# RFID Inventory Management System Sampling Optimization Based on Zebra Android Framework

I. Benke, I. Heđi and E. Ciriković
Virovitica University of Applied Sciences, Virovitica, Croatia
ivan.benke@vuv.hr, ivan.hedi@vuv.hr, enes.cirikovic@vuv.hr

**Abstract -** ***In services with a high rate of input and output of items, the inventory data collection process often imposes high demands on staff in terms of reliable evidencing and overall repeatability. Certain business services, such as on-site laundry facilities in public hospitals, provide a good platform for the ad-hoc implementation of custom ICT (Information and Communication Technology) inventory management solutions, especially if they have already partially implemented proven technologies like RFID. Considering the relatively low cost and ease of design, development, and implementation of such systems in public institutions or other industry segments where accurate and reliable daily-based item inventory is crucial, it offers a viable alternative to commercially available inventory management systems. Processing operations for data elements are essential components of any information system. These operations are optimized with a focus on speed and reliability to meet the timing requirements of RFID sampling. This includes improvements to the database scheme, as well as the application and adaptation of existing algorithms for retrieving duplicate elements from data sets.***

***Keywords – RFID; inventory; algorithms; optimization (key words)***

## I. INTRODUCTION

### *A. Data collection and processing in general*

Data collection and processing using inventory means are common in work organizations, whether manual or automated. Inventory assets are marked with inventory codes, such as barcodes or QR codes, or more advanced RFID (Radio Frequency Identification) transponders. The interface to the database is typically realized through a screen form, where fields can be filled out using electronic readers or by entering inventory codes directly. However, the latter method is not suitable for recording a large number of inventory assets due to its inefficiency. In contrast, using electronic devices and supporting technologies, such as RFID, offers advantages in terms of speed, reliability, and mobility of the system. The basic architecture of an RFID-based system for recording inventory assets consists of a fixed or mobile RFID reader with an associated antenna and a remote transponder attached to the inventory asset. The codes of inventory assets are stored in the transponders' memory, such as the Electronic Product Code (EPC) memory or the Tag ID (TID) memory. The RFID reader sends an electromagnetic wave to the transponder, which powers the transponder to send back a response with the unique product code. The data is then saved on the RFID reader or sent to a remote terminal for further processing. The paper is structured as follows: it will describe the general system components and architecture, touch on basic implementation aspects of management system architecture and searching algorithms in a data structure after the short introduction.

### *B. Related work*

RFID technology is well-known term described in many papers. Technology has been widely applied in inventory management, logistics tracking, identity recognition and so on. Most of papers focused on tag searching in large warehouses. In [1] authors highlight two major challenges in tag searching, one is how to exclude the interference caused by unexpected local tags, and the other is how to utilize the known IDs of wanted tags to improve the searching efficiency. They proposed a Bloom filter based tag searching protocol called BFSearch, which consists of two phases: non-wanted tag deactivation phase and target tag verification phase. Bloom filters also applied in [2]. Tags that carry similar information can be grouped into a category. To collect such information, most existing works have to query all tags in each category, which is time-consuming. In [3] authors proposed a new solution called arithmetic coding based sampling (ACS) protocol. They construct a sparse vector to sample only a subset of tags from each category, which can not only avoid repetitive information collection but also reduce interference from unsampled tags. Problem that occurs during RFID data capturing is false positives (i.e., tags that are accidentally detected by the reader but not of interest to the business process). In [4] machine learning algorithms described to filter false positives.

RFID scanner identifies all visible tags in each time interval. However, more than 90 percent of tag serial numbers are duplicates if you perform multiple scanning. These serial numbers need to be eliminated. Our problem is how to efficiently and quickly compare the field of existing tag serial numbers with the field of newly read tag serial numbers during each scan in order to obtain unique values. This problem specifically applies on our test device which is described in next chapter. Multiple scanning in one position is applied to ensure that all tags on a single stack are read.

## II. ZEBRA RFID READER AND APPLICATION SUPPORT

The RFID reader is an essential component of RFID systems and comes in different versions, including fixed and portable types. The first type is mostly associated with industry and production plants where products or their components are moved through the reading zones of fixed-installed readers. A similar application can also be recognized in retail chains, where fixed detection systems at the entrances of stores are used to protect against item theft. Portable readers are used in applications where mobility is required, such as warehouses and other facilities. One such application of RFID reader used refers to hospital laundry management services, where incoming and outgoing articles of clothing, bed linen, and other laundry factory-marked with RFID transponders in the form of sewn-in labels are recorded and entered into the central management system. As part of the solution presented in this paper, the mobile RFID reader RFD40 manufactured by Zebra was used (Fig. 1).

Figure 1. Zebra RFD40 RFID UHF reader with Zebra TC21 management unit

In addition, the Zebra TC21 smart mobile device with the Android 10 system was also used as part of the control module, as shown in Fig. 1. Given that the said mobile device is part of the manufacturer's accessory equipment, it is added to the RFID reader using the appropriate integrated compatible docking station. Although the reader can be operated independently, without the additional element TC21, complete mobility of the entire system and extension of functions is possible only by using the entire package as shown in Fig. 1. The reader itself is designed to work in the UHF frequency range (865-868MHz) and is based on EPC Class 1 Gen 2 and EPC Gen2 V2 wireless access protocols. [5] Communication with the computer is provided via the USB-C interface, which provides the possibility for direct synchronization with management application solutions on the host computer side. In contrast, the built-in system memory buffer, according to the manufacturer, provides temporary storage of up to 40000 read RFID tags and thus enables excellent system autonomy and mobility. The RFID reader also has a built-in battery with a capacity of 7000 mAh, a reading speed of over 1300 RFID tags per second, and a range of about 6 m. It supports Windows, Android, and iOS development environments, and offers open support for upgrading and developing software solutions based on the offered source code and application programming interfaces (APIs). Ready-made applications such as 123RFID Desktop and Mobile provide functionality for searching and recognizing nearby or directly connected RFID readers, as well as antenna parameter adjustments, among others. [6] Both versions of the 123RFID application provide the functionality of searching and recognizing nearby or directly connected RFID readers as well as antenna parameters adjustment, and the latest firmware updates. The central applicative module of the system is responsible for managing the readings of RFID tags. This includes various tasks such as filtering, sorting, grouping, graphical presentation, and exporting raw data, as shown in Figure 2. The module also displays information such as the total number of readings, which is proportional to the default frequency or read rate, and the time of sampling or read time.

One important piece of information displayed in Figure 2 is the number of unique RFID tag values that have been read. This information is particularly valuable in inventory asset record-keeping systems. Keeping track of uniquely read tag values allows for accurate inventory management, as it helps identify individual items or assets that have been tagged and read by the system. This can be useful in tracking the movement of inventory, identifying missing or misplaced items, and reconciling inventory records with actual stock levels. By providing the number of unique RFID tag values, the system can offer a comprehensive overview of the inventory status, helping to streamline inventory management processes and improve overall operational efficiency.

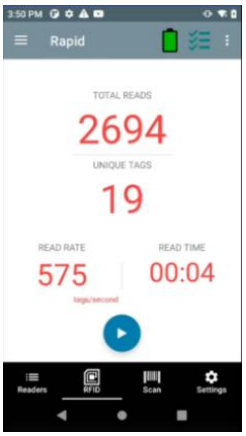
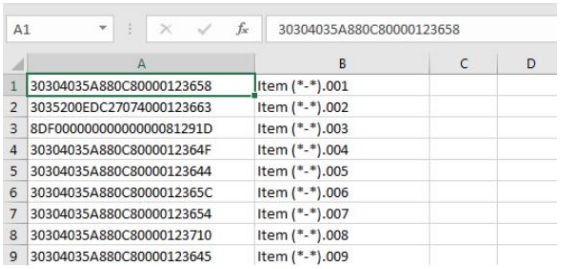



Figure 2. 123RFID application Rapid module window used for RFID tags display [7]

As part of the presented application, there are also options for writing unique identifiers on corresponding RFID transponders using the Tag Write program module, as well as the possibility of locating single or multiple RFID transponders according to the given identifier values. Regardless of the tag reading mode, all readings are written to a CSV file located in the previously created root directory of the Android operating system.



| | A | B | C | D |
|---|---|---|---|---|
| 1 | 30304035A880C80000123658 | Item (*-*).001 | | |
| 2 | 3035200EDC27074000123663 | Item (*-*).002 | | |
| 3 | 8DF00000000000000081291D | Item (*-*).003 | | |
| 4 | 30304035A880C8000012364F | Item (*-*).004 | | |
| 5 | 30304035A880C80000123644 | Item (*-*).005 | | |
| 6 | 30304035A880C8000012365C | Item (*-*).006 | | |
| 7 | 30304035A880C80000123654 | Item (*-*).007 | | |
| 8 | 30304035A880C80000123710 | Item (*-*).008 | | |
| 9 | 30304035A880C80000123645 | Item (*-*).009 | | |

Figure 3. Scanned RFID tags displayed in corresponding CSV file

In the RFID Settings application module, there are specific options related to the operation of the reader's antenna and the type of communication link. In the Profiles configuration group, there are options for manual control of parameters such as the transmitted power of the reader (Power Level) expressed in units of dBm, and the link profile parameter (Link Profile) which determines the product of the symbol transmission rate and the Miller number of senders per symbol (2, 4, or 8), as indicated by the symbols shown in Figure 4.

The choice of the link profile affects the system's ability to successfully detect passive RFID tags, and it involves finding a compromise between data transfer speed, signal-to-noise ratio, and the number of incorrectly detected bits in relation to the number of transmitted ones. For example, the M2 320K profile was chosen for testing and production environment, as it proved to be the most reliable under the given conditions. This choice is based on the EPC Gen2 V2 standard and specific requirements of local regulators for electronic communications.

Furthermore, the system allows defining the Pulse Interval Encoding (PIE) and Type A Reference Interval (Tari) of the symbols of the discrete modulation procedure PR-ASK (Phase Reversal Amplitude Shift Keying), following the standards and requirements mentioned above [8]. These settings are important for ensuring compatibility with relevant standards and regulations, and for optimizing the performance of the RFID system in different operating environments.

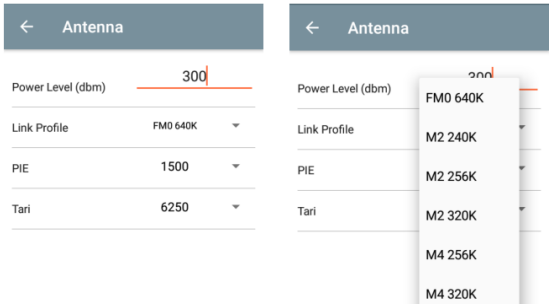



Figure 4. The display of available Zebra RFID reader communication parameters in 123RFID application [7]

The continuation of this paper will present a complete software solution that is based on the described hardware Zebra RFID platform and utilizes the freely available Zebra RFID Software Development Kit (SDK) package for Android systems.

## III. MANAGEMENT SYSTEM ARCHITECTURE

Management system architecture like multilayered architecture is shown on Fig. 5. At the bottom of the system architecture is an RFID handheld scanner used for wirelessly scanning tags at a rate of more than 1000 tag reads per second. The RFID handheld scanner is controlled by a mobile device with an Android operating system and Zebra SDK. The Zebra Scanner SDK is integrated with a mobile application for Android, enabling cordless scanners to be connected and controlled by a tablet or smartphone without using a cradle, after pairing over Bluetooth. Scanning a pairing barcode that appears on a tablet or smartphone display will pair the device and connect it with the application. Each tag is identified by a 24-character string with numbers and capital letters, known as the serial number.

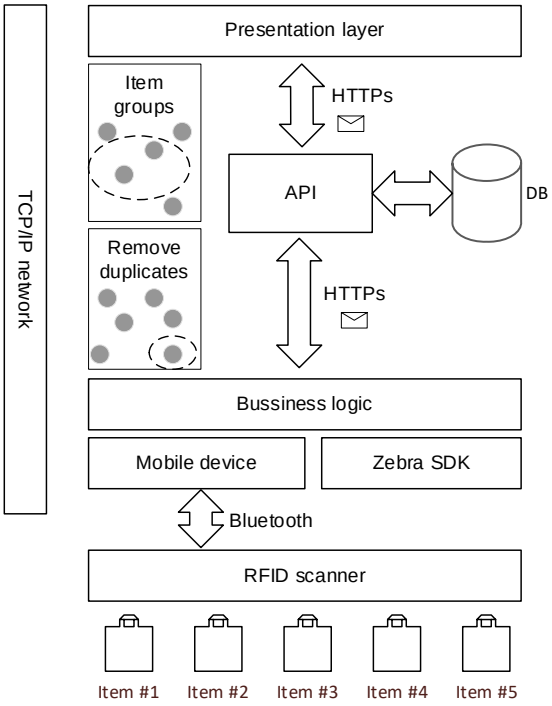



Figure 5. Management system architecture

An example of a serial number written on an RFID tag is:

"300ED89F335000B99A421245".

Given that the sampling frequency is 1kHz, the RFID scanner identifies all visible tags in each time interval. However, more than 90 percent of tag serial numbers are duplicates that need to be eliminated. This action is implemented in the Business logic layer, also known as the mediation layer, which is implemented in the mobile application. The main functions of the mediation layer include data collection from tags, aggregation of collected data on a low level, and preparation for storage in a real-time data warehouse. By removing duplicate values, every HTTP request of the API (Application Programming Interface) becomes lightweight and faster. The storage of the read data in the SQL Server Express database, processing of the results, and the complete web solution are located on a separate remote server location. The prepared data is then sent to the system backend for

further processing. Regardless of the tag reading mode, all readings are written to a CSV file located in the previously created root directory of the Android operating system. Our solution uses direct readings from the RFID reading device shared memory. To send data, it is only necessary to call a function in the API with data stored in JSON structure.

The published message looks like this:

```
{
   "datetime": 1581700066476,
   "operator": 1,
   "warehouse": 1,
   "direction": 1
   "serials": [300ED89F335000B99A421245,…],

}
```

The direction attribute is used to set if selected tags are coming to the warehouse or from the warehouse. The serials attribute is an array of unique values of serial numbers.

The RFID scanner and the mobile device are two interconnected objects with the ability to transfer data to the Internet. When devices are connected to the Internet, they can communicate with other devices or deliver information to specific endpoints [9]. These devices can connect to the Internet directly using standard technologies such as 3G, 4G, and 5G, or they can connect to a local area network that is connected to the Internet. On the other hand, devices can form M2M (Machine to Machine) networks, where devices are connected using radio communication standards and protocols such as Wi-Fi (based on the IEEE 802.11 standard), Bluetooth (based on the IEEE 802.15.1 standard), Zigbee (based on the IEEE 802.15.4 standard), or 6LowPAN over Zigbee (IPv6 over Low Power Personal Area Networks). In most cases, application layer protocols are used for handling communication. Some of the most representative application layer protocols are CoAP (Constrained Application Protocol), MQTT (Message Queue Telemetry Transport), XMPP (Extensible Messaging and Presence Protocol), RESTFUL Services (Representational State Transfer), AMQP (Advanced Message Queuing Protocol), and web socket [10].

If devices are connected to the Internet, there are several cloud providers that offer connectivity between devices and the cloud, such as AWS IoT (Amazon Internet of Things). The primary function of an IoT platform is to act as a middleware layer to connect devices or applications from one end to another end [11].

The top layer of a given architecture is the presentation layer. The presentation layer consists of various reports for controlling item flow. A high level of detection of inconsistency is an algorithm for generating item groups. Several types of serial numbers can form one group in one direction. If there is a group that is not complete, there is a high probability that something was not detected, or items were lost.

The most important problem in this architecture is the causal relationship between the sampling frequency and the speed of duplicates search. If the sampling frequency is increased due to a large number of items, optimization for duplicates search becomes necessary.

## IV. SEARCHING ALGORITHMS IN DATA STRUCTURES

A search algorithm is an algorithm that provides a solution to a problem after evaluating a set of possible solutions. The set of all possible solutions to the problem is called the "search space" [12]. In data structures, two search algorithms are commonly used: sequential and binary. Sequential search is the simplest form of searching and is used on small, unsorted data sets. It goes through every element of the array, making it the slowest search algorithm with a worst case complexity of O(n). Sequential search can also have some subvariants, such as Probability Search, Sentinel Search, or Ordered List Search [13]. An example of sequential search is shown in Fig. 6

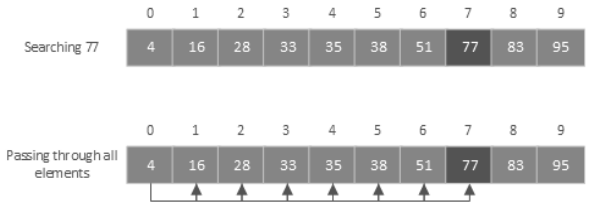



Figure 6. Sequential search

Binary search is a more advanced and faster search algorithm compared to sequential search. The search starts by testing the element in the middle of the array. The middle element is obtained as the average value of the first and last index of the array, mid = (fi+li)/2. If the middle element is the required element, the search ends. If the requested element is smaller than the middle, the search continues in the first part of the field, while the second part of the field is eliminated from further consideration, and vice versa. A simple example is shown in Fig. 7. The worst case complexity of the binary algorithm is O(log n). To use binary search, the main condition is that the array must be sorted. Since the elements of the array are not necessarily arranged according to some criteria, array indexing is used to make search and record sorting faster. Array indexing occurs when adding a new record to the array, and the main disadvantage of this search is worse performance in the case of a large number of additions. There are different variations of binary search (triple search, exponential search, Fibonacci search), and their effectiveness depends on the size of the array being searched.

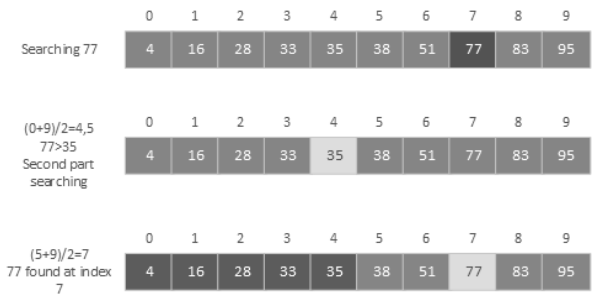



Figure 7. Binary search

A graphical overview of the measured performances of these two types of searches is shown in Fig. 8.

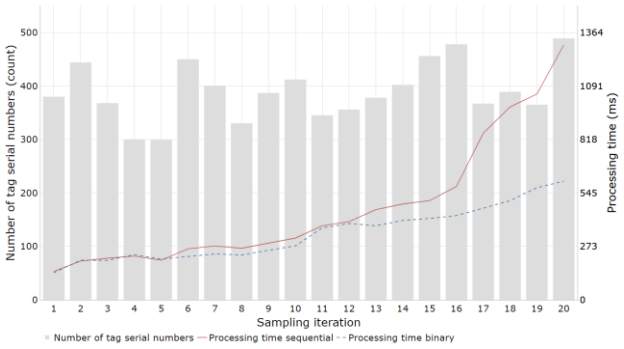



Figure 8. Statistical data comparison

The frequency of sampling serial numbers of tags in a time frame of 1000 ms limited the choice of algorithm in terms of speed. Therefore, binary search was found to be advantageous compared to sequential search. Even though sorting the existing serial numbers was required for binary search, it turned out to be 40% more efficient.

## V. CONCLUSION

RFID technology has been widely adopted for recording inventory goods due to its benefits such as good industrial support, usage flexibility, and widespread standard adoption. The basic architecture of an RFID-based inventory recording system consists of a fixed or mobile RFID reader with an associated antenna, and remote transponders physically attached to the inventory assets. Codes of inventory assets, unambiguously associated with individual transponders, are generated based on a preset sampling frequency. The use of algorithms for handling large data sets is necessary to ensure efficient system performance. In conditions of high sampling frequency of RFID tag serial numbers, it has been demonstrated that the binary search algorithm can save enough time to meet strict time frame requirements before the arrival of a new set of data. Furthermore, given the technical limitations of the HTTP protocol, the application of more modern solutions, such as the MQTT protocol, could further optimize the system in terms of reliability and reading speed, which are critical parameters.

## REFERENCES

[1] Na Yan, Honglong Chen, Kai Lin, Zhichen Ni, Zhe Li, Huansheng Xue, BFSearch: Bloom filter based tag searching for large-scale RFID systems, Ad Hoc Networks,Volume 139, 2023

[2] Z. An, Q. Lin, L. Yang, W. Lou and L. Xie, "Acquiring Bloom Filters Across Commercial RFIDs in Physical Layer," in IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1804-1817, Aug. 2020

[3] J. Liu, S. Chen, Q. Xiao, M. Chen, B. Xiao and L. Chen, "Efficient Information Sampling in Multi-Category RFID Systems," in IEEE/ACM Transactions on Networking, vol. 27, no. 1, pp. 159-172, Feb. 2019

[4] G. Alfian, M. Syafrudin, B. Yoon, and J. Rhee, "False Positive RFID Detection Using Classification Models," Applied Sciences, vol. 9, no. 6, p. 1154, Mar. 2019

[5] https://www.zebra.com/us/en/products/rfid/rfid-handhelds/rfd40.html (21.01.2023.)

[6] https://www.zebra.com/us/en/support-downloads/software/demo/123rfid-mobile.html

[7] Signal Coding in Physical Layer Separation for RFID, Tag Collision, Yi Li1, , Haifeng Wu1, and Yu Zeng1

[8] https://www.zebra.com/content/dam/zebra_new_ia/en-us/manuals/rfid/rfd40/rfd40-prg-en.pdf (15.01.2023.)

[9] F. Xia, L. T. Yang, L. Wang and A. Vinel, "Internet of Things", International Journal of Communication Systems, Vol. 25, pp. 1101-1102, 2012.

[10] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. AlonsoZarate „A Survey on Application Layer Protocols for the Internet of Things", Transaction on IoT and Cloud Computing 2015.

[11] O. Jukić, I. Špeh and I. Heđi, "Cloud-based services for the Internet of Things", Proceedings of the 41st International Convention MIPRO 2018, pp. 407-412, MIPRO, Opatija, 2018.

[12] N. Pavković, D. Marjanović, N. Bojčetić, Programiranje i algoritmi, skripta II, Zagreb, 2005.

[13] https://aits-tpt.edu.in/wp-content/uploads/2018/08/DS-UNIT-5.pdf (16.02.2023.)