiple readers, or temporal, where the same tag is read repeatedly by the same reader [17]. The presence of this data redundancy results in a high processing load, excessive memory usage (inefficient), and the potential disruption of data accuracy and integrity [20]. To address these

issues, several studies with various approaches and methods have been conducted. Some of them are displayed in Table 1.1.

Rui et al. [21] introduced the use of the sliding window technique with a temporal-spatial bloom filter to address the data redundancy. In line with that research, Dihua et al. [22] proposed the use of the dynamic temporal bloom filter (DTBF) method, which considers the time difference during RFID data reading. Wang et al. [17] improved upon this by proposing the Time-Distance Bloom Filter (TDBF) method, which considers both time and distance as filter parameters. A similar approach was also taken by Cao et al. [23] by introducing the R-TDBF method, which is adaptive to signal strength and environmental conditions. Meanwhile, Dash et al. [24] introduced the use of Voronoi diagrams and temporal filters to reduce redundant data in multi-reader systems with spatial region partitioning. On the other hand, by analyzing the pattern of duplicate data at 21 RFID stations, Yang et al. [25] proposed a cleansing method based on statistical analysis. Improvements to the statistical smoothing for unreliable RFID (SMURF) algorithm for dynamic data cleansing have also been conducted in the research by Xu et al. [26].

In the context of RFID middleware, He et al. [27] introduced a middleware-based algorithm to manage reader elimination to address spatial redundancy efficiently. Similar research was also conducted by Ma et al. [28], who used the Threshold Selection Algorithm (TSA) in the elimination of excessive reader readings for energy efficiency. Another approach was taken by Ouadou et al. [29] by dividing the redundancy problem into intra- and extra-path duplication and suggesting the use of proactive filtering through network topology. Meanwhile, Li et al. [19] developed an adaptive sliding window method that operates on edge devices for cleansing single-tag and multi-tag RFID data.

The bloom filter method is widely used in research related to filtering RFID redundancy data because it offers good memory space efficiency and operational speed. However, the use of multiple hash functions in Bloom filters has the potential to burden devices with limited resources, such as edge devices or embedded devices [30]. Therefore, several studies have attempted to address this by making modifications, specifically by using a single hash function, as done by Kamaludin et al. [20] and Mahdin et al. [31]. Additionally, hash-based solution approaches and deduplication techniques also have the potential to be implemented to address the redundancy issue [32], [33]. Ahmed et al. [34] have developed a lightweight

deduplication method using triple-level content-based hashing. Meanwhile, Hassan et al. [35] have developed a 32-bit hash function using genetic programming that is efficient for embedded systems such as FPGA.

Table 1.1 State of The Art

Authors	Year	Methode	Research Field	Contribution
Hassan et al. [35]	2024	Genetic Hashing 32-bit	Hardware Oriented hashing	Development of lightweight 32-bit hash for resource-constrained hardware devices
Deore et al. [33]	2024	AI-driven Deduplication	AI-Based data deduplication	AI-based deduplication trends
Cao et al. [23]	2022	R-TDBF (Threshold Adaptive) – Bloom Filter	Environmental Adaptive RFID Filtering	Adaptive filtering to the environment using signal thresholds
Ahmed et al. [34]	2022	Triple-level Hash + Chunking	Leightweight Hash deduplication	Efficient deduplication with triple-level hash
Li et al. [19]	2022	Adaptive Sliding Window + Double Estimation	Adaptive data Filtering	Real-time data cleansing on edge systems
Dash et al. [24]	2022	Voronoi + Temporal Filter	Spatio-Temporal Filtering	Reducing multiple reading by multi-reader with spatial region division
Dabhade et al. [7]	2021	RFID Tracking	UHF RFID System Applications	RFID applications for event tracking
Wang et al. [17]	2020	TDBF (Time-Distance Bloom Filter)	Spatio-Temporal Filtering	Time and distance-based filtering for dynamic RFID data
Ouadou et al. [29]	2019	Intra & Extra Path Filtering	RFID-Sensor network filtering	Filtering based on reading topology paths
Ma et al. [28]	2018	TSA (Threshold Selection Algorithm)	Redundant Reader Elimination	Elimination of excess readers in reading tag at the same time
Xu et al. [26]	2018	SMURF Improvement	Statistical RFID Data Cleansing	Redundancy reduction/data cleaning in dynamic environments
Zhang et al. [32]	2017	Deduplication Techniques Overview	General Deduplication survey	Introduces data deduplication techniques, principles, and future recommendations."
HE et al. [27]	2017	Middleware + Redundant Reader Removal	RFID Middleware Optimization	The algorithm effectively identifies redundant readers in RFID systems.
Kamaludin et al. [20]	2016	Bloom Filter Hash Tunggal	RFID Middleware Filtering	Fast and efficient filtering with low false positives
Yaacob et al. [30]	2016	kinds of Bloom Filter	RFID Data Filtering theory	Providing a theoretical basis for Bloom Filter-based filtering
Dihua et al. [22]	2016	Dynamic Time Bloom Filter	Temporal Filtering	Dynamic time-based hash filtering
YANG et al. [25]	2016	Redundancy analysis	Redundancy cleansing characterization	Redundancy analysis based on duplication and similarity patterns
Mahdin et al. [31]	2015	Bloom Filter	RFID Supply Chain Filtering	Effective redundancy data filtering algorithm in RFID supply chain systems
Rui et al. [21]	2014	Sliding Window (Temporal-Spatial Bloom Filter)	Spatio-Temporal Filtering	Dynamic filtering based on time and location

While numerous studies have enhanced the performance of UHF-RFID systems, especially in redundancy data (as shown in Table 1.1), particular challenges persist that must be addressed to achieve greater efficiency, especially in the implementation of edge computing with embedded devices in UHF-RFID-based systems. Although the spatio-temporal filtering and reader elimination approaches as conducted by Rui et al. [21] , He et al. [27], Ma et al. [28], Wang et al. [17], dan Dash et al. [24] can reduce data redundancy and improve energy efficiency, these methods are still focused on implementation on centralized computing platforms, making them less suitable for implementation on embedded devices with limited resources, and they also require complex infrastructure for their implementation. Meanwhile, the research results do not indicate that the method entirely eliminates data redundancy, suggesting that data redundancy cannot be entirely eliminated by applying this method.

Bloom Filter, which has been claimed by most previous research to have high memory usage efficiency and computational speed [23] [17] [20] [22] [31] [21]. However, this algorithm has a weakness, namely the possibility of False Positives. This condition occurs when the Bloom Filter identifies that a specific RFID tag "might" have been read before, even though it has not. The occurrence of false positives in RFID systems indicates the potential loss of valid data/missing data, reduced tracking accuracy, and difficulties in data diagnostics. All of these effects are conditions that pose significant risks for applications with minimal tolerance for missing data, such as those in retail systems, supply chains, manufacturing, and even sports.

The development of lightweight hash algorithms by Ahmed et al. [34], dan Hassan et al. [35] has the potential to be a new approach to addressing deterministic data redundancy on middleware devices with resource limitations. However, the research has not specifically targeted the deduplication process on RFID data streams. Additionally, deduplication approaches that directly integrate hash functions into RFID edge middleware have not been widely encountered.

To address the limitations and gaps in previous research related to RFID data redundancy, this study proposes a system model for filtering RFID data redundancy using a lightweight hash-based deduplication algorithm approach optimized for RFID data streams to run on embedded devices. Unlike the approaches in previous research that used fixed sliding windows [21] and adaptive sliding windows [19], the system proposed in this study employs an idle-time-based window mechanism, where the

window for deduplication is only opened when an incoming RFID data stream is detected. It will only be closed when there is no data incoming within a specific time interval (i.e., an idle condition). The approach proposed in this study combines the efficiency of hash data structures with an adaptive time mechanism based on data absence (session-based trigger) optimized for the trade-off between memory usage and processing speed to be implemented in an RFID middleware device model based on embedded devices. With this approach, it is expected to minimize the occurrence of missing data and eliminate data redundancy.

The system developed in this study includes parsing mechanisms, data stream deduplication using the proposed hash algorithm, time labeling and time synchronization, as well as data output formatting. In this research, a lightweight hash function is developed that can efficiently manage RFID datastream duplication in memory. The system's performance is tested based on four main parameters: accuracy, latency, throughput, and memory usage. The system's performance will also be fairly compared with several other deduplication methods to determine which deduplication algorithm is the most optimal and efficient to integrate with the system model. This research is expected to provide a practical and efficient solution to address data redundancy issues in RFID-based systems, particularly within the context of implementation on embedded devices and small-scale IoT-based applications. These solutions are low-cost and contribute to research in the field of RFID in general.

1.2 Problem Identification

One of the key challenges in RFID-based systems is data redundancy, which occurs in two forms: spatial redundancy, when multiple readers simultaneously detect the same tag within overlapping zones; and temporal redundancy, when a single reader reads the same tag repeatedly within a short time window. This redundancy not only increases memory consumption and processing latency, but can also compromise data integrity, especially in embedded systems with limited computational capacity.

While existing methods such as Bloom filters [23] offer fast and memory-efficient filtering, they suffer from false positives, where a new or unique tag is mistakenly identified as a duplicate. This condition poses a significant risk in RFID applications where data accuracy is critical. Other strategies, including sliding windows [21] [19], and statistical filters [26], though effective in server-based architectures, are typically too resource-intensive or complex to be deployed on

lightweight platforms like the ESP32. Moreover, despite recent efforts to develop lightweight hash functions [34], [35], there is a notable lack of approaches that apply such functions directly to real-time redundancy filtering in streaming RFID data at the edge.

There is a clear research gap in designing a solution that achieves low memory usage, high accuracy, and minimal latency, explicitly tailored for resource-constrained embedded systems. Therefore, this study aims to address this gap by proposing a lightweight, hash-based deduplication approach optimized for real-time RFID data filtering on embedded middleware, enabling efficient, accurate, and scalable edge processing.

1.3 Objective

The objectives of this research are as follows :

1) To design and implement a lightweight deduplication system for RFID data streams using an idle-time window mechanism, optimized for embedded edge devices such as the ESP32.
2) To develop and integrate a modified version of the xxHash32 hashing algorithm that enables high-accuracy, low-latency, and memory-efficient redundancy filtering.
3) To construct a real-time embedded middleware that supports efficient hash-table-based duplicate detection, static memory allocation, and chaining for O(1) lookup performance.
4) To implement a time labeling mechanism using a combination of real-time clock (RTC) and internal counter, with periodic synchronization via network time protocol (NTP), ensuring accurate and consistent timestamping of RFID events.
5) To format and output the filtered RFID data into structured, clean formats compatible with IoT platforms and complex event processing (CEP) systems.
6) To evaluate the proposed system comprehensively across multiple metrics—accuracy, memory usage, deduplication latency, processing latency, and throughput—and compare it with other deduplication methods.

1.4 Scope of Work

The limitations of this research are:

1) The system was developed using the UHF RFID Reader model Middle Range Integrated UHF RFID Reader (UHFReader18 / Electron HW-VX6330) and waterproof wristband UHF RFID tag EPC Gen2 Class-1 (ISO18000-6C) with a frequency range of 840-960MHz.
2) The EPC dataset (Unique ID of UHF RFID) is generated randomly according to the data format output by this type's integrated reader and tag.
3) Edge computing is implemented on an edge device in the form of an embedded device (controller) ESP32.
4) At this stage, the research focuses only on the issue of temporal redundancy, not yet optimized to address spatial redundancy.
5) Only one reader device is used as reference for testing.
6) This research focuses on the most optimal and effective parsing algorithm/method for implementation on edge devices (embedded devices), data filtering (duplicate removal), time synchronization, and time labeling on the detected unique IDs.
7) Does not include communication between readers or advanced backend.
8) Performance testing simulation focuses on processing input data from the reader, which is continuously transmitted and processed by the edge device, without considering non-ideal conditions that may occur during tag reading by the reader.

1.5 Expected Result (Hypothesis)

This research aims to develop an edge computing solution for embedded devices that can present clean and structured RFID data, ready for further processing, such as by a Complex Event Processing (CEP) system. The expected outcomes from the development of this middleware include:

1) Efficient and responsive parsing algorithms

   Development of parsing algorithms capable of handling large volumes of UHF-RFID data in real-time within embedded device environments.
2) Optimal deduplication algorithms for embedded devices

   Development of lightweight and efficient deduplication algorithms to eliminate duplicate data while maintaining memory and power consumption efficiency on ESP32 devices.
3) Efficient time labeling algorithms

   Implementation of time labeling algorithms capable of providing accurate timestamps, enabling good data integration for subsequent processes.

4) Middleware that produces clean and structured data for further processing

RFID middleware that can present data filtered from duplication, equipped with accurate timestamps, and ready to be used in the subsequent processing pipeline.

This research hypothesizes that modifications to the xxHash32 algorithm, when integrated into an idle-time window-based deduplication system, will result in more efficient performance on ESP32-based RFID middleware, particularly in reducing latency, conserving memory usage, and maintaining high accuracy and optimal throughput. This proposed system aligns with the needs of modern RFID systems, which demand not only high accuracy but also resource efficiency at the edge level.

1.6 Novelty and Contribution

This research presents two main novelties that distinguish it from prior studies in the field of RFID data deduplication for embedded middleware systems:

1) Idle-Time Windowing as a Temporal Boundary for Real-Time Deduplication

Most previous approaches rely on fixed or sliding time windows, which typically require full buffering of incoming tag data before deduplication is performed. This can lead to memory inefficiency, increased latency, and the risk of missed tag reads if the window duration does not align with the dynamics of the data stream.

This study proposes a novel idle-time windowing mechanism, which uses a tag's inactivity period as a dynamic temporal boundary for deduplication. Unlike conventional methods, this approach performs real-time processing—each tag is immediately evaluated upon arrival, without waiting for buffer accumulation. This strategy significantly reduces memory usage and improves responsiveness, while maintaining the integrity of the continuous data stream. The strategy is highly suitable for edge-based RFID streaming systems, particularly in memory-constrained embedded environments like the ESP32.

2) Direct Integration of an Optimized Lightweight Hash Algorithm into Embedded Middleware

While lightweight hash functions such as FNV-1a, Murmur3, and genetic-based hashing have been explored in earlier studies for reducing memory complexity, they have not been directly employed as primary deduplication engines within real-time RFID middleware on embedded platforms.

This research addresses that gap by modifying and optimizing the xxHash32 algorithm specifically for short EPC data, and embedding it directly into a streaming

middleware running on the ESP32. The implementation achieved 100% accuracy, with deduplication latency as low as 1.137 μs, and memory usage of only 404 KB RAM and 754 KB flash, making it a practical and scalable solution for real-time data processing in IoT and edge computing applications.

1.7 Structure of Thesis

Here is the thesis structure used:

1) **Chapter 1:** Introduction, background, problem identification, objectives, scope, and research hypothesis.
2) **Chapter 2:** Literature review and basic theory on RFID, deduplication technique, hash function, and time syncronization.
3) **Chapter 3:** Research roadmap, data systematic, proposed system model, proposed deduplication algorithm, performance parameter, and testing scenarios.
4) **Chapter 4:** Result of testing algorithm and comparative analysis of proposed algorithm.
5) **Chapter 5:** Conclusion, limitation, and future work.