University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Cloud Counter (C-Count) 2.0

Cepilov Ile
Cugnasco-Gerra, Ticino, Switzerland
Student ID: 13-929-740

Supervisor: Bruno Rodrigues, Christian Killer, Simon Tuck, Prof.Dr. Burkhard Stiller
Date of Submission: August 10, 2021

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland

Independent Study
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Zusammenfassung

Bei akuten Gesundheitskrisen wie COVID-19, bei denen soziale Distanzierung und sichere Belegungsraten von größter Bedeutung sind, ist die Analyse von Bewegung, Kapazität und Engagement von entscheidender Bedeutung für die strategische Planung von Live-Marketing-Initiativen für Einzelhandelsflächen, Fachmessen und Werbeveranstaltungen. Lösungen, die die Einhaltung gesetzlicher Vorschriften gewährleisten und wertvolle Analysen liefern, können nicht immer das erforderliche Mass an Genauigkeit für sichere Belegungsmessungen bieten. Durch die Kombination mehrerer Datenquellen lässt sich jedoch die Genauigkeit der Verfolgung erhöhen. In Anbetracht dessen wird in dieser Arbeit eine Lösung vorgeschlagen, die eine solche Analyse durch die Integration bestehender passiver Erfassungsansätze von 3D-Kameras mit Eingaben von RFID-Tags erweitert. In einem Versuch wurde eine Gruppe von Personen gefragt, sich in einer kontrollierten Umgebung zu bewegen, um die Wirksamkeit dieser Methode in realistischen Szenarien zu testen. Die Ergebnisse zeigen, dass die implementierte Lösung in der Lage ist, Personen innerhalb einer kleinen Gruppe gleichzeitig zu erkennen und ihnen die richtigen IDs zuzuordnen.

Abstract

During acute health crises such as COVID-19, where social distancing and safe occupancy rates are paramount, the analysis of movement, capacity, and engagement are of critical importance in the context of strategic planning of live marketing initiatives for retail spaces, trade shows, and promotional events. However, solutions that ensure compliance with regulatory requirements and produce valuable analytic cannot always provide the required level of accuracy for safe occupancy measurements. Nevertheless, by combining multiple sources of data, it is possible to increase the tracking precision. Hence, this thesis proposes a solution that expands such analysis by integrating existing passive sensing approaches from 3D cameras with inputs of RFID tags. A group of people has been asked to walk in a controlled environment in order to test the effectiveness of this method in realistic scenarios. Results show that the implemented solution is capable of simultaneously recognizing and properly assigning IDs to individuals within a small group.

Acknowledgments

Ack

Contents

Zusammenfassung **i**

Abstract **iii**

Acknowledgments **v**

1 Introduction **1**

1.1 Motivation 2

1.2 Thesis Goals 2

1.3 Report Outline 3

2 Related Work and Background **5**

2.1 Related Work 5

2.1.1 ID-Match: A Hybrid Computer Vision and RFID System for Recognizing Individuals in Groups 5

2.1.2 RFID Vision-Based Indoor Positioning and Identification System . 6

2.1.3 Optical Localization of Passive UHF RFID Tags with Integrated LEDs 7

2.1.4 Computer Vision-Assisted 3D Object Localization via COTS RFID Devices and a Monocular Camera 7

2.1.5 Mandigo 7

2.1.6 Livealytics 9

2.1.7 VCARE 9

2.1.8 Sensalytics 9

2.1.9 Axper 10

2.1.10 Cloud Counter 1.0 10
2.1.11 Ipsos 10
2.2 Background 11
2.2.1 RFID technology 11
2.2.2 Characteristics of a UHF RFID tag 14
2.2.3 Impinj Speedway Reader & Connect Software 15
2.2.4 Timeseries: InfluxDB 15
2.2.5 Xovis 3D Camera 17
2.2.6 Milesight IoT 18

3 System Design **19**
3.1 Requirements Specification 19
3.2 System Overview 20
3.3 Architecture Overview 21
3.4 Data Collection and Processing Overview 23

4 Implementation **25**
4.1 RFID Readers Setup 25
4.1.1 Ursalink/Milesight Router 25
4.1.2 Speedway Connect Plugin 27
4.1.3 Impinj Reader Modes 30
4.2 C-Cloud 2.0 Backend 32
4.2.1 Languages and Frameworks 32
4.2.2 Design Patterns 37
4.2.3 Data Collection Process 40
4.2.4 ID Matching Process 43
4.2.5 Environment and Deployment 48

5 Evaluation **53**

5.1 Comparison of RFID Reader Modes 53

5.2 Controlled real world test scenarios 55

5.2.1 Single RFID Tag 56

5.2.2 Two RFID Tags 58

5.2.3 Multiple RFID Tags 58

5.2.4 Visitor counts: 3D Camera vs. RFID Reader 60

5.3 Discussion and Findings 62

5.3.1 Matching solution and the RFID reader accuracy 62

5.3.2 Synchronization Offset on the RFID Reader 62

5.3.3 Measuring Statistical Dependence 63

6 Final Considerations **65**

6.1 Future Work 66

Bibliography **67**

Abbreviations **71**

List of Figures **71**

List of Tables **75**

Listings **77**

A Evaluation Results of Reader Modes **81**

B Installation Guidelines **85**

C Contents of the CD **87**

Chapter 1

Introduction

Analyzing consumer behavior is one of the most important methods of identifying positive and negative features in products and services. A company needs to identify how movement, capacity, engagement and their outcomes can be essential to the planning of proactive marketing initiatives in retail stores, trade shows, and promotional activities. In an increasingly digitized society, there are secure methods of visualizing an audience's mobility based on non-invasive, privacy-preserving methods that would enable the extraction of Key Performance Indicators (KPI) for the efficient planning of marketing strategies of business events or campaigns, which improves the offering of a product or service to a particular type of audience. For example, tracking a person over a defined area such as a public transport system [1] or a university campus [2, 3, 7] could give the tracker essential information about the individuals. This information can be used for crowd control, for city planners [2], for emergency services, for lowering energy consumption of buildings [3] and also business owners who can gather marketing information which then are used to tailor advertisement to future customers [8].

Concerning digital products or services (*e.g.,* e-commerce) offered to utilize the Internet, such extraction of information is relatively easily performed by tracking each movement of consumers (*i.e.,* click) within a website and establishing (KPIs), based on his/her browsing behavior. Using the number of times a given page is accessed, and the time a customer spends browsing that page, one can estimate the customer's interest in a product. Taking into consideration products and services offered in real life, however, this kind of analysis becomes more challenging. It is relatively straightforward to track consumer behavior in the digital world, but in the physical world (based on methods that preserve privacy and do not encroach on consumers' rights), it can be nearly impossible to track consumer behavior accurately. The privacy of consumers must be protected, in particular, when tracking methods require them to actively interact with a product or service, which would expose their identity (*e.g.,* documents or face) to being profiled, ripping away their right to privacy. In this sense, there are methods based on the capture of wireless signals emitted by mobile devices that allow tracking individuals in a non-invasive way [4, 5].

1.1 Motivation

As an essential component of the strategic planning of live marketing initiatives in retail spaces, trade shows, and promotional campaigns, the analysis of movement, capacity, and engagement is of even greater importance during acute health emergencies as COVID-19, where issues of social distancing and safe occupancy rates are paramount. Therefore, it becomes imperative that technical solutions can ensure compliance with regulatory requirements and provide valuable insights and analytics to restore the normal flow of business and society [6]. Using various wireless protocols such as 802.11b or 802.11g (2.4 GHz) or 802.11a (5 GHz), Livealytics provides an analytics solution based on mobile devices. Although this solution provides the ability to analyze visitor behavior, it lacks the accuracy necessary for safe occupancy measurements (i.e., calculating distance in real-time and issuing alerts relating to social distance). The Cloud Counter (C-Count 2.0) project is designed to address this problem based upon existing solutions. C-Count 2.0 extends the previous version of C-Count (C-Count 1.0), which implemented a customizable API for the collection, aggregation, and analysis of additional counting implementations, with the analysis and matching of RFID data. Therefore, the proposed solutions can be used with existing Livealytics and C-Count 1.0 products to perform highly accurate visitor flow analysis, valid occupancy counts (with unique visitor counts), and behavioral maps that enable regulatory compliance, assure acceptable social distance, and provide strategic and real-time business intelligence.

1.2 Thesis Goals

This thesis aims at developing a solution that collects, stores, processes, and matches data from 3D cameras of third parties with data from RFID tags. In order to accomplish the objectives of this thesis, it is necessary to understand the theoretical and conceptual elements involved at an early stage, as well as the core concepts of passive wireless tracking, including RFID tags. The second stage of C-Count 2.0 is targeted at creating unique identification links between the RFID tags and tracked objects in a unique and easy-to-use manner.
To increase C-Count's tracking accuracy, camera path information should be analyzed along with RFID tag tracking data. In addition, since RFID tags are not associated with any personally identifiable data, the approach is compliant with the GDPR regulations, thereby maintaining privacy.
In terms of research, the objective is to improve quantitatively the overall accuracy of C-Counts by incorporating RFID tracking into the existing system. It is necessary to develop the engineering component of C-Count that handles the capture of data from the RFID tracker through APIs provided by the manufacturer of the devices as well as the incorporation and analysis of this data in the system.
The third and final stage involves the development and evaluation of the proposed work. In accordance with the design decision outlined in the second stage, these elements will be integrated and evaluated in a modular fashion.

1.3 Report Outline

This report is divided into the following sections: **Chapter 2** provides a summary of existing solutions currently available on the market along with background information of the tools and hardware employed in this project. **Chapter 3** discusses the requirements of the proposed solution and provides an high-level architecture overview. In **Chapter 4** the implementation details of the application are discussed in detail. In **Chapter 5** the proposed solution is evaluated in terms of both usability and overall back-end performance and reliability. Finally, **Chapter 6** concludes this report by summarizing the findings.

Chapter 2

Related Work and Background

2.1 Related Work

This chapter presents an extensive discussion concerning different solution aspects of movement, capacity, engagement, and occupancy measurements. All the presented solutions below tackle the goal of exploiting gathered data from RFID readers or 3D Cameras to provide a better understanding and insights of events. Since this work is an extension of *C-Count 1.0*, some parts are based on and have been taken from it.

2.1.1 ID-Match: A Hybrid Computer Vision and RFID System for Recognizing Individuals in Groups

The work presented in [10] aims at bringing a novel hybrid computer vision and RFID system that is capable of identifying individuals (based on their RFID tags) along with their 3D images as captured by a depth camera. Specifically, the authors claim that the system can determine an individual's identity within four seconds with an accuracy rate of 96.6% and identify groups of five individuals within seven seconds with a 95.6% accuracy rate.
The ID-Match system can robustly identify people at sufficient speed and accuracy to allow a humanoid autonomous robot to interact with up to five individuals at once naturally. Furthermore, the system is effective at passively monitoring both tagged and untagged participants without requiring active participation for identification and tracking.

Their approach consists of correlating the time traces of the free-roaming synthetic apertures of the worn RFID tags (using a single RFID reader antenna) with the position traces of people (using a single depth camera). They have employed an Impinj Speedway Revolution UHF RFID reader with a single antenna and a Kinect v2 depth camera. Based on a support vector machine, their new technique is further enhanced because it correlates the changes in the low-level parameters of RFID channels, such as (RSSI and Phase), as the tags are moved in space, with the motion of the individuals as observed by the depth camera. In the final phase, a probabilistic voting system assigns IDs to the individuals in

the scene.

It should be noted that the ID matching system's chief strength originates from the fact that IDs are associated based on similarity ranking between a limited number of possibilities rather than being based on the true ability of the RFID reader to locate people and tags. To do this, they measure small variations in the tag and body motion of individuals as they walk to differentiate among them. Although these motion differences are difficult to visualize, they are statistically distinct.

In situations where multiple people are walking as part of a group toward the ID Match System, they create many dynamic events in which the one in the foreground can visually obscure (or possibly RF obscure) people in the background. A second issue is that since the RFID SAR motion paths are a measure of distance rather than an indication of position over time, there is a potential for non-uniqueness in the motion paths, which leads to difficulties in the matching process.

Occupancy Monitoring Scenario: In the same project, a second uncontrolled study examined ID-Match's capability to passively track the movement of individuals in an office as they pass through virtual checkpoints and gates. During an extended period, the system must recognize an individual's identity while visible to the camera and determine how they are moving.
A total of 122 of the 129 tags worn by participants were recognized correctly by the ID-Match system (94.5% accuracy). Five incorrect identifications were made for people who did not wear identification tags, resulting in a false positive rate of 2.9%, while the rest of the population was correctly identified as not wearing identification tags. Moreover, as a result of the Kinect's 3D skeleton tracking, the system recognized the direction of movement of the correctly identified skeletons with a 100% degree of accuracy.

2.1.2 RFID Vision-Based Indoor Positioning and Identification System

A new system combining active RFID and vision (called RV-based) positioning and identification is presented in the paper [11], which by cross-referencing video images with RFID position information improves positioning accuracy and supports personnel identification and positioning based on ID identification functions of RFID.
The RFID positioning system involves arranging several active RFID tags in the scene, then using a mobile reader to read the tags in the scene and analyzing the signal strength levels of all tags present in the scene. A database is used to store the positioning blocks based on analyzed data. As soon as the initialization is complete, a mobile Reader receives the signals of active tags present in the scene. The signals are addressed immediately, and received signals are compared with signal strength information in the database, thus realizing the initial positioning of RFID.

Through the use of a web camera, real-time images are captured for visual analysis. Positioning information of moving personnel is derived by analyzing real-time images, which are then converted into absolute coordinates of personnel in the scene through coordinate transformation. Following, RFID positioning results are converted to absolute positions

corresponding to the camera vision results, and finally, the individual's information is merged, thus, completing positioning and identification.

2.1.3 Optical Localization of Passive UHF RFID Tags with Integrated LEDs

The method presented in [12] utilizes an enhanced RFID tag that is augmented with an LED which can be flashed upon command by a reader in order to determine the location of passive RFID tags in uncontrolled and unstructured environments. Unlike traditional RFID location methods, their passive solution does not suffer of multipath problems and provides greater position accuracy than other solutions. Using the LED enhanced tag, people can more easily locate tagged objects that can be addressed individually using an RFID reader.

2.1.4 Computer Vision-Assisted 3D Object Localization via COTS RFID Devices and a Monocular Camera

In the research done in [13], RF-MVO, a novel system combining RFID and computer vision to enable stationary RFID localization in 3D space by parallelizing a monocular 2D camera and two reader antennas are presented.
Based on 2D images, the existing monocular visual odometry can only reconstruct the camera/antenna trajectory in the camera view. The combination of this model with the RF phase allows estimating the scale factor for real-world trajectory transformation, along with the direction of the RFID tag concerning the virtual antenna array.

The authors then present a novel RFID localization algorithm that does not require an exhaustive search of all possible positions within a predetermined area. In a second step, they propose an optimization algorithm that can improve the localization accuracy and speed up the process of searching between the results. Finally, the study introduces the concept of horizontal dilution of precision (HDOP), which is used as an accuracy metric to evaluate the degree of confidence in the localization decision.

2.1.5 Mandigo

In the field of system integration, Mandigo is a company that has particular expertise and makes it a unique player in its field [34]. The company's solution makes it easy for businesses to demonstrate the business value they can generate from trade shows and events straightforwardly. According to the company, they can keep track of the number and movements of their clients and visitors in real-time. Therefore, employing this monitoring facilitates the estimation of how many customers and visitors per square meter of space are required.
Regarding live tracking, they offer two variants:

1. **Registration of the customer via RFID technology using a chip in the shopping cart or ticket:**
   With this variant, an entry and exit control are carried out with the help of normal shopping trolleys or entrance tickets. Assuming that every customer has a shopping cart/ticket, this is recorded and counted using RFID technology.
   This allows the movement profile, location, and length of stay to be recorded and displayed. This data is available in real-time, and the evaluation of the data is interesting from a marketing perspective. In this way, visitor flows can be measured, and less frequented stands can be optimized.

2. **Via mobile tracking via the customer's cell phone:**
   The WLAN tracking records the position of the individual cell phones of the customers; their whereabouts and length of stay are determined and evaluated in a movement profile in real-time. For this mobile variant, WLAN routers are used. The configuration takes place fully automatically, and the system is ready for use within a few hours. This data can be called up online at any time via a monitoring dashboard. Since only the cell phones are identified and not their owners, their solution is General Data Protection Regulation (GDPR) compliant.

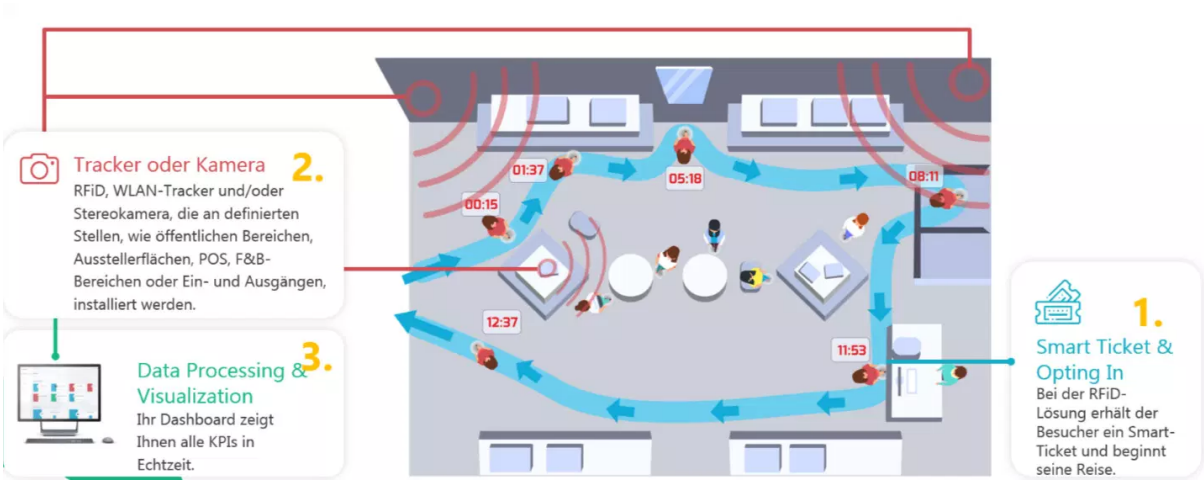

Figure 2.1: Mandigo: example of visitor flow measurement [34]

As part of their RFID solution, they have developed a chip called **bibchip UHF** [35]. The system employs a barrier-free ground antenna and it reads the data coming from the chip with the corresponding timestamp. A4 papers containing the bib numbers are given to the participants as tags with the information on them. By using an RFID reader that is attached to the ground antenna, it is possible to read the information on the back of the bib number. At every intersection, data is provided in real-time.
As a further alternative, the company provides a solution that makes use of a different frequency spectrum, i.e. High Frequency. This solution is known as **bibchip HF** and is used in indoor events where UHF technology is not suitable. With the HF technology, antenna gates are placed along the length of the measurement line.

2.1.6 Livealytics

Livealytics is a Swiss company located in Zurich that offers IoT solutions and analytics to companies in retail, live marketing, smart buildings, smart cities, and many other industries. In addition to improving live customer experiences, optimizing operational costs, and increasing sales, they can enhance live customer experiences with their tracking and measurement technologies. As an added feature, Livealytics provides an option for counting visitors entering a store or an event location and providing a comprehensive overview of the total number of visitors.

2.1.7 VCARE

In order to increase traffic volumes in a business, VCARE is a solution designed by V-Count, an industry leader in creating systems for collecting and analyzing visitor data. The V-Count 3D hardware enables VCARE to track and manage customer traffic anonymously, which helps it satisfy government social distancing restrictions for COVID-19 by ensuring compliance with social distancing restrictions. Furthermore, it allows to display occupancy recommendations and accordingly display optimal queueing and wait times for customers as part of the solution. For example, if a store occupancy limit is exceeded at any time, the employees will immediately be notified by email or push notifications [33]. In spite of this, this solution does not yet support the calculation of distance and the analysis of visitor paths in real-time.

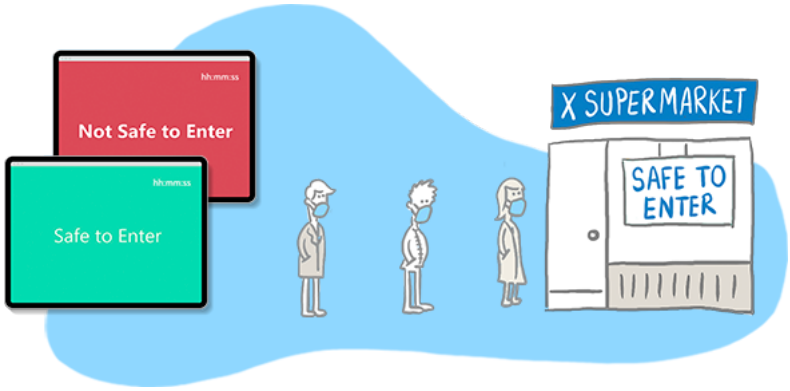

Figure 2.2: VCARE occupancy recommendation [33]

2.1.8 Sensalytics

Sensalytics is a German company that has been in the business of tracking real-world events since 2014. It is currently offering a variety of services, such as people counting, path analytics, and occupancy management. The Sensalytics platform uses highly accurate 3D sensors from third parties to produce accurate heat maps and calculate dwell time for each individual visitor [29]. Furthermore, Sensalytics also offers a revenue tracking tool that combines people counting data with revenue data. There is, however, no support for real-time distance calculation.

Figure 2.3: Sensalytics path analytics [29]

2.1.9 Axper

People counting and tracking solutions are among the core competencies of Axper. Similarly to V-Count and Sensalytics, Axper employs 3D cameras to provide reliable and accurate people counting systems. With Axper's 3D cameras, which are powered by artificial intelligence, it is possible to differentiate between children and adults in the count of people. The system also provides precise measurement of dwell time and tracking time [31]. Besides that, the company is not offering any distance calculation services at the moment.

2.1.10 Cloud Counter 1.0

A customizable API (Application Programming Interface) is provided by *C-Counter 1.0* for collecting, aggregating, and analyzing additional counting solutions or behaviors, such as distance calculation. A web-based application is used to provide extremely accurate visitor flow analysis, safe occupancy distances depiction, and behaviour maps for ensuring acceptable social distancing. Additionally, dwell time, a key component of evaluating how appealing a web site is, is collected using non-invasive and privacy-preserving methods in real-life scenarios.

2.1.11 Ipsos

Founded in France, Ipsos is a company whose goal is to support customers in the interpretation and application of data collected. Nowadays, the company offers innovative solutions in the field of retail performance, such as people counting, queue management systems, behavior insights, etc. Their solutions allow customers to increase revenues in retail sectors by presenting tailor-made promotional offers and anticipating the needs of their clients. Ipsos claims [30] to have an accuracy of 95%, which they consider higher than other competitors who promise 98% accuracy because their accuracy is calculated in a live environment in which lightning, flow, dwell, and loitering are taken into account

during the validation of the data. In similar fashion to others, their solutions rely solely on cameras or WiFi sensing methods where a close to real-time distance calculation is not possible.

2.2 Background

2.2.1 RFID technology

RFID stands for Radio Frequency Identification and it refers to a technology where digital data is encoded in RFID tags and captured by one or multiple reader through radio waves. RFID belongs to a group of technologies called auto-ID and it enhances them by allowing tags to be read without line of sight and, depending on the type of RFID, having a read range up to 20+ meters [37].
The main advantages of RFID technology are efficency, flexibility and robustness. Since data acquisition can be performed without any human intervention, it makes them very efficient. As mentioned, the tags do not need to be withing optical reach and therefore, the placement of tags on objects can take place with fewer limitations thus improving the degree of flexibility and lastly, RFID tags are very robust and are not damaged by humidity, dirt and they will usually be replaced because of newer versions. One of the first applications of RFID technologies was to identify planes during World War II [38]. In the meantime, as the technology evolved and improved year by year, the costs of implementing and employing RFID systems continued to decrease, making it more accessible and cost-effective. Thus, their reliability and effectivness has been proven in many industries to perform multiple tasks such as:

- Supply Chain Management
- Asset tracking
- Inventory management
- Controlling acces to restricted areas
- Personnel and people tracking

Topologies of tags

The RFID tag is the main component of the whole system as it allows to read and write data contained in it. Tags can vary in shape, size, material and operating frequency but they can all be divided into three different groups: Active, Passive and Semi-Passive or Semi-Active tags.

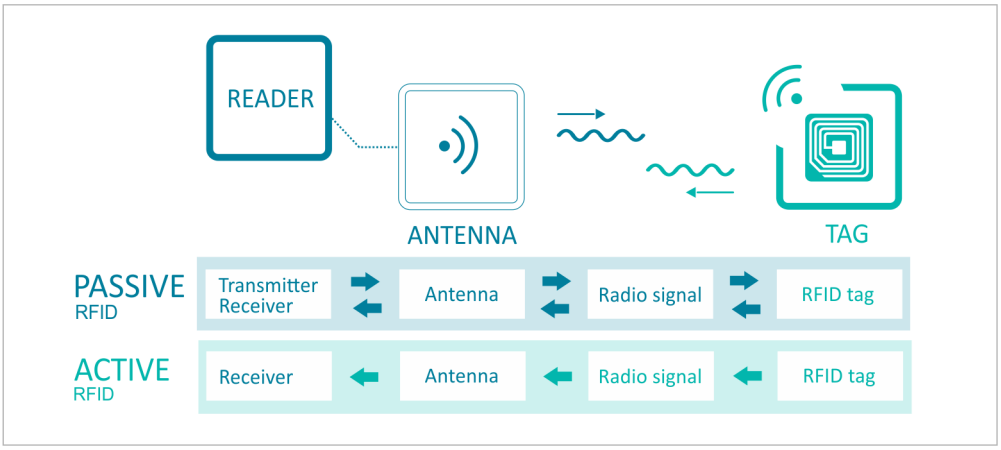

Figure 2.4: Active and passive RFID tags [36]

Active tags: they receive energy from their own power supply in order to operate, which usually comes from a long-lasting battery. They are able to autonomously transmit data to the reader and cover far greater distances than passive tags. They have large memory, are often re-writable and can contain sensors to take measurements, such as heat, temperature and pressure. They are usually designed for durable use.

Passive tags: they do not have their own energy source but receive it from the signal coming from the Reader. They consist of a chip (with a unique identifier and memory), an antenna and a support or container. As the reader passes, the radio frequency activates the microchip inside the tag, providing the energy needed to operate. This energy during the reading phase will be reused to respond to the reader by transmitting a signal with all the information stored inside it; in the writing phase, however, it will allow to save the data sent by the reader. The distances in which they can operate are of the order of a few meters or a few centimeters depending on the operating frequency.
Passive tags can be divided into **Near-Field RFID** and **Far-Field RFID**, depending on the frequency band used to communicate and the electromagnetic phenomenon. In the near-field region, the interaction between the components is dominated by the magnetic field generated by the antenna, which induces an electric current in the tag by inductive coupling and allows the chip to be activated. Tags of this type are part of the Low Frequency (LF) and High Frequency (HF) classes.
Whereas in the far-field region, the interaction of the components is dominated by the electromagnetic (EM) field created by the antenna. The RFID tag resonates with the frequency of the EM field and the current generated activates the chip. Tags of this type are part of the Ultra High Frequency (UHF) class.

Semi-passive/Semi-active tags: they are equipped with their own energy source which is used to power the microchip or other devices (such as sensors) but not to power the transmitter. To be able to transmit information, they must be within range of the reader/antenna.

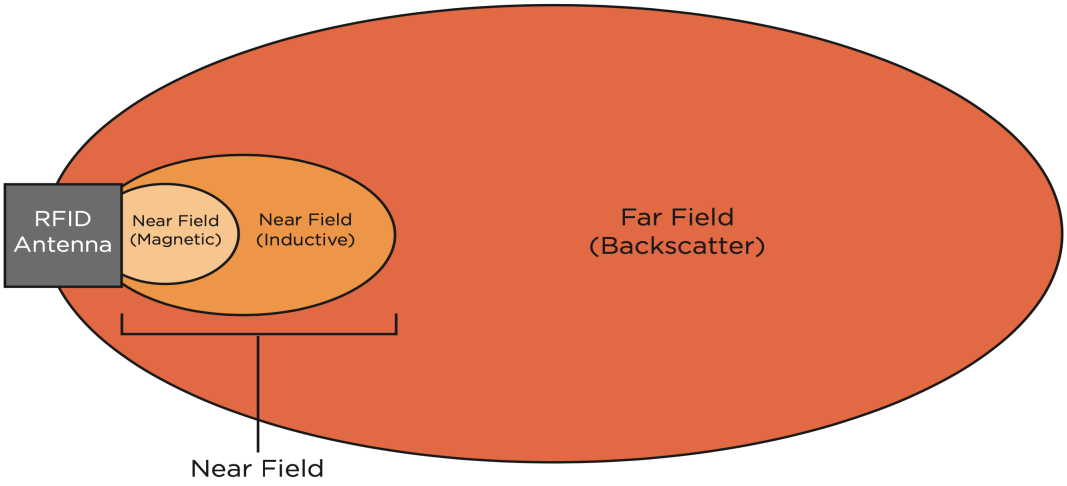

Figure 2.5: RFID Far-Field and Near-Field.

Operating frequencies of RFID tags

RFID tags can be further classified based on the frequency of the signal used for communication. These frequencies depend on the nature of the tags and the intended applications. They are regulated by international and national organizations. The frequency bands most used in RFID technology are the following:

Low Frequency Band: it is located in the lower part of the RF spectrum, within the LF band there are two most used operating frequencies: 125 Khz or 135 Khz. Historically it was the first one to be adopted and it still has a significant presence on the market today. Characterized by a very limited reading distance, it operates almost in contact, and given its characteristics it is ideal to be used for anti-theft, attendance, access control, traceability of animals and etc.

High Frequency Band: they operate in the frequencies that work at 13.56 Mhz, they are supported in the NFC (Near Field Communication) standards. Nowadays, it is considered as the universal band, given its possibility of use all over the world. It can be used for identification and tracking, pallets, access control and other applications that require a short reading distance.

Ultra High Frequency Band: they are frequencies that work at 868 MHz or 915 MHz, normalized in the Electronic Product Code (EPC) reference standards. The use of this frequency may be restricted by the authorities of individual countries in terms of maximum power and frequency bands. In fact, the band is not assigned uniformly in the US, Europe and Asia. Tags of this type have a greater reading distance than the others and are therefore mainly used for retail, logistics, warehouse management, etc.

	PASSIVE			ACTIVE
FREQUENCY	LF	HF & NFC	UHF	UHF-µW
	125/134 KHz	13.56 KHz	860-915 MHz	433 MHz-5.8 GHz
READ DISTANCY	max. 50 cm	max. 1 m	max. 15 m	max. 300 m
TAGS	small, cheap, easy to produce			more expensive own battery

Figure 2.6: RFID frequencies [36]

2.2.2 Characteristics of a UHF RFID tag

A UHF RFID tag is composed of an antenna and a Integrated Circuit (IC). The antenna is unique to the type of tag and its task is to receive the RF waves, energize the IC and finally backscatter the modulated energy to the RFID sensor.

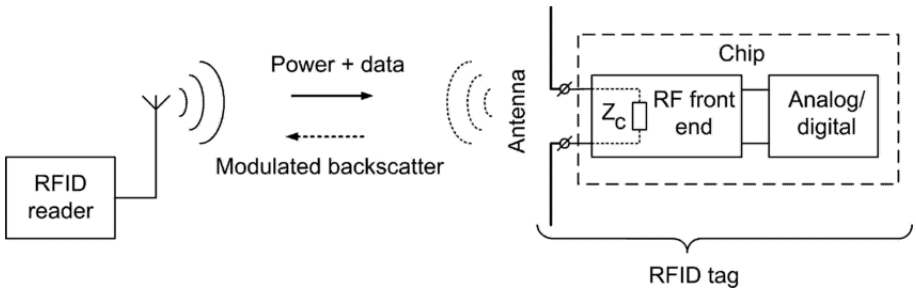

Figure 2.7: Passive RFID tag backscatter [15]

The IC contains four memory banks, it processes, sends and receives pieces of information and it is provided with an anti-collision protocol. Each IC is unique and the main difference betweens ICs is the number of bits provided in the respective memory banks.

The four memory banks are as follows:

Reserved Memory Bank: this memory bank contains the passwords relating to particular functions such as lock or access. The lock password is rarely used and is used to completely disable the transponder. The access one, on the other hand, is used when it is necessary to enable or disable the possibility of writing to the tag.

EPC Memory Bank: this memory bank contains the EPC code of the tag which can vary in length from 96 to 496 bits. This code is used as an identification code in most RFID applications and it is usually a randomized unique number.

TID Memory Bank: this memory stores the unique identification code of the tag (Tag Identifier). This bank cannot be written or rewritten by anyone. The code is entered directly by the manufacturer and no one can change it.

User Memory Bank: if the tag does have a User memory bank, it can be used for all those applications where it is not sufficient to use the EPC code and therefore this bank is used to enter further information to be stored and coupled to the tag. The smallest memories are usually from 32 bits up to a capacity of 64k bits.

2.2.3 Impinj Speedway Reader & Connect Software

Impinj is a company that manufactures RFID devices and softwares. The company designs, manufactures, and distributes UHF RFID chips, readers, antennas, and software applications. The Impinj Reader xArray R680 has been used to analyze movement, capacity, and engagement of people in this study. xArray gateways are RFID readers that provide continuous, wide-area monitoring of RFID tags. The system was designed for large-scale applications such as in retail, healthcare, and manufacturing, and a near-real-time identification, location and authentication of each item is provided. They claim that a single reader can cover an area of up to 139 square meters.

Using the **Speedway Connect** software, data can be transmitted from the readers to a backend. Speedway Connect is a licensed software that runs on a reader and gateway, enabling users to connect RFID technology easily and enhance its data collection capabilities through providing a solution for many different types of applications.
A HTTP (Hypertext Transfer Protocol) post request may be made to a remote server at regular intervals. This feature allows a `Reader Name` to be specified so that the reader can be uniquely identified. While the reader is powered on, the software is able to hold approximately 5'000 tags in memory if the connection drops or there are network problems. Upon reconnection, the reader will automatically send saved data to the specified destination.

When the reader profile is set to `Location`, in addition to the tag's EPC, the tag's x, y location is computed and reported. The location accuracy is typically within 1.5 meters. In `Direction` mode, the direction of movement of up to 50 tags passing by or under a gateway is determined. Since the model xArray is in use, up to 4 direction of travel can be computed (North, South, East, West). The Speedway Connect software provides an heartbeat feature which allows to monitor the devices status regardless if there are any tags in the field of view of the reader. Useful when it is necessary to know whether the devices is online and the connection is correctly configured.

2.2.4 Timeseries: InfluxDB

To guarantee the highest levels of efficiency and performance, Industry 4.0 and the field of the Internet of Things have led to an increasingly urgent need for data analysis and

Figure 2.8: Impinj xArray R680

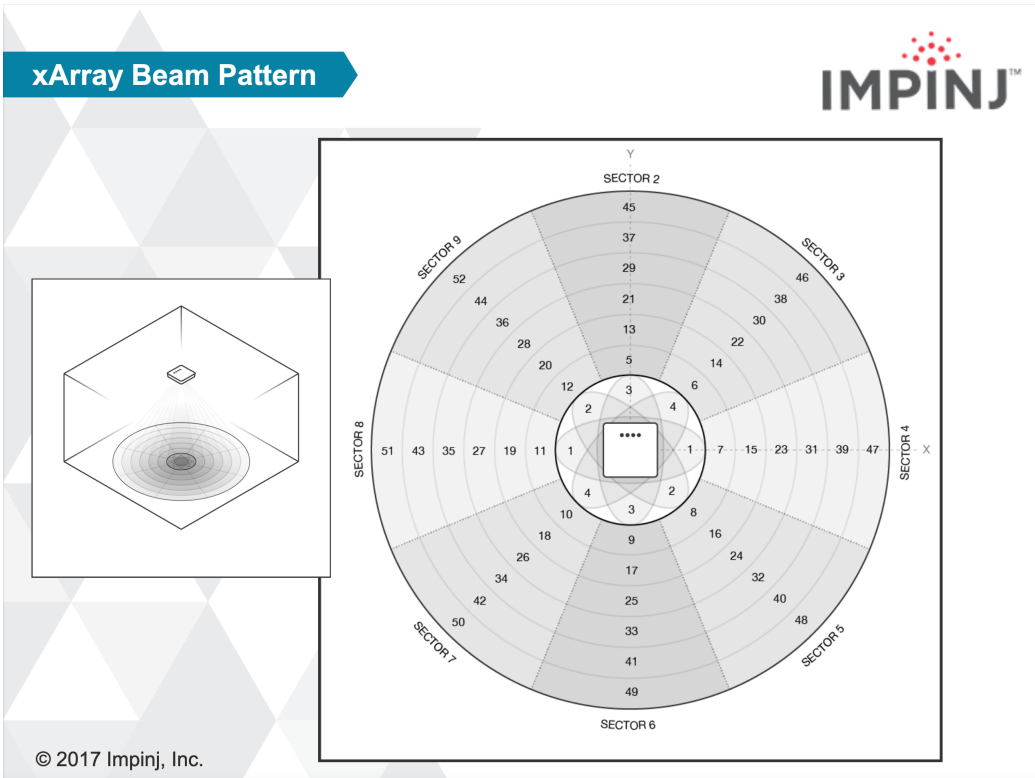

Figure 2.9: xArray beam pattern

monitoring, which in the IoT field are characterized by the presence of the timestamp on the data points, the Time Series Data.
It is not the timestamp itself that makes monitoring data in the IoT complex, but rather the amount of data generated and the frequency of monitoring to which they are subjected to. For all these data, the temporal dimension means that the data acquires value even after acquisition and must be kept over time for analysis and monitoring purposes. This type of data, however, accumulates easily. It is therefore necessary to have a performing database that allows to transform these data flows efficiently.
The key difference between a time series of data and simply adding a time field to a data set is how updates are tracked in the data set. In the IoT field, in fact, the data collected by a sensor cannot be overwritten as it is generated if you want to be able to monitor the status of the system over time.

Theoretically, any type of DB can be used for storing historical data series, but problems arise when it comes to performing analyzes on them. Given the nature of historical series data sets, the choice of database must be based on scalability and high availability. Relational databases do not provide sufficient scope and storage capacity for Time Series Data, which are linked to fast processes and for which it is essential to adopt a DB solution created specifically for their analysis.
Choosing a Time Series Database for historical data is the most appropriated if there is the need to optimize bandwidth and memory resources: these databases use the data collection timestamp (the moment in which it is read) as primary key and are optimized for applications that focus on the scanning of points detected over a long period of time.

Among the most performing Time Series Databases there is InfluxDB, which is an open source DB specialized in high writing and consistency of events, even on distributed system. It is designed to simplify the interaction with time series. InfluxDB 2.0 introduces the new programming language Flux, which the InfluxData company publishes as open source with the MIT license on GitHub.

2.2.5 Xovis 3D Camera

With regard to analysis of movement, capacity and engagement of people, Xovis 3D Camera are employed in this work. Xovis 3D cameras are equipped with a 3D sensor and two wide-angle lenses which are able to perceive the scene from different perspectives, therefore achieving a precise depth map or 3D image of the whole scene. Thanks to deep learning-based algorithms, they are able to recognize and track people allowing to analyze individual path of each person.
As stated by the producer [32], the 3D stereo vision technology is resistant to external influences like shadow, light changes and still able to pull off a counting accuracy of more than 99%, exceeding the conventional people counting and tracking technologies. The exact model utilized in this work is the `Xovis PC2`. It offers data processing on the sensor itself and thanks to the real-time data feature, it enables immediate traffic counting and zone occupancy. Moreover, in order to meet privacy requirements, it offers four privacy protection levels.

Figure 2.10: Xovis PC2 3D camera

2.2.6 Milesight IoT

Milesight IoT, previously known as Ursalink, is a high-tech leading-edge company which provides IoT solution that enable the connectivity of "things" to the cloud. Thanks to their products, the complexity of data collection, storage and retrieval can be simplified into the cloud. In order to provide connectivity to the 3D Xovis cameras and Impinj xArray RFID readers, both the `UR32` and `UR35` LTE Router have been used. They are industrial cellular router which provide 2, respectively 5, fast Ethernet ports with Power Over Ethernet support, thus simplifying the installation of the devices. Moreover, Milesight offers a VPN solution that allows remote access and configuration of attached devices and a DeviceHub for remote management of the routers.

Figure 2.11: Milesight IoT UR32 and UR35 routers

Chapter 3

System Design

This chapter illustrates the developed system and method to improve the overall level of accuracy of C-Counter based on the combination of inputs from 3D cameras with the tracking of RFID tags. Improving the level of accuracy of the previous system (C-Count 1.0) is essential in order to be able to guarantee precise level of occupancy measurement in periods of a pandemic or crisis such as the COVID-19.
Hereafter, Section 3.1 describes an initial requirement specification by determining functional and non-functional requirements. In section 3.3 an high-level architecture overview of *C-Counter 2.0* is given. It describes the process of collecting, storing the data from RFID reader and also the integration with the existing architecture of *C-Counter 1.0*.

3.1 Requirements Specification

As a mean to provide a detailed and documented specification of what a software is supposed to achieve and how it should perform, a Software Requirements Specification (SRS) can be established. A SRS is a selection of requirements, which describe the system to be develop as well as its objective.
A set of use cases that describe all the interactions that users will have with the software is included as well. Use cases are known as functional requirements, whereas non-functional requirements, provided as well, are requirements that impose restrictions on design or implementation and quality standards.
Well-specified requirements are needed to reduce development effort and costs, thereby increasing productivity. When done correctly, SRS ensures that the system's objectives are efficiently and effectively reached.
In partnership with Livealytics, some aspects that need to be addressed in advance have been identified.

Functionality. The *C-Counter 2.0* back-end should be able to collect, aggregate and store data coming from multiple Xovis 3D cameras, thus being backwards compatible with the existing system. Moreover, the new solution should be able to calculate the total number of visitors in scenarios where 3D cameras are installed and configured and extract the

number of unique visitors where RFID readers are present.
The system should be as flexible as possible and require almost no configuration so that the installation process together with 3D cameras do not require almost any manual input. Since the extrapolation of unique number of visitors will be based on a statistical data analysis, the desired level of accuracy and the confidence level can be dynamically defined.

External interfaces. The *C-Counter 2.0* makes use of third-party hardware and therefore it heavily relies on them. The data collected and used comes from two completely different technologies and sources and those devices are made by different producers. Therefore, to guarantee a proper and successful integration, the devices have to be reliable and accurate.

Quality. As mentioned before, to evaluate the performance, usability, security and scalability of a software, non-functional requirements are usually used. Those can be labeled as quality attributes as well. Since *C-Counter 2.0* is an extension of the previous version, the same non-functional requirements defined for it must be respected, *i.e.,* C-Counter should not take longer than 5 seconds to perform any computation, even when the number of cameras increases over time. Beside that, the new part responsible for extracting the unique number of visitors, shall continuously run in the background and process data in near real-time without affecting any other component and avoid to reprocess data already analyzed.
With respect to security, the previously defined aspects are guaranteed in *C-Counter 2.0* as well, *i.e.,* data collected and process can only be accessed by authorized people and devices can only be configured and linked to a particular event by authorized people. In the new solution, a 3D camera is going to be running and collecting data from an event and a RFID sensor is going to be paired to it as well, thus increasing the mole of data generated. Therefore, *C-Counter 2.0* is implemented with Amazon Web Services as well and thus offered scaling features, such as Auto Scaling, can be put in place by increased workloads.

Constraints. The existing solution based solely on 3D cameras is already GDPR (General Data Protection Regulation) compliant. The linking of RFID tags with tracked object must therefore be compliant as well by not associating any personal information to RFID tags.

3.2 System Overview

For the system to be able to uniquely count people in a scene, it must be capable of correctly matching the data incoming from 3D cameras with the data collected by RFID readers. The challenge here is to identify which RFID Tag belongs to which ID assigned by the 3D camera, and thus to which individual. If a person who left the scene later returns, the system should recognize it and prevent it from being counted twice.
To accomplish this, a method for correlating the ID of the 3D cameras with the EPC of the RFID tags has been developed.
Figure 3.1 shows a typical implementation of *C-Counter 2.0* consisting of an Impinj xArray R680 UHF RFID reader along with a Xovis 3D Camera. Participants are supposed to wear or hold a low cost UHF RFID tag in form of a badge. As participants walk within

the view of the 3D Camera and the RFID reader, the RFID tags are continuously read and the 3D camera tracks the participant's position under it.

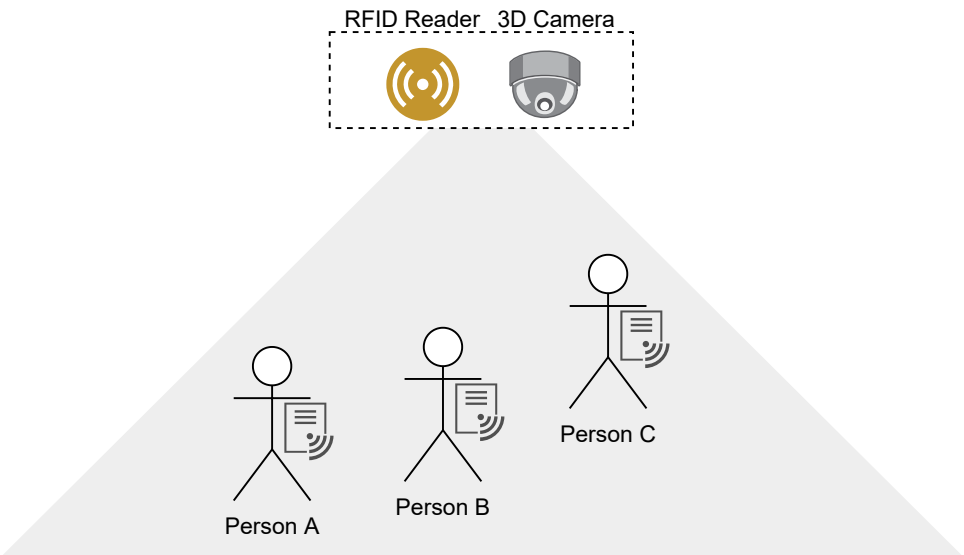

Figure 3.1: An illustration of a typical implementation of C-Counter 2.0 consisting of a 3D camera and a RFID reader

3.3 Architecture Overview

Figure 3.3 depicts the architecture overview of the system with its components. The red box illustrates the existing and unchanged architecture of *C-Count 1.0*. It is the front-end that manages the interaction with the user. This encompasses the selection of the desired camera, the associated event date, and a graphical representation of the dashboard associated with the selected event. Data is requested from the back-end and is processed as efficiently as possible by the front-end. The real computation occurs in the backend, which is hosted on AWS and based on lambda functions. Its serverless architecture built upon microservices enables it to easily scale when necessary while maintaining a high level of extensibility.

Due to the system's simple architecture, it is possible to incorporate RFID Readers into the system as easily as writing a new lambda function to collect the data and store it in a separate database. In fact, the RFID tag data is stored in an InfluxDB, which is superior to relational databases when it comes to storing and processing time-stamped measurement data. There is a cron job running in the background that gets triggered every five minutes. In this cron job, a python script is executed inside a lambda function that retrieves the most recent data from both the camera and RFID sensors, analyzes it, combines it, and stores the result of the computation inside InfluxDB. A new endpoint has been created in order to retrieve the data. The endpoint accepts any query written

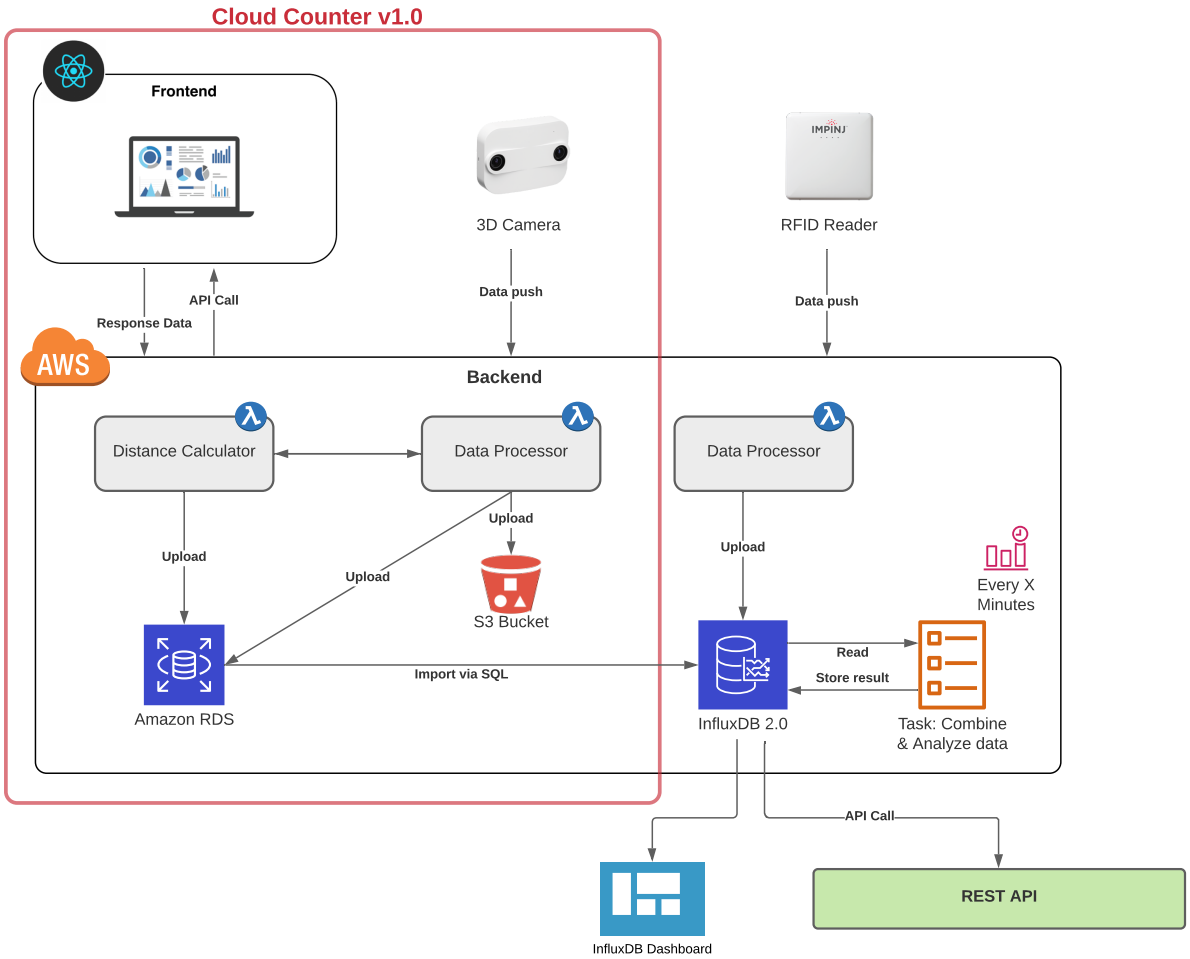

Figure 3.2: Architecture overview of *C-Counter 2.0*

in Flux which gets then executed in the database and returns the corresponding result. Furthermore, a dashboard in Influx has been created to monitor and visualize the collected data. It displays several metrics simultaneously in real time, for example the number of points collected for each RFID tag, the last known position of a tag on a scatter plot, and the most recent timestamp of the data for each RFID reader. The dashboard facilitates the debugging process.

3.4 Data Collection and Processing Overview

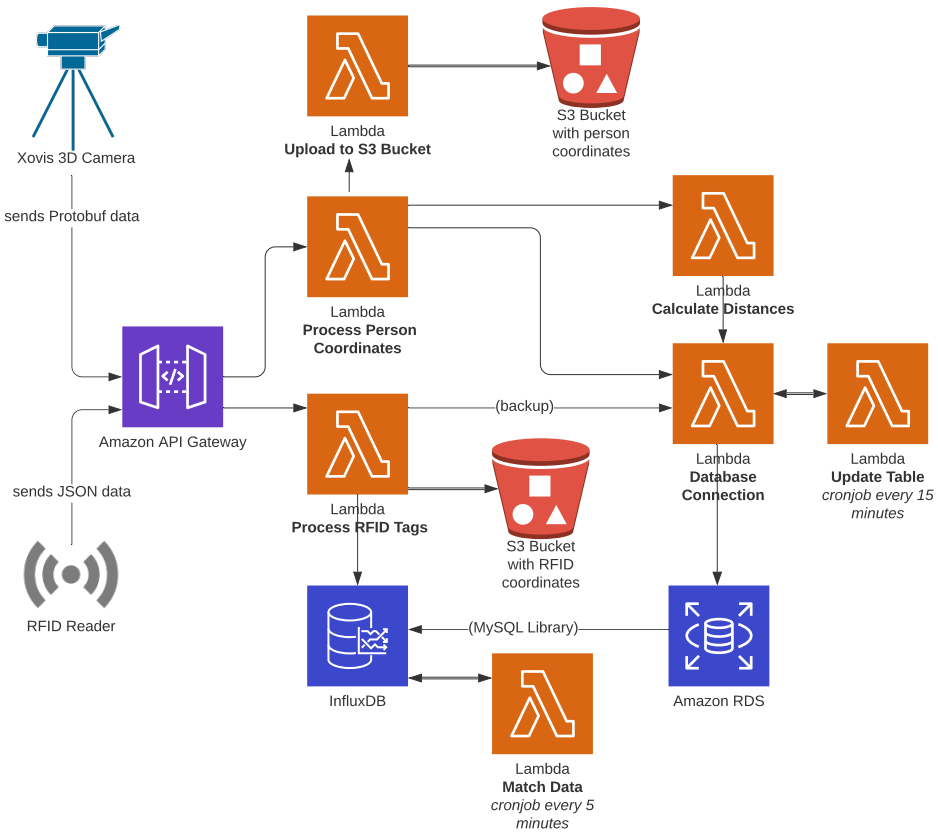

Figure 3.3: Data Collection Overview of *C-Counter 2.0*

An explanation of how data is simultaneously collected from cameras and RFID readers is presented in this section. Similarly to the previous version in *C-Count 1.0*, a camera sends data to a lambda function called `Process Person Coordinates` that is hosted by AWS. Additionally, a copy is forwarded to the second function (`Upload to S3 Bucket`), which stores the data in the AWS Simple Storage Service (S3), as well as a lambda function (`Database Connection`), which saves a copy into a relational database that is hosted in the cloud. Asynchronously, the same data is sent to a third lambda function (`Calculate Distances`). By analyzing the coordinates of the objects in the scene, this function calculates the distance between them. In the event that any points are closer than a predefined threshold, then the matching coordinates and the minimum distance

are saved in the relational database. For the purpose of aggregating the data coming from different camera data pushes and enabling a faster data retrieval, every 15 minutes, an Amazon Cloudwatch event rule is triggered. The data for the last three hours is retrieved, aggregated, and stored in a different table.

Due to the different structure of the data, another lambda function (`Process RFID Tags`) is used to parse the data appropriately and insert it into a time series database, in this case InfluxDB. In the same manner as before, the data is passed to a lambda function that stores it in a relational database and a copy is stored on Amazon S3. It is important to note that the data stored in the relational database is only used as a backup in case InfluxDB gets corrupted or any data loss occurs.
With regard to extrapolating the unique number of counts in a specific scene, a lambda function (`Match Data`) is triggered every 5 minutes. This function, written in Python, retrieves the data collected from the RFID readers and queries the corresponding data for the paired 3D cameras. Once the whole dataset regarding the last 5 minutes is available, the data is processed, analyzed and a match between both sources is calculated. In the event of a successful match, the matching entries are inserted into the InfluxDB, along with the matching metrics, which can be used to evaluate the accuracy of the system.

Chapter 4

Implementation

This chapter details the concrete implementation of the design proposed in the previous chapter. Firstly, in order to achieve the design goals, the configuration of the RFID readers and the Ursalink router are detailed followed by a close-up to the backend hosted on AWS and based on lambda functions. Secondly, the implementation details of the database powered by InfluxDB along with an Influx dashboard are given.

4.1 RFID Readers Setup

4.1.1 Ursalink/Milesight Router

Impinj's xArray R630 RFID readers are not equipped with mobile internet access. Thus, in order to simplify the installation of the readers, they are connected to a cellular router, the `UR32 or UR35`. These two models are industrial cellular routers with a built-in LTE CAT4 modem, and they support Power over Internet (PoE), so the RFID Reader does not require any additional power sources. Furthermore, the manufacturer of the routers provides a VPN network connection (MilesightVPN) to ensure that customers can communicate securely over the internet to remote devices. Using this feature, remote troubleshooting, viewing and control can be accomplished.

Figure 4.1 shows the official DeviceHub Portal software that allows the managing and controlling of the routers. It shows important pieces of information like the *Time*, the *Name*, the *Serial Number* and the *VirtualIP*, useful to identify and reach the routers while connected to the VPN service. When typing the VirtualIP of the router into the browser, the user is welcomed with the *Ursalink* portal. Besides showing the status of the router and the connected devices, the portal offers the possibility to configure the built-in firewall and map the required ports in the *Port Mapping* section. Figure 4.2 shows the configuration of the required port needed to remotely access and connect to the RFID reader, reachable under the internal IP **192.168.0.2** assigned by the router.

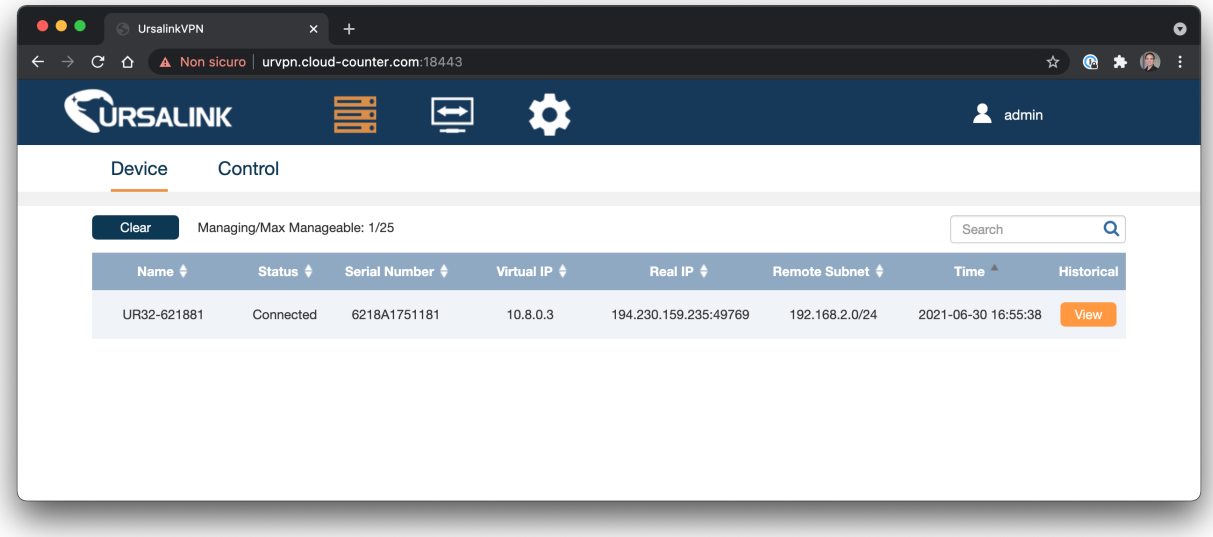

Figure 4.1: DeviceHub portal

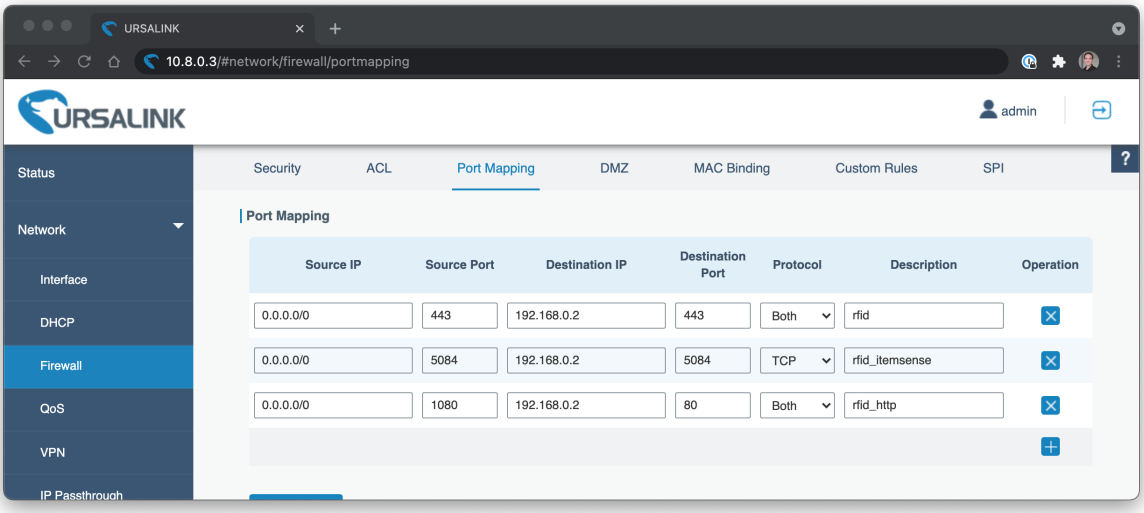

Figure 4.2: Ursalink port mapping

Port 80: this port is necessary to be able to remotely access the software running on the Impinj xArray R680 Gateway. Important to mention is that since the Ursalink Portal is already running on port 80, the source port (or the external port) has to be different, in this case the port **1080** has been chosen. Like shown in figure 4.3, the software displays the status of the reader but most important, the software allows to upgrade the firmware running on the reader and also the ability to reboot it when needed. Through this pannel, it is also possible to install other plugins, like the *Speedway Connect* software which allows the reader to send data using the HTTP protocol in JSON (JavaScript Object Notation) format.

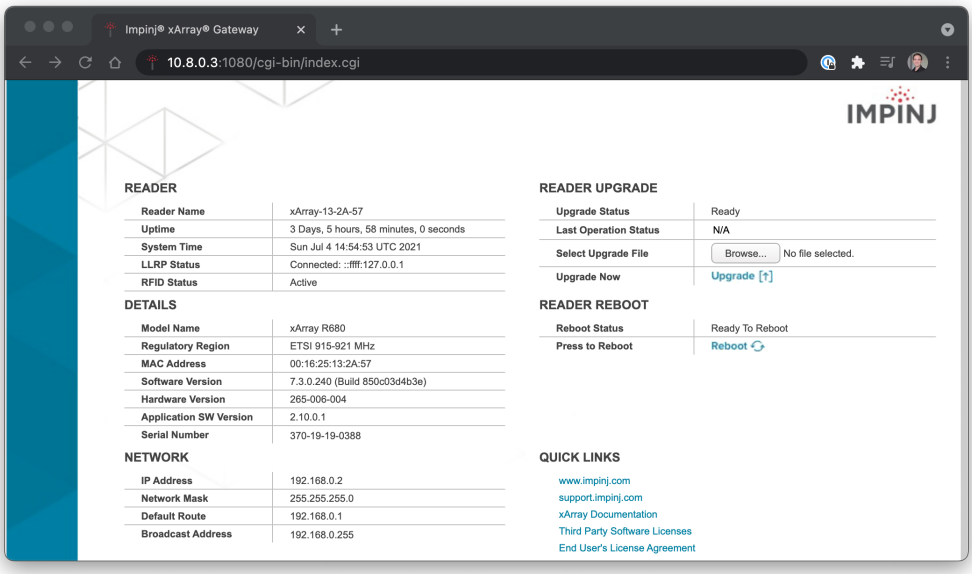

Figure 4.3: Impinj xArray gateway

Port 443: this port is used to connect to the *Speedway Connect* plugin which is reachable under the **HTTPS** protocol.

4.1.2 Speedway Connect Plugin

Speedway Connect is a software plugin that runs on Impinj speedway readers and enables users to deploy, manage and enhance the data collection capabilities for RFID technology. It takes advantage of a simple graphical user interface that allows users to change the reader's configuration and also deliver RFID data in JSON format using the HTTPS post protocol, without any software or middleware, such as a computer. The following section identifies the main capabilities associated with the Impinj xArray R680.

Reader Profile

- **Location**: for each tag's EPC it computes the x and y location of the tag and reports it with an accuracy of 1.5 meters [39]. This is the profile used to collect the data for this study.
- **Direction**: it can detect the direction of up to 50 tags passing by or under a reader. With an xArray it provides four directions: North, South, East and West.
- **Inventory**: it provides the timestamp when the TAG was latest seen, no additional fields or computation is performed.

Data Delivery Options

- The software is capable of storing the tag reads into a file on a USB stick and access them remotely by FTP.
- It can outputs tag reads to a TCP/IP Socket.
- Send the tag reads to a device connected on the serial port.
- Send a HTTP post request to a remote server on a regular interval. This last option was employed in this study. Additionally, the reader can be given a name (Reader Name) in the body of the message so that it can be distinguished from other readers sending data to the server.
  The tags information will only be sent if a tag has been recorded and the minimum update interval has expired. For backup purposes, it ensures that up to 500 tags are saved locally when an HTTP post fails, and attempts to resend them as soon as possible. Once the limit of 500 tags has been reached, any new tags will overwrite any older tags. [39].

Data Content Options: it outputs different pieces of information collected about the tag and sends only the ones included in the chosen reader profile, which could be:

- Antenna Port
- Timestamp
- Peak RSSI
- TID
- User Memory
- Heartbeat (it allows to monitor the reader's health even if no tags are present in the field of view).

Figure 4.4: Reader profile & location configuration

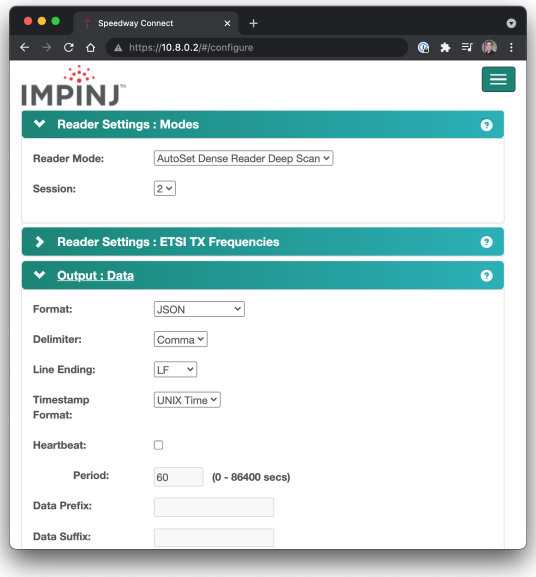

Figure 4.5: SWC: Reader modes data output

Figure 4.6: SWC: output connection

4.1.3 Impinj Reader Modes

Reader modes describe the data transmission over-the-air between the reader antenna and RFID tag. By changing the modulation, it is possible to choose best balance between read rate and resistance to interference by changing the modulation scheme that encodes the data.
In general, more modulations used to encode a bit, the less likely it is for the data to be affected by interference from other nearby devices or readers who happen to use the same frequency. Three examples, each sending the same information with a different amount of modulation, are provided below to illustrate how signals are modulated to encode binary data [41]. The Miller (M) algorithm determines the number of modulations used to encode each bit. An increased Miller value means more modulations per bit, which generally requires more transmission time but yields a more robust signal.

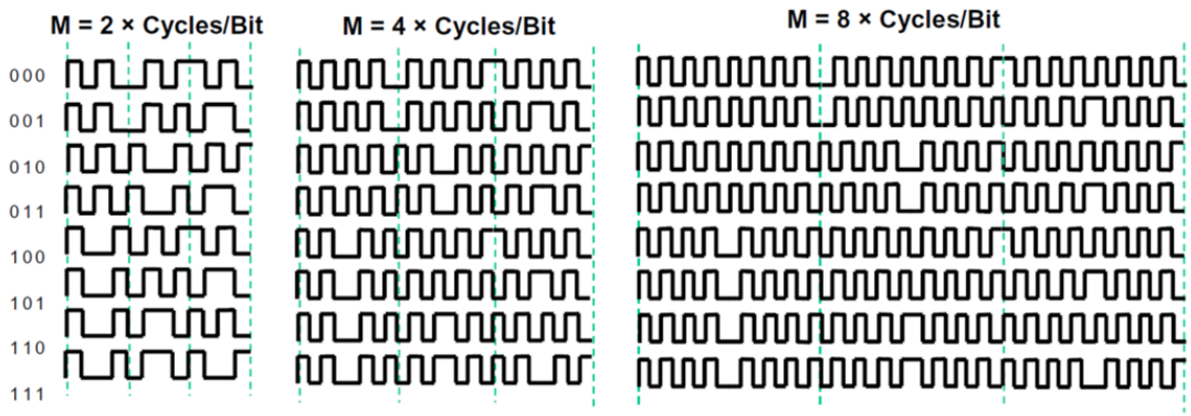

Figure 4.7: Levels of modulation taken from [41]

As part of Impinj's Speedway readers and gateways, there are a number of predefined modes to select from. Table 4.1 shows a brief overview of the main features of the various predefined reading modes.
By selecting a reader mode that has a high reading rate, it is possible to obtain the most accurate tag location calculation. In Location Mode, the greater the number of times the xArray reads the tag, the greater the number of samples the algorithm uses to determine the (x,y) tag position with reasonable accuracy. Figure 4.8 shows a flowchart made by Impinj which describes the best approach to take in order to chose the best algorithm depending on multiple factors such as the environment and the number of readers.

Table 4.1: Impinj Speedway RAIN RFID reader modes

Mode	Name	Type	Encding	Notes
0	Max Throughput	Standard	FMO	Fastest data rate, but most susceptible to interferences
1	Hybrid	Standard	Miller 2	
2	Dense Reader M4	Standard	Miller 4	
3	Dense Reader M8	Standard	Miller 8	Most interference tolerant of the standard modes.
4	Max Miller	Standard	Miller 4	Higher read rate with tolerance for multiple readers in the area
5	Dense Reader M4 Two	Standard	Miller 4	Higher read rate with tolerance for multiple readers in the area. Faster forward link than mode 2
1000	AutoSet Dense Reader	Impinj	Various	Listens to RF environment before selecting reader mode 1 to 5
1002	AutoSet Dense Reader Deep Scan	Impinj	Various	Combination of faster modes and slower Dense Reader Modes (DRMs). Maximizes the number of unique tag reads
1003	AutoSet Static Fast	Impinj	Various	Combination of faster modes only Higher read rate with some tolerance for multiple readers in the area
1004	AutoSet Static Dense Reader	Impinj	Various	Combination of DRMs only
1005	Impinj Internal	Impinj	Various	Do Not Use

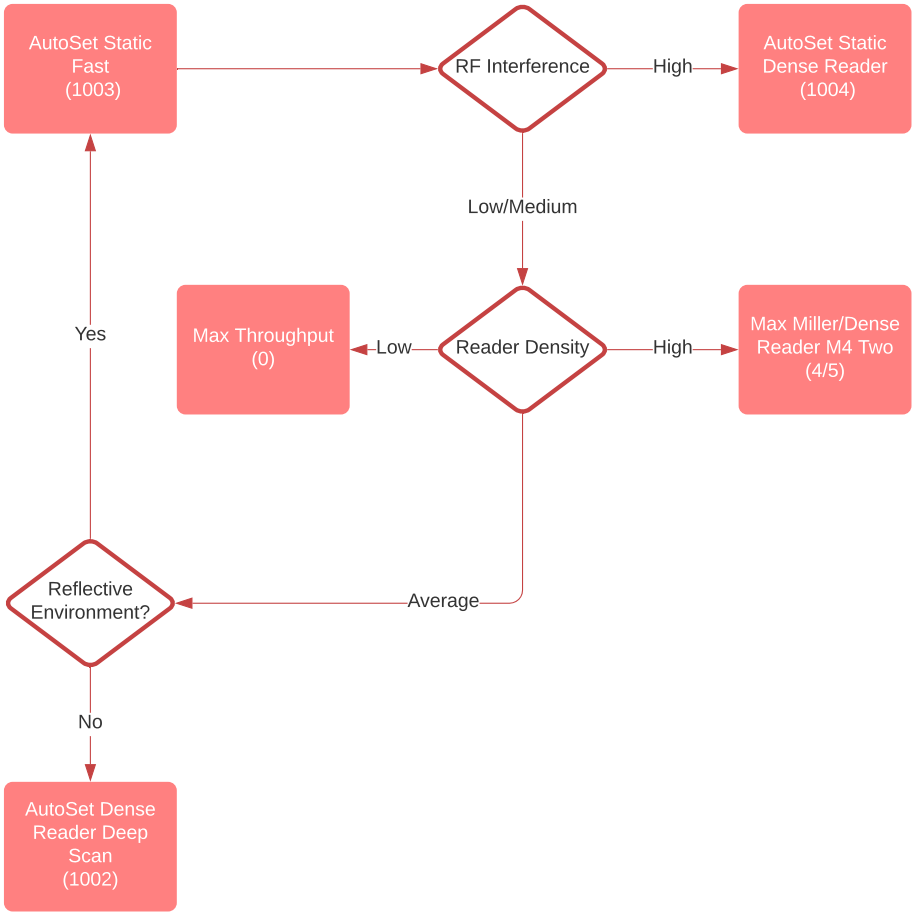

Figure 4.8: Location mode recomendation from [41]

4.2 C-Cloud 2.0 Backend

4.2.1 Languages and Frameworks

Amazon Web Services (AWS)

C-Counter 2.0 is hosted on Amazon Web Services, a cloud service provider. AWS offers many services, including REST APIs, automatic scaling of the computation power based on usage and security for applications. Considering that the Livealytics platform is already hosted on Amazon Web Services, it was justified to use AWS for the *C-Counter 2.0* as well.

AWS Lambda: Throughout the AWS Lambda function framework [18], a single microservice resides in a stateless container and is executed by a serverless compute service only when triggered by events fired by multiple clients. Due to their decoupled nature, lambda functions can be independently scaled based on the incoming network traffic. As soon as the code has been uploaded to a Lambda function, the service handles all the necessary capacity, patching, scalability and administration of the infrastructure to allow the contained code to be executed. Lambda functions also generate real-time metrics and logs that are published to AWS Cloud Watch [19] in order to enhance visibility of performances.
To access on of the Lambda function, a path, accessible through an API Gateway, has to be specified. For example, to retrieve coordinates of a camera the path could be */rfid/-coordinates/schlieren?readerName='center'*, where **/rfid/coordinates/schlieren** specifies the path and **?readerName='center'** the parameter. This path is usually appended to the API Gateway domain and thus accessible from the outside.

AWS CloudFormation Template: The maintainability of a scalable microservice-based architecture depends on its well-defined structure. In this respect, Amazon Web Services Cloud Formation [24] provides a common language for describing and provisioning the cloud-based infrastructure. Several templates can be modeled that contain declarative APIs, which are used as input data by the API Gateway to enable the provisioning of applications and services at a rapid pace. Upon creating the stack from the template code, it is possible to think of it as a collection of AWS resources that can be managed as a single entity within Cloud Formation. In some cases, stacks are useful for encapsulating multiple API versions into a single container, so that in the event that a certain API version is no longer needed, all of its associated resources are removed from the cloud by simply deleting the stack. By using Cloud Formation, all resources needed and deployed to the cloud are declared and standardized using one source of truth for the entire application.

AWS Relational Database Service: When dealing with a serverless architecture, the most common approach involves utilizing a multi-tenant relational database that is able to release and open multiple DB connections for different microservices. Amazon Relational Database Service (RDS) [59] offers a number of different databases and offers the capability of scalability in the cloud. For this study, an Amazon Maria database previously used by Livealytics for development purposes, was chosen. AWS MariaDB is a cost-effective

MySQL relational database built for the cloud which enbles fast lookups.
The exact version of MariaDB used is 10.5.

AWS CloudWatch: The Amazon CloudWatch service monitors applications and provides reports to developers [19]. A log file will be stored within CloudWatch after every request is processed by each of the Lambda functions. This log file contains vital information about the request just served. The log is essential when developing new features that are interacting with different services, and it is necessary to verify the functionality of the method called. Besides, it can be useful for the identification of bugs or the explanation of why certain actions occur. It is quite useful to check the log within CloudWatch to see what exactly happened with the request, for instance, if an RFID reader did not send any data or if the data could not be stored in the InfluxDB.

AWS S3: S3 is an Amazon Simple Storage Service [21] that offers an object storage service through a browser interface. In addition to being highly scalable, it is also relatively inexpensive. It is a type of storage that can be utilized to store any kind of object for backup, recovery, and data warehousing purposes.

InfluxDB

Developed by InfluxData, InfluxDB is an open-source time series database with optional closed-source components. The software is written in the Go programming language and is optimized to handle time series data. The older version of InfluxDB, namely version 1.8 and earlier support a SQL-like query language, whereas the newest version, 2.0, has its own query language, namely Flux.
The TICK Stack (See Figure 4.9) is a loosely coupled yet tightly integrated set of open source projects which provide a full time series database platform with a variety of services, including InfluxDB.
It can run on cloud and on premises on a single node as well. Closed-source versions, namely InfluxEnterprise and InfluxCloud, offer features including high availability, scalability and backup-and-restore.

Key Concepts: In order to fully understand InfluxDB, a few important concepts must be defined. Presented below is a simple example that illustrates these concepts in an easy to understand manner (See table 4.2). An example similar to this is presented in the InfluxDB documentation [42]. It illustrates the RFID tag `E28011700000021428BF80B1` sensed by the RFID readers `center` and `left` at location `schlieren` between 01/07/2021 at 11:00 and 01/07/2021 at 11:02.

Table 4.2: RFID data example

_time	**_measurement**	**filialeId**	**readerName**	**EPC**	_field	_value
2021-07-01T11:00:00Z	rfid_location	schlieren	center	E28011700000021428BF80B1	xCm	253
2021-07-01T11:00:00Z	rfid_location	schlieren	left	E28011700000021428BF80B1	yCm	120
2021-07-01T11:01:00Z	rfid_location	schlieren	center	E28011700000021428BF80B1	xCm	270
2021-07-01T11:01:00Z	rfid_location	schlieren	left	E28011700000021428BF80B1	yCm	170
2021-07-01T11:02:00Z	rfid_location	schlieren	center	E28011700000021428BF80B1	xCm	574
2021-07-01T11:02:00Z	rfid_location	schlieren	center	E28011700000021428BF80B1	yCm	198

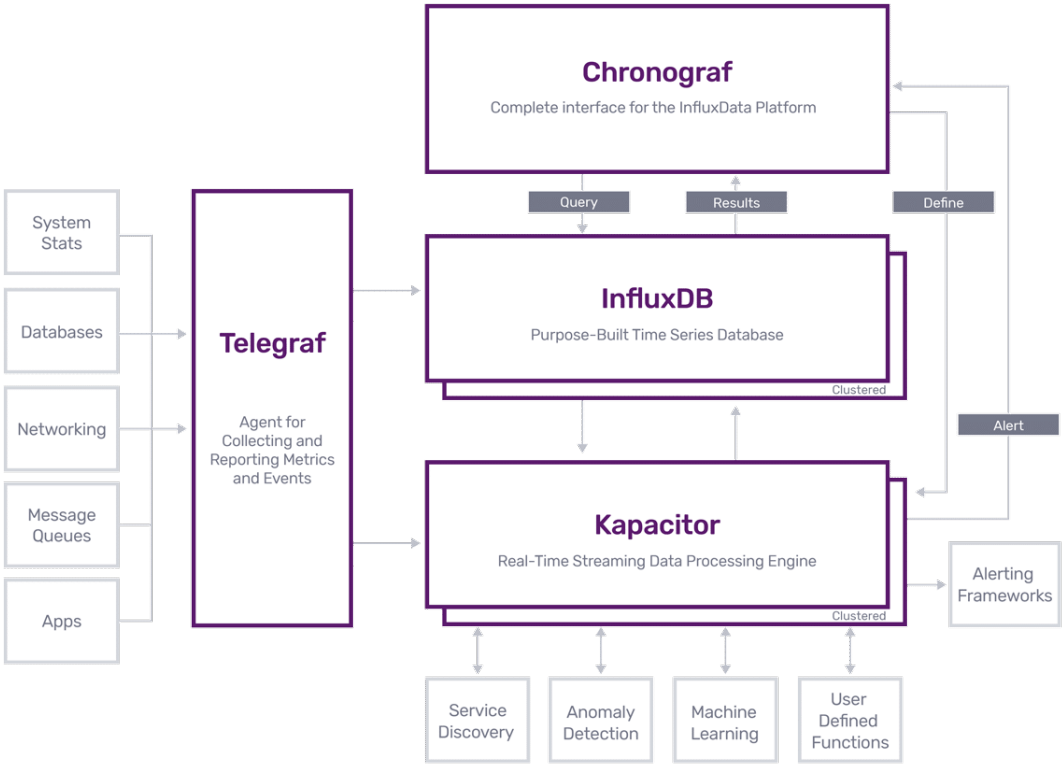

Figure 4.9: The TICK stack

First of all, time is the most important concept in InfluxDB. Each InfluxDB database contains a `_time` column that stores discrete timestamps associated with specific data. On disk, timestamps are stored in epoch nanosecond format. The `_measurement` column shows the name of the measurement `rfid_location`. Measurement names are strings. A measurement acts as a container for tags, fields, and timestamps (think of it as an SQL Table). The next three columns, `filialeId`, `readerName` and `EPC` are tags. Tags include tag keys and tag values that are stored as strings and metadata. The tag key `readerName` has two tag values, namely `center` and `left`. Fields are stored in the last two columns, the field key stored in the `_field` column and the field value stored in the `_value` column. Field keys are string that represent the name of the field, in this case `xCm` and `yCm` and field values contain the actual data like `253`, `120` and `270`. Field values can be strings, float, integers or booleans.
Field sets are collections of key-value pairs associated with a timestamp. The following field sets are included in the example data:

- `rfid_location xCm=253,yCm=120 1625137200000000000`
- `rfid_location xCm=270,yCm=170 1625137260000000000`
- `rfid_location xCm=574,yCm=198 1625137320000000000`

Tag sets are collections of tag key-value pairs which make up a tag set. The following tag sets are included in the example data:

- `filialeId=schlieren, readerName=center, EPC=E28011700000021428BF80B1`
- `filialeId=schlieren, readerName=left, EPC=E28011700000021428BF80B1`

A query on tags will be faster than a query on fields since tags are indexed. Consequently, tags are ideal for storing frequently accessed metadata. All InfluxDB data is stored in a bucket. A bucket is a combination of a database and a retention period (the period of time for which each data point is retained).

NodeJS

Most of the microservices running in the back-end are developed using Node.js [44], a well-known runtime environment for javascript that executes code and can be used to create fast and scalable network applications. Node.js runtime compilers from Lambda are used to execute code in response to cloud events. An environment that includes the AWS SDK for Javascript[43] runs the code itself.

Python

The Python programming language is an interpreted, object-oriented, high-level programming language with dynamic semantics [45]. Because of its high-level data structures, dynamic typing, and dynamic binding capabilities, the language is very suitable for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components. The simple syntax of Python, which is easy to learn, emphasizes readability, thereby reducing the cost of updating a program. Python supports modules and packages, which promote modularity and code reuse in programs. AWS Lambda supports the Python programming language. The Lambda framework provides Python runtimes that execute the code to respond to events. In the code's environment, Python SDK (Boto3) is installed along with credentials from an AWS Identity and Access Management (IAM) role that can be managed.
Python is often the programming language of choice for the daily tasks that data scientists handle, and among the top data science tools used across industries [46]. In the case of data scientists who need to integrate statistical code into production databases or integrate data with web-based applications, Python is often the way to go. Additionally, it is ideal for implementing algorithms, which is a task frequently performed in this field. Furthermore, there are Python packages tailored specifically for such functions, including pandas [47], NumPy [48], and SciPy [49]. Python's scikit-learn module can be extremely useful and valuable to programmers working on various machine learning tasks. As part of this study, Python is used to match the data collected from both sources, namely from cameras and RFID readers. A detailed explanation is given in section 4.2.4.

Flux

The Flux programming language is a standalone data scripting and querying language designed to increase efficiency and simplify code reuse. A planner and optimizer are incorporated into Flux to facilitate ETL (Extract Transform Load), monitoring, and alerting. In order to develop Flux, the open source community drove innovation around time series data [50]. With its easy-to-use interface and excellent readability, Flux is a highly productive and easy-to-learn program. The Flux scripting language has both a command line interface and a web-based interface. Due to the nature of Flux queries, they can be tested and checked into source control systems. A query can be tested in parts, and a complex query can be built from components tested individually. Composability is a feature of Flux. For specific use cases, developers can extend the language. It is possible to include other Flux modules in the code and add new functions to the platform and due to the nature of Flux queries, they can be tested and checked into source control systems. A query can be tested in parts, and a complex query can be built from components tested individually. Flux is designed with the capability of integrating with other systems in mind so that it makes it easy to integrate different data sources, such as databases, third-party APIs, and filesystems.

```
from(bucket: "cloudcounter")
|> range(start: 2021-01-01)
|> filter(fn: (r) => r["_measurement"] == "rfid_location")
|> keep(columns: ["_time", "filialeId", "readerName"])
|> group(columns: ["filialeId", "readerName"])
|> last(column: "_time")
|> group()
```

Listing 4.1: Flux query to get the lastSeen timestamp of all the RFID readers

As part of its chaining operations, Flux uses pipe-forward operators `(|>)`. Each of Flux's functions and operations produces a table or collection of tables containing data. These tables are piped forward to the next function or operation in which they will be further processed or manipulated. This allows sophisticated queries to be built by chaining together functions [54]. Listing 4.1 shows an example of a Flux query that finds the last timestamp present in the database for each RFID reader. The first step is to select the correct bucket, *i.e.,* `cloudcounter`, which corresponds to the database in MySQL. Flux always requires a range to be specified, so a timestamp older than the actual start of the experiment is used in order to scan the entire database. The filter function is used first to identify the correct table, *i.e.,* `rfid_location`, and then other conditions may be chained. The third pipe operator, *i.e.,* filter, selects only the desired column. Subsequently, the data is grouped by `filialeId` and `readerName` and only the last known value in the column `_time` is taken.

SQL

The Structured Query Language (SQL) is a standardized programming language used to manage relational databases and perform various operations on them. As an initial

creation in the 1970s [51], SQL is widely used not only by database administrators, but also by developers writing data integration scripts and data analysts looking to set up and execute analytical queries. Modifying database tables and index structures, adding, updating, and deleting data rows, as well as retrieving subsets of information from a database are various uses of SQL. The query language is a command language where SQL statements are commonly used for querying and performing other SQL operations such as select, add, insert, update, delete, create, alter, and truncate.

4.2.2 Design Patterns

Software design patterns are generalized, reusable solutions to a common software problem within a specific context. The design is not a finished product that can be converted directly into source or machine code. Instead, it is a description of how to solve a problem that can be applied to many different contexts. Programmers can use design patterns to solve common problems when developing an application or system.
Through the use of tested, proven development paradigms, design patterns can accelerate the development process. A successful software design takes into account issues that may not become apparent until later in the implementation process. As a result of reusing patterns, small problems can be prevented from becoming large problems in the future and improve readability for programmers and architects familiarized with the pattern. It is therefore important to have a well defined design pattern in advance of implementing a system.

Serverless and Microservice Architectures

Cloud computing services provided by Amazon Web Services are often offered in serverless context. This means that there is no operating system or server to maintain or configure and there is access to a wide range of computing power. In such a context, where resources can be automatically scaled when needed, flexibility is also a crucial factor. In addition, rather than paying a full price for a server that may not be fully utilized, AWS applies a pay-per-invocation billing model. In other words, the costs are based on the time spent during the execution of the code.
Microservice architecture is an architectural style in which an application is divided into multiple services. As a result, these services are independent from each other, loosely coupled, highly maintainable, well-defined and they can be exposed through an API endpoint. It is the API Gateways' responsibility to orchestrate the communication and to distribute the traffic among the services in response to demand. In our case, the fact that lambda functions can be automatically scaled or replicated ensures that, even under high demand, such as when many cameras and rfid readers send data to the backend or when many users access the *C-Counter 2.0* GUI, the system stays responsive.

Monolithic vs Serverless Architecture

The traditional application development paradigm is characterized by monolithic architectures in which processes are tightly coupled, wrapped, and exposed as a single interface. In the event that there is a demand for a single API endpoint, the entire instance must be replicated in order to scale the application. In addition to being costly, this approach may lead to a single error being repeated multiple times. As the code base grows, adding more functionalities to a single container could introduce more bugs [52].
The result is that it is challenging to introduce new features, maintain the current code base, and simultaneously prepare to manage more incoming traffic within one monolithic application. A serverless architecture can be constructed by using independent components that run each application process as a service. There is no doubt that these services are loosely coupled, well defined, encapsulated, independent of one another, and exposed as a single container or API endpoint.
Contrary to a monolithic approach, each service can be regarded as a reusable capability that requires as much granularity and abstraction as possible. An API Gateway orchestrates communication among these functions and distributes the incoming traffic according to demand. Serverless functions are isolated from each other and can, therefore, be updated, scaled and deployed independently. There is no need to replicate the entire application instance if the application has a spike in demand for a specific service. Separation of concerns [46] is a design principle that is followed in this architectural pattern and is intended to separate a orchestrated solution into distinct services, where each service should address a distinct concern. Figure 4.10 shows a typical example of a serverless architecture.

As well as their benefits, serverless functions such as AWS Lambdas have some disadvantages as well. It is not possible for them to be instantiated independently and they require a complex architecture, such as AWS Cloud Services which are typically subscription-based services. Moreover, most of the time, duplicated code will be found within the functions because they do not share a codebase. The file must be copied and pasted into each lambda function if it is referenced across multiple functions. If the file is modified at a certain point, it must be copied and pasted into each lambda function. If the file is not copied to every serverless function using it, or if there is a mistake on the copied file, this can lead to errors.
AWS introduced the concept of Layers to overcome this problem. A Layers is simply a file archive that can contain additional code or data and be shared between different lambda functions. However, Layers also introduces new tooling challenges and requires additional security considerations.
First of all, they are more difficult to invoke locally. It is difficult to execute functions locally when there are dependencies that exist only within the execution environment. Tools must be able to fetch relevant layers from AWS and include them in the build process before executing local functions. Prior to calling a function, the SAM CLI would fetch the layers and cache them and therefore, upon invoking the function, the content of the layers would be available in the container SAM utilizes [53]. It is also difficult to test Lambda functions that depend on layers. Unit and integration tests should mirror the behavior of SAM when testing the functions. It is possible to download the layers in question using either the AWS CLI or AWS SDK as part of the setup process. However,

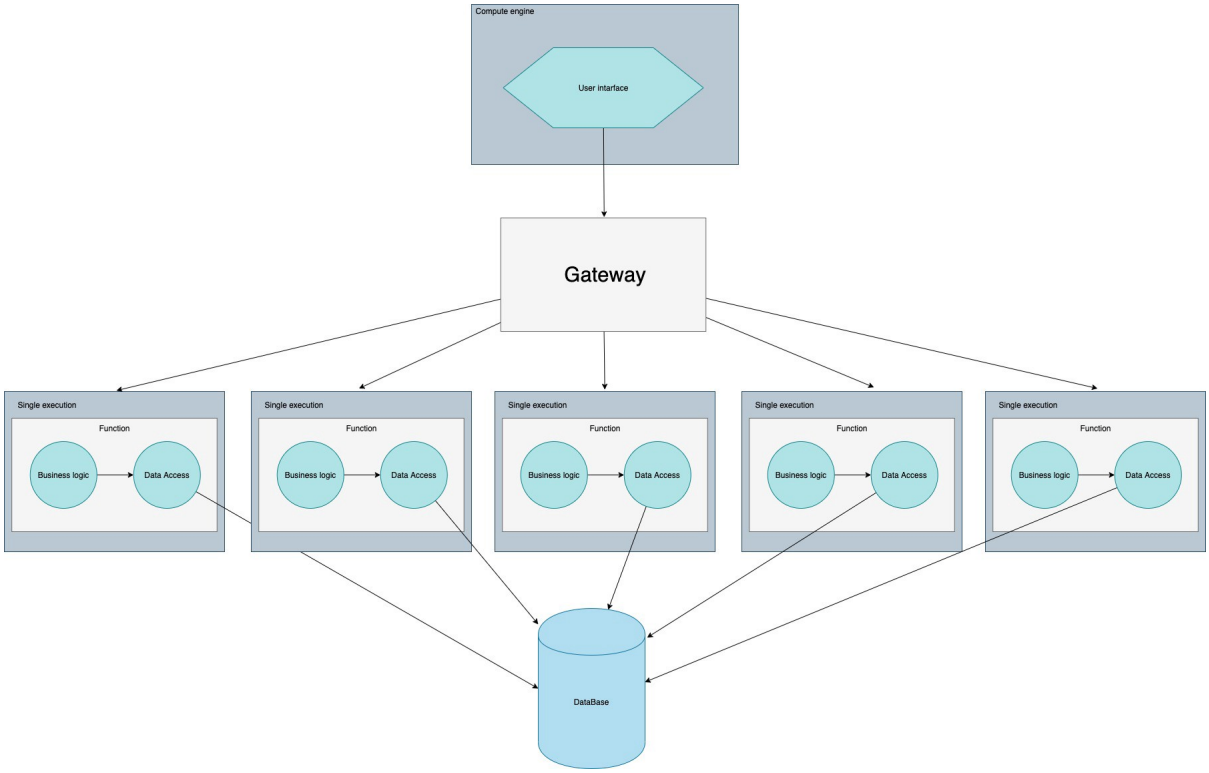

Figure 4.10: Typical serverless architecture taken from [52]

this adds a considerable amount of complexity to the tests. If testing the layers is not feasible, it might be possible to create a mock or stub. However, it is not the same as executing layer-specific handler code.

The second difficulty is dealing with changes in the layers. In the event that a layer is deleted, the functions that were deployed with it will continue to function. This function will, however, no longer be able to be updated. It is the case even if the layer is still available but not the specific version required. A potential problem may arise when a hotfix is required for a function in production. In the context of layers, this is a risk that should be considered, especially when dealing with layers controlled by third parties, where no control over their release cycle is assured. In addition, when applying layers, a good practice is to scan the layers for vulnerabilities and malicious code, thus requiring even more effort.

As a result, monolithic architectures are more easily implemented. The monolithic approach is quite effective in the early phases of a project, and most large and successful server-side projects that can be witnessed today were started using this method. For projects with extremely complex dependencies that change over time, serverless architecture is a better choice.

4.2.3 Data Collection Process

The purpose of the following subsection is to provide detailed information regarding the collection and storage of data. The configuration relies on the Xovis 3D camera and the Impinj RFID reader, both of which are discussed in details. The implementation on the back-end side is also thoroughly described.

Xovis 3D Camera

In *C-Counter 1.0* it has been mentioned that the Xovis camera can transmit data in various formats. The camera can automatically push data (count data, object coordinates, sensor status, and others) to a remote server by using HTTP(S), (S)FTP or MQTT. The data used in *C-Counter 1.0* is based on object coordinates. Data is pushed more frequently with an interval of 5 seconds, whereas the granularity of 1 second was chosen to correspond to the frequency with which RFID readers capture a Tag position. In this way, the process of matching the data between the two different data sources becomes easier and requires less data cleaning and preprocessing.

```
Xovis 3D Camera data format:
{
    "timestamp": 1600404912560,
    "id": "50185",
    "x": 95,
    "y": 112
}

RFID Reader data format:
{
  "epc": 'E28011700000021428BFA201',
  "lastSeenTime": 1625134778370154,
  "type": 'update',
  "xCm": -117,
  "yCm": 89,
  "confidenceWeight": 195,
  "confidenceData": [Array]
}
```

Listing 4.2: Xovis 3D camera data format and RFID reader data format in location mode

Impinj RFID Reader

A RFID reader must possess the location profile, as shown in figure 4.4. Listing 4.2 shows how an entry point captured by an RFID reader is formatted differently from that captured by the 3D camera, and therefore, a new Lambda function is required to correctly parse the data. The new function has the capability of parsing all three profiles

that can be sent out by an RFID Reader. This function parses the data according to its structure and then passes it to a second Lambda function that handles the connection to the database and inserts it to the database, similar to how it is handled in *C-Counter 1.0* with a relational database.

```
1  exports.lambdaHandler = async event => {
2      const {body, pathParameters } = event;
3      const {filialeId} = pathParameters;
4      const {tag_reads, reader_name} = JSON.parse(body);
5      await writeToInfluxDB(filialeId, reader_name, tag_reads);
6      return responseHandler(true, "OK", 200)
7  }
8
9  const writeToInfluxDB = (filialeId, readerName,tags) =>{
10     let measurement = `rfid_${getTagsType(tags)}`;
11     let points =
12         tags.map(fields =>{
13         return {
14             ...fields,
15             filialeId, readerName,
16         }
17     })
18     if (env === 'local') {
19         ...
20     } else {
21         //Invoke the InfluxDB lambda function
22         return lambda
23             .invoke(params)
24             .promise().then(({Payload}) => {...})
25     }
26 }
```

Listing 4.3: Parse RFID reader request in a lambda function

Listing 4.3 exposes the main functionality of the lambda function in charge of parsing the data sent from RFID readers. The first step is to parse the event passed to the lambda function from the API Gateway containing the RFID reader's request. In a second step, the `filialeId` parameter, which is sent through the query path, must be extracted and then the actual body of the data is parsed. The function `writeToInfluxDB` takes all the arguments and enriches the datapoints with the filialeId and readerName parameters. Finally, the data is sent to a lambda function that will be responsible for storing it in InfluxDB.

The core components of the Lambda function in charge of establishing a connection to the InfluxDB and store data into it are shown in listing 4.4. The official InfluxDB client [55] library written for nodejs has been utilized. In order to be able to establish a successful connection to the InfluxDB instance, an authentication token has to be generated and utilized. Moreover, a bucket and an organization have to be specified as well in the newest version of InfluxDB. Once those parameters are known, a client object can be instantiated as shown in `line 4`. In order to write to and retrieve data from the Influx

database, a queryApi and a writeApi can be instantiated from the client. In contrast to a SQL query, a clear distinction must be made between a get and a write query.

The function `writeQuery` shows how a `Point` has to be instantiated. According to the official documentation found in [56], a `measurement` and a `field set` are required, whereas a `tag set` and a `timestamp` can be optional. If no timestamp is provided, InfluxDB will assign a timestamp when storing the data. As synchronization problems related to the timestamps assigned by the RFID reader have been encountered during this study, the timestamps are assigned directly by the InfluxDB instance. A detailed explanation is given in chapter 5.

```
1  const token = 'XMy_RRgLyJYgD5t_...';
2  const org = 'cloudcounter';
3  const bucket = 'cloudcounter';
4  const client = new InfluxDB({url: 'https://prod.tsdb.livealytics.net
       ', token});
5  const queryApi = client.getQueryApi(org)
6  const writeApi = client.getWriteApi(org, bucket)
7
8  const writeQuery = async (data, measurement) => {
9      let points
10     if(measurement ==="rfid_location"){
11         points = data.map(d =>{
12             const {filialeId, readerName, epc, type, xCm, yCm } = d
13             return new Point().measurement(measurement)
14                 .tag("filialeId", filialeId)
15                 .tag("readerName", readerName)
16                 .stringField("epc", epc)
17                 .stringField("type", type)
18                 .intField("xCm", xCm)
19                 .intField("yCm", yCm)
20         })
21     } else if(measurement === "rfid_heartbeat"){
22         ...
23     }
24     writeApi.writePoints(points)
25     const writeApiPromise = await new Promise(async (resolve, reject
           ) => {
26         writeApi.close().then(result => {
27             resolve(result);
28         }).catch(err => {
29             console.error(`Error saving data to InfluxDB! ${err.
                   toString()}`)
30             reject(err);
31         })
32     });
33 }
```

Listing 4.4: Store RFID reader data in InfluxDB

Since queries performed against the InfluxDB primarily use or filter data based on the filialeId and readerName, those values are indexed by declaring them as `tag`, thus improving performances. The EPC, type, xCm and yCm values are stored as fields since they are only stored and no operation is performed on them directly into Influx.

4.2.4 ID Matching Process

As previously noted, in order to accurately determine the number of people in a scene, the system must be able to precisely match data incoming from 3D cameras with data collected by RFID readers. Identifying which RFID tag is associated with which ID assigned by the 3D camera, and so to which individual, is the primary task. It is important that the system recognizes the return of an individual who left the scene and prevents it from being counted more than once. For this purpose, a method for correlating the ID of the 3D cameras with the EPC of the RFID tags was developed.
C-Counter 2.0 accomplishes this by using a independent correlation pipeline running in the background. The procedure is shown with the help of a sequence diagram in 4.12 and covered step by step below:

1. Data retrieval: Initially, data concerning the RFID tags and 3D cameras will be gathered from InfluxDB and the Relational Database, respectively. As not all 3D cameras are paired to a RFID reader, the first dataset to retrieve is that of the RFID readers that contain the data collected within the last five minutes. Once the data has been successfully retrieved, only the data of the corresponding 3D cameras will be queried. This process is meant to avoid gathering unusable data and overload the pipeline.

2. Data pre-processing: Pre-processing of data is a critical step in the data matching process. As a result, the second step is to translate the camera coordinates into a reference system that is the same as that used by the RFID readers. A comparison of the reference systems used by the RFID reader and the 3D camera is shown in Figure 4.11. Afterwards, the timestamp is parsed from a string to a data object, so that the algorithm can understand and utilize the value. In addition, because the timestamp for the RFID tags is assigned in InfluxDB rather than on the devices themselves, a few seconds have to be accounted for and subtracted. In section 5 regarding the evaluation of the system, a detailed explanation is provided. Once the data has been pre-processed, the algorithm can proceed to the next step.

3. Datasets merging: Since the measurements are taken at very short time periods (*e.g.,* every second) by two different sources, when merging two dataframes some data points may be off by a second or more. Therefore, a smart way of merging two dataset is by using the built-in Pandas function `merge_asof`. Assume that when merging dataframes A and B, a row in the left dataframe (A) does not have a matching row in the right column dataframe (B), `merge_asof` will allow to take a row whose value is close to the value in the left dataframe (A). How close can be defined as well, in our case as *1 second*. If no match is found, the previous timestamp is taken.
Table 4.3 shows an example of a dataset containing information about 3D cameras, while table 4.4 displays the corresponding dataset containing RFID information. Table 4.5

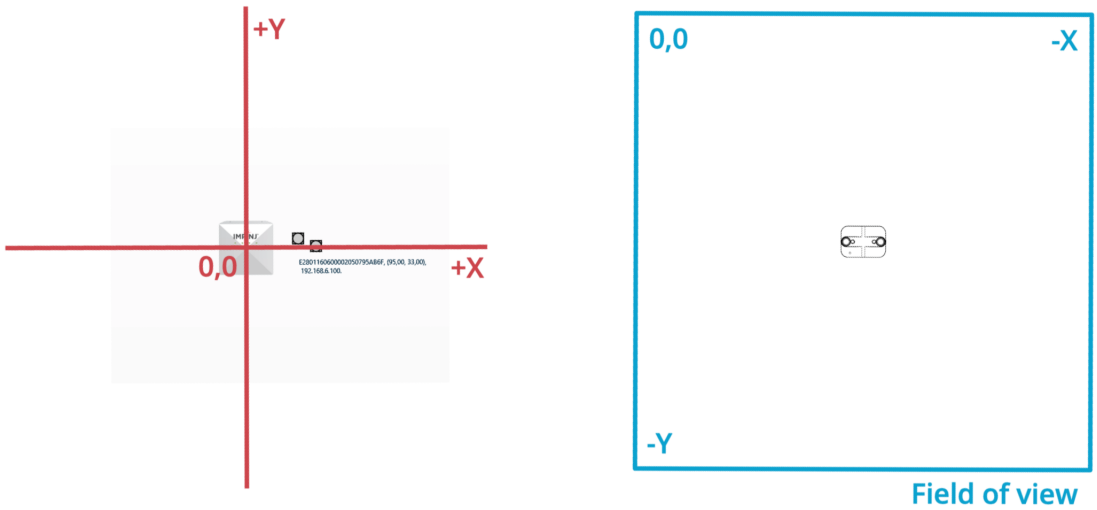

Figure 4.11: RFID readers and 3D cameras reference systems.

illustrates the result of the merge operation. The merge has been accomplished by running the following code:

```
pd.merge_asof(A,B,on="timestamp",tolerance=pd.Timedelta('1s'),direction='nearest')
```

In this specific instance, an exact match was found and thus no `NaN` value were present. Since `merge_asof` performs a left join between the two datasets, the final timestamp considered is the one from the 3D cameras, the timestamp from the RFID data is automatically dismissed.

4. Ratio calculation: Ratio coverage is the amount of a path that is captured by a 3D camera corresponding to an individual and has been covered by an RFID tag based on the matching timestamp. In table 4.5 all the data points from the 3D camera have been covered and therefore the ratio corresponds to `1.0`. Whereas in table 4.6, the last two entries could not have been matched with a RFID Tag and therefore the row contains `NaN` values. The ratio coverage in this case would correspond to `0.67`. The formula used is the following:

$$1 - \frac{number_of_NaN_values}{length_of_dataset}$$

Calculating the ratio is an important method of filtering out the results. RFID tags could strongly correlate (more than 80%) with a person's path under the 3D camera, however, if the ratio is small, the match must be disregarded since the likelihood of the RFID Tag matching a person is very low.

5. Correlation calculation: An index that measures the strength of the linear relationship between two variables, such as x and y, is the correlation coefficient. Correlation coefficients higher than zero indicate a positive correlation, a value that is less than zero

Table 4.3: Example of a 3D camera dataset

timestamp	filialeId	readerName	EPC	xCm	yCm
2021-07-15 10:51:40.567	schlieren	center	E28011700000021428BF8061	-270	172
2021-07-15 10:51:41.612	schlieren	center	E28011700000021428BF8061	-257	164
2021-07-15 10:51:42.206	schlieren	center	E28011700000021428BF8061	-197	102
2021-07-15 10:51:43.883	schlieren	center	E28011700000021428BF8061	-142	82
2021-07-15 10:51:44.745	schlieren	center	E28011700000021428BF8061	-70	128
2021-07-15 10:51:45.426	schlieren	center	E28011700000021428BF8061	-31	130

Table 4.4: Example of a RFID reader dataset

timestamp	filialeId	camera	id	x	y
2021-07-15 10:51:40.015	schlieren	center	1600	-182.0	119.0
2021-07-15 10:51:40.975	schlieren	center	1600	-137.0	124.0
2021-07-15 10:51:42.015	schlieren	center	1600	-101.0	117.0
2021-07-15 10:51:42.975	schlieren	center	1600	-66.0	104.0
2021-07-15 10:51:43.935	schlieren	center	1600	-32.0	102.0
2021-07-15 10:51:44.815	schlieren	center	1600	0.0	103.0

indicates that the relationship is negative and, lastly, a value of zero indicates no relationship between the variables x and y. This study uses the `pandas.DataFrame.corrwith` correlation function from [58]. The function performs pairwise correlation between rows or columns, and it uses the Pearson [59] correlation coefficient by default, which is a standard correlation coefficient. One can specify other correlation types, such as Kendall [60] and Spearman [61]. Once the correlation has been computed, the correlations higher than a specified threshold are filtered and the lower ones are discarded.

6. Duplicates removal: A personId may be assigned to multiple RFID Tags at the same time because all the combinations between 3D camera's ID (personId) and EPC (RFID tag's ID) are being calculated. Due to this, duplicate RFID tags must be removed and only the RFID tag with the highest correlation and ratio should be kept.
Similarly, RFID tags can be assigned to multiple person IDs simultaneously. When the matching entry with the highest correlation and ratio is assigned to a personId, the corresponding tag's EPC is added to a different list so that it can no longer be associated with another personId.

7. Data reconstruction and insertion: The final step of the algorithm is to reconstruct the data to be inserted into the database. It utilizes the original dataset of the 3D camera and combines it with the result of the computation along with the correlation and ratio percentage. After this procedure has been completed, the new data is inserted into a new table in the database for later use.

Table 4.5: Merged dataframes with tolerance of 1 second

timestamp	filialeId	camera	id	x	y	EPC	xCm	yCm
2021-07-15 10:51:40.015	schlieren	center	1600	-182.0	119.0	E28011700000021428BF8061	-270	172
2021-07-15 10:51:41.612	schlieren	center	1600	-137.0	124.0	E28011700000021428BF8061	-257	164
2021-07-15 10:51:42.206	schlieren	center	1600	-101.0	117.0	E28011700000021428BF8061	-197	102
2021-07-15 10:51:43.883	schlieren	center	1600	-66.0	104.0	E28011700000021428BF8061	-197	102
2021-07-15 10:51:44.745	schlieren	center	1600	-32.0	102.0	E28011700000021428BF8061	-70	128
2021-07-15 10:51:45.426	schlieren	center	1600	0.0	103.0	E28011700000021428BF8061	-31	130

Table 4.6: Merged dataframes with tolerance of 1 second and with NaN values

timestamp	filialeId	camera	id	x	y	EPC	xCm	yCm
2021-07-15 10:51:40.015	schlieren	center	1600	-101.0	117.0	E28011700000021428BF8061	-197.0	102.0
2021-07-15 10:51:41.612	schlieren	center	1600	-66.0	104.0	E28011700000021428BF8061	-197.0	102.0
2021-07-15 10:51:42.206	schlieren	center	1600	-32.0	102.0	E28011700000021428BF8061	-142.0	82.0
2021-07-15 10:51:43.883	schlieren	center	1600	0.0	103.0	E28011700000021428BF8061	-142.0	82.0
2021-07-15 10:51:44.745	schlieren	center	1603	-14.0	97.0	NaN	NaN	NaN
2021-07-15 10:51:45.426	schlieren	center	1600	66.0	114.0	NaN	NaN	NaN

Table 4.7: Final dataset containing a match between a personId and a RFID tag

timestamp	filialeId	camera	id	x	y	EPC	correlation	ratio
2021-07-15 10:51:40.015	schlieren	center	1600	-182.0	119.0	E28011700000021428BF8061	0.752792	1.0
2021-07-15 10:51:41.612	schlieren	center	1600	-137.0	124.0	E28011700000021428BF8061	0.752792	1.0
2021-07-15 10:51:42.206	schlieren	center	1600	-101.0	117.0	E28011700000021428BF8061	0.752792	1.0
2021-07-15 10:51:43.883	schlieren	center	1600	-66.0	104.0	E28011700000021428BF8061	0.752792	1.0
2021-07-15 10:51:44.745	schlieren	center	1600	-32.0	102.0	E28011700000021428BF8061	0.752792	1.0
2021-07-15 10:51:45.426	schlieren	center	1600	0.0	103.0	E28011700000021428BF8061	0.752792	1.0

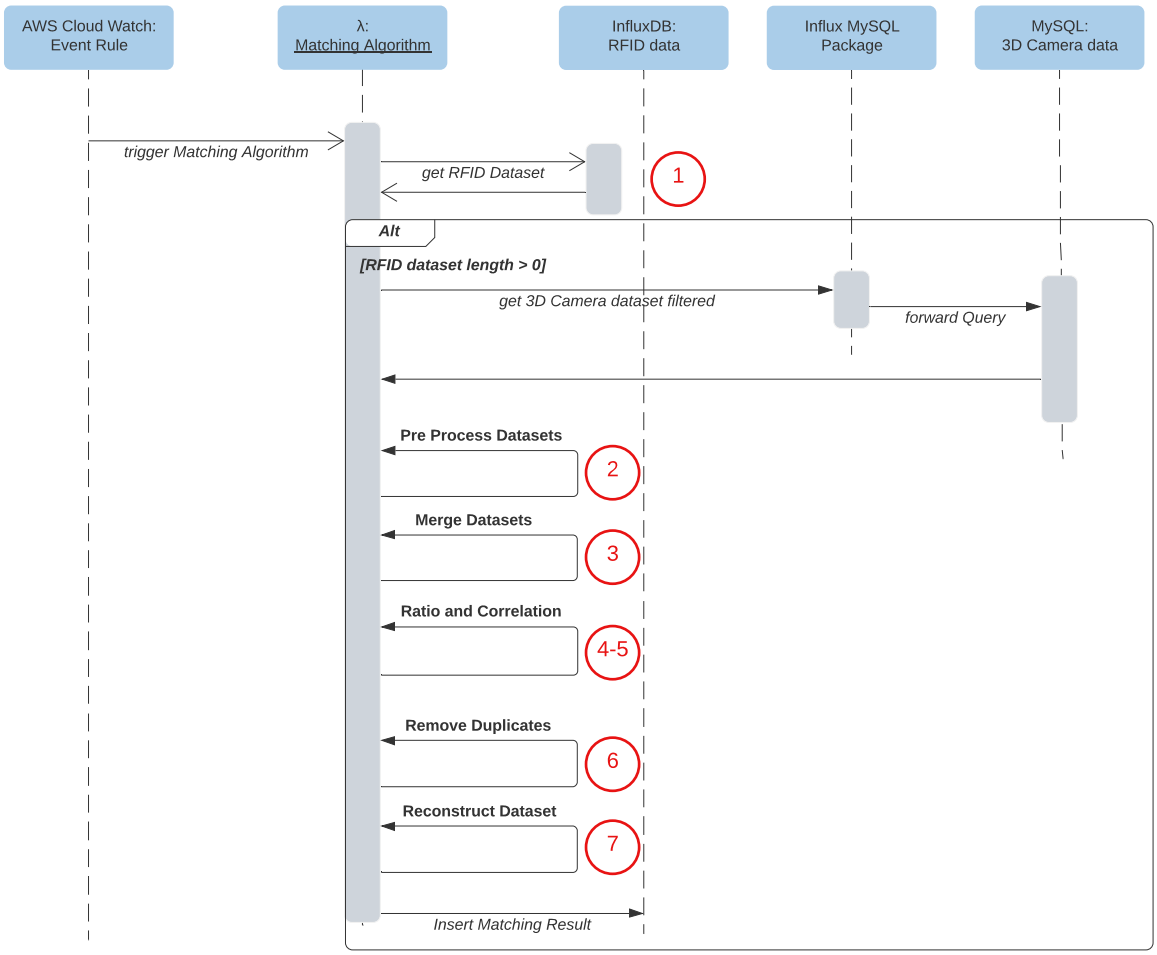

Figure 4.12: 3D Camera and RFID reader data matching algorithm explained with a sequence diagram

4.2.5 Environment and Deployment

The deployment process in software and web development refers to the process of pushing changes or updates from one deployment environment to another. Setting up a software always involves a live environment, also called the production environment.
Adding additional environments will enable to make changes without affecting the production website. These environments are referred to as development environments or deployment environments. The following subsection presents the process taken for developing *C-Counter 2.0*.

Local Development

The advantage of working in a local environment comes into play at an early stage of a project when a trial and error approach is employed. Not only does this improve the overall development experience by reducing the amount of waiting time and allowing for instant feedback, but it also lowers costs since the infrastructure does not have to be deployed on the cloud each time, the lambda functions run locally, and the database is hosted locally.
Consequently, the first step was to migrate the existing database of *C-Counter 1.0* to a Raspberry Pi 4 and enable the existing infrastructure to run locally with the SAM CLI. Because the study was no longer focused on collecting as much data as possible from 3D cameras, but rather on RFID data, the performance of the RPi 4 proved adequate for this study. Indeed, there are no noticeable differences in performance. As already mentioned, the SAM CLI was employed to run the project locally. AWS SAM stands for Serverless Application Model and it is an open source framework used to build serverless applications. By utilizing the CLI, it is possible to verify that AWS SAM template files are written in accordance with specifications. The CLI enables to run Lambda functions locally, debug them step-by-step, package and deploy entire serverless applications to the AWS Cloud, and so on [57]. The command shown below creates a local HTTP server listening to port `3004` that hosts the serverless functions contained in the same directory as the AWS SAM template. All Lambda functions can be set up with an environment file in `JSON` format. The advantage of this approach is that it can be used to overwrite global attributes such as names of other Lambda functions, database names, credentials, and so on.

```
sam local start-api –profile cc –host 192.168.0.196 -p 3002 –env-vars env.json –region
eu-west-1
```

Once the code has been developed locally and all the desired functionalities have been implemented, it can be built and deployed on the cloud by means of the AWS CLI.

As part of the development process, a local InfluxDB instance has also been configured. As opposed to previous scenarios, this time a docker image with persistent storage has been employed. This ensures that the configuration and the data collected remains available

between reboots. Getting a InfluxDB up and running is simple as running the following command on the terminal:

```
docker run -p 8086:8086 -v influxdb2:/var/lib/influxdb2 influxdb:2.0
```

Cloud Deployment

While a local environment comes in very handy in an early stage of the project when many changes and therefore manual tests are required, a cloud deployment was employed towards the end of the study. A cloud deployment means no need for onsite hardware and stable internet connection, characteristics very important when testing the system in a warehouse with limited access and control. To make sure that the system was fully and correctly working in the cloud, AWS CloudWatch was used. With CloudWatch, it is possible to get a unified view of all AWS resources, applications, and services that run on AWS and on-premises servers, such as logs, metrics, and events. Moreover, a cloud monitoring application such as CloudWatch allows to detect anomalous behavior in the environment, set alarms, visualise Lambda functions' logs and metrics side-by-side, take automated actions, troubleshoot issues, and find insights into how well the applications work [19].
An aggregated view of a Lambda's function parameters is shown in Figure 4.13 during a test conducted on the 01-07-2021 at approximately 11:00 o'clock. The `Invocations` chart illustrates how many times the function has been invoked over a period of time. A peak is evident at the point where a real-world test was conducted, demonstrating that the data from the RFID readers was reaching the cloud infrastructure. Additionally, the `Duration` chart indicates how long a function takes to complete, this indicator can be used to improve the performance of the function by reducing its execution time. According to the `Error count and success rate(%)` chart, no errors have occurred during the test. What is noteworthy is the fact that the system scaled automatically in response to increased activity. The execution time for the lambda function significantly decreased upon increasing the concurrent executions to two as shown in the `Concurrent executions` chart.

InfluxDB Dashboard

A tool such as AWS CloudWatch has proven very effective in monitoring the health of the infrastructure and system. Although, when performing real-world tests onsite, ensuring that data reaches the cloud and is stored in the database is not sufficient to fully understand the meaning of it. Therefore, a InfluxDB Dashboard was conceived to support the experiments. Visualizing the data using dashboards is a great way to enhance the understanding of it and providing a meaningful and easy-to-understand format for the data is essential for reaching the intended goal.
Time series data in InfluxDB can be viewed in dashboards, whose visualization types include line graphs, gauges, and tables. The options are diverse, including dashboards that come with InfluxDB, Open Source projects like Grafana, and IoT dashboarding tools

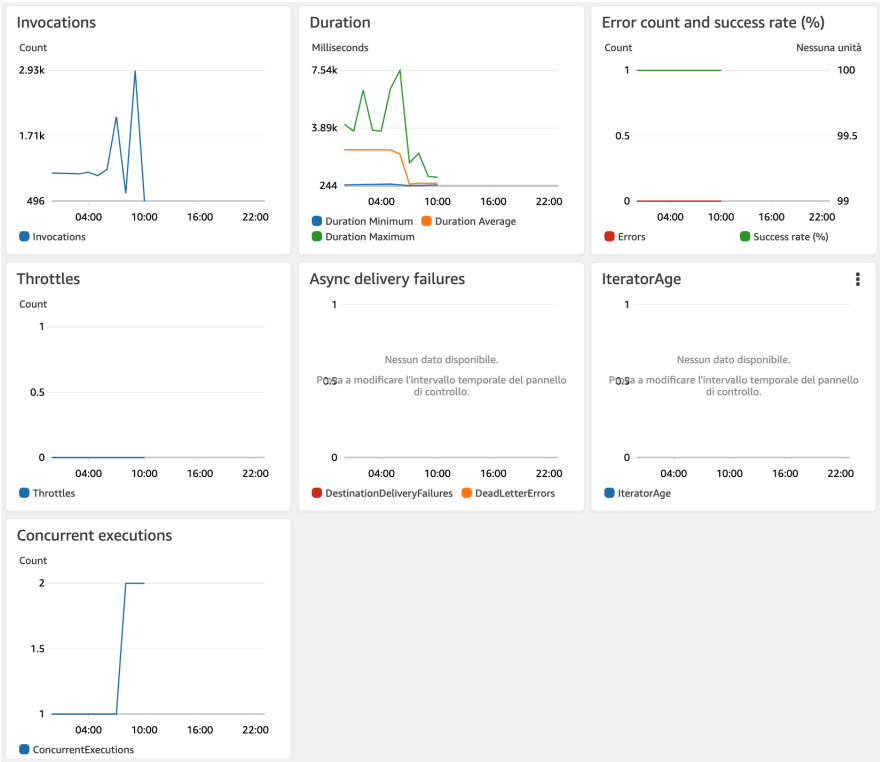

Figure 4.13: AWS CloudWatch lambda function's metrics

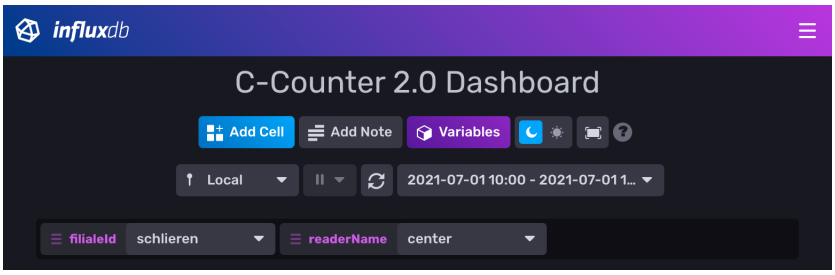

Figure 4.14: Dashboard control panel with variables.

like Seeq. They usually include pre-built dashboards built by the community so that it is possible to get started right away. To support debugging during real-world tests, a custom dashboard with variables has been created in this study. Dashboard variables permit to specify changes to certain components of a cell's query without editing the underlying query, making it simple to interact with cells on the dashboard and explore the data.
Figure 4.14 illustrates the control panel of the *C-Counter 2.0* dashboard. By changing the `filialeId` and `readerName`, the dashboard will be updated accordingly. Further, a time range may be specified for filtering the data, and an automatic polling of the data may be activated so that the dashboard will always display the most current and updated information from the database without having to reload the entire page. During the experiment, the real-time position is useful to assess the quality of the data. A scatter plot has therefore been constructed in order to display the last known location of the RFID tags almost in real time. The plot is automatically updated every 5 seconds, providing feedback on the performance of the system. Figure 4.15 shows an example taken during

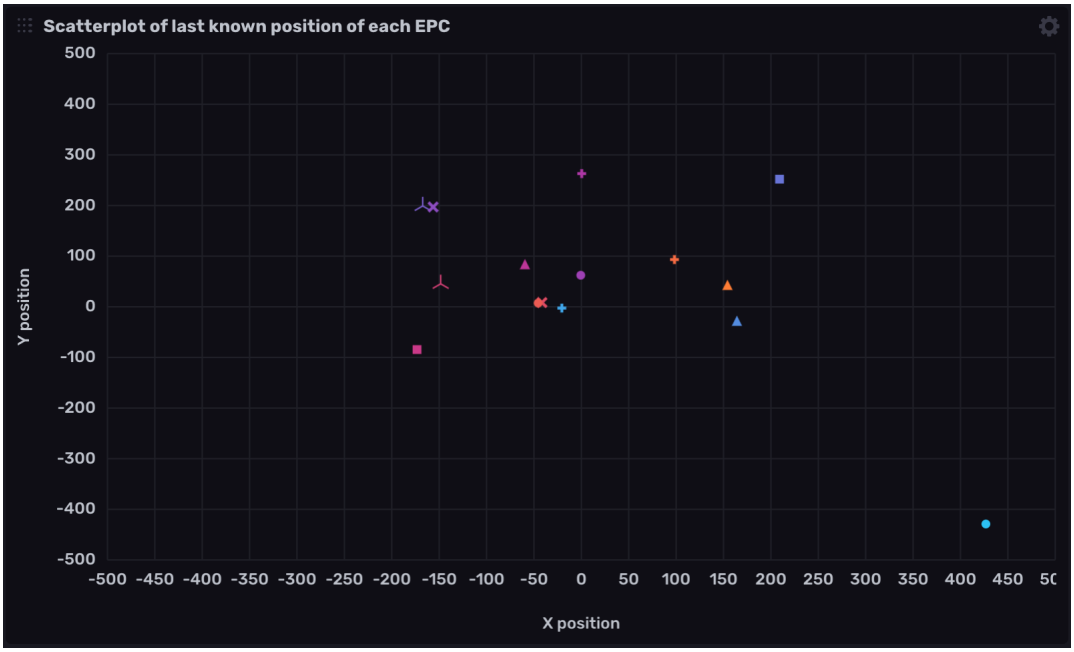

Figure 4.15: Scatter plot of last known EPC's position.

an experiment. The scatter plot is complemented by a heatmap to enhance the value of the data. In Figure 4.16, it is presented a heatmap of the same experiment that is based on the complete dataset rather than only the last known positions of the RFID tags.

Since the sensitivity of RFID Tags can be compromised by slightly covering them with the hand or with an object, a live counter of how many RFID tags are being discovered has been produces. Moreover, a table of EPCs found and their corresponding last-seen timestamp is shown as well in figure 4.17. It has demonstrated its usefulness in practice while assigning different EPCs to different persons for the purpose of analyzing and evaluating the data at a later step.

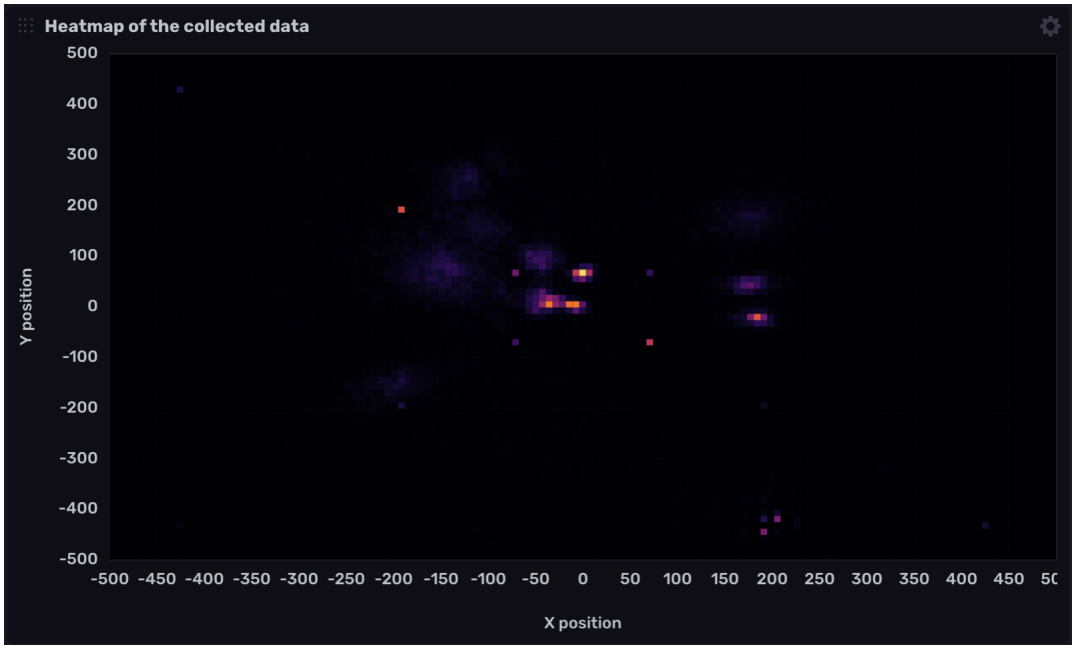

Figure 4.16: Heatmap of the recorded EPC's position.

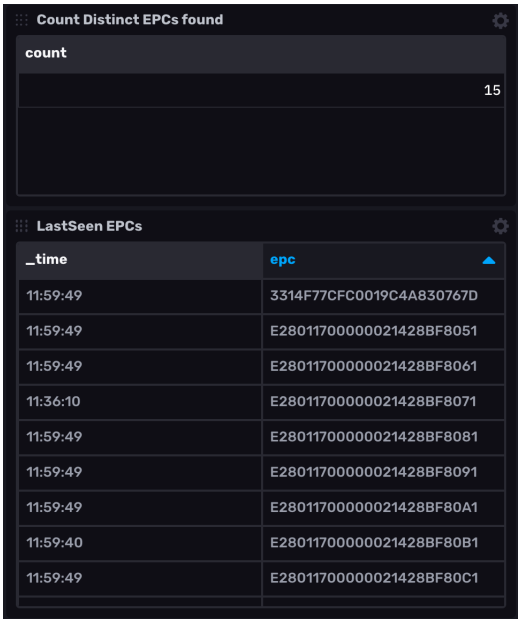

Figure 4.17: Dashboard control panel with variables.

Chapter 5

Evaluation

This chapter presents the evaluation of *C-Counter 2.0*. During the evaluation, attention will be focused on the system developed to combine and match the data collected by RFID readers with that of 3D cameras. A comparison of different RFID reader configurations will be done by demonstrating the results from a practical example. Secondly, an analysis of the algorithm used to match RFID tags with people will be done in order to assess its accuracy and precision. The system has been subjected to a wide range of test cases in order to identify which scenarios it performs best and which ones are more challenging. The third section provides a discussion section that highlights a few major concerns with the system.

5.1 Comparison of RFID Reader Modes

In this section, the performance of the RFID reader will be evaluated under several controlled scenarios in order to better understand and quantify the behavior of the system. The first scenario evaluated refers to the precision and frequency with which the RFID reader can locate RFID tags.
As previously discussed in section 4.1.3, choosing the right reader mode based on the environment is key to obtaining the most accurate tag location calculation. In order to ensure consistency, the Impinj xArray R680 reader has been tested in all possible modes under the same conditions. The test bed consisted of fourteen passive UHF RFID tags that were arranged in specific locations within the scene, as well as an RFID reader mounted on the ceiling. After running each mode for a minute, the collected data was analyzed.

Table 5.1 shows the first analysis performed. In spite of the use of multiple reader modes, 13 tags out of 14 were able to be captured, *i.e.,* with a success rate of *92.9%*. As for the missing tag, it was not isolated from the others, rather it was placed in a dispersed location where the signal was not able to be successfully captured. As part of the matching algorithm between 3D cameras and RFID tags, a critical metric is the frequency with which a particular RFID tag can be captured, *i.e.,* how many data points a reader can deliver within a certain period of time. As a result of changing several parameters in the

Table 5.1: Average count of entries and number of EPCs found for each reader mode

Algorithm	Average Count	Number of EPCs found
Auto Set Custom	59.538462	13
Auto Set Dense Reader Deepscan	62.153846	13
Auto Static Dense Reader	62.384615	13
Auto Static Fast	60.692308	13
Dense reader M4	60.769231	13
Dense reader M8	61.384615	13
Hybrid M2	61.307692	13
Max Miller	60.076923	13
Max Throughput	61.461538	13

configurations, all modes were able to deliver approximately 60 entries within 1 minute, which corresponded to the granularity of a 3D camera of one data point per second. Thus, the accuracy of the various algorithms can be compared.

As all of the RFID tags were not moved and thus statically located, an accuracy comparison was performed to asses the best reader mode that was most appropriate in the environment for that particular scenario. The figures A.1 to figure 5.2, present in the appendix section of this work, illustrate a pair of maps for each reader mode. The left plot represents the heat map for all the RFID tags captured within a minute, whereas the right graph shows the approximate position of all the data points captured within the minute for each RFID tag. In regions where tags are more distant from one another and therefore the density is low, all approximated positions are very similar across all the maps. The lower portion of both pair maps exhibits this behavior. As can be seen from the heat maps, the position of the tags are clearly visible. An illustration can be found in figure 5.1.

However, the RFID reader does not seem to be able to locate RFID Tags when they are clustered together closely, resulting in a higher density. On the plots, this issue can be seen in the upper left hand corner. As a result, the heat map is unclear and it is not possible to distinguish between the various tags. The RFID tags therefore appear to move continuously, despite the fact that they are not in motion.

The result of this behavior is a large amount of noise in the data, thereby making it more difficult for the matching algorithm to correctly identify individuals wearing a RFID tag, as they appear to be moving around. Overall, it is difficult to draw conclusions and clearly define which algorithm is the most effective. Optimal performance requires that this algorithms are run within a computing window of more than one second. However, since a tagging locations that occur every 1 second is required in order to match the data with the 3D camera, it is not possible to expand the computing window.

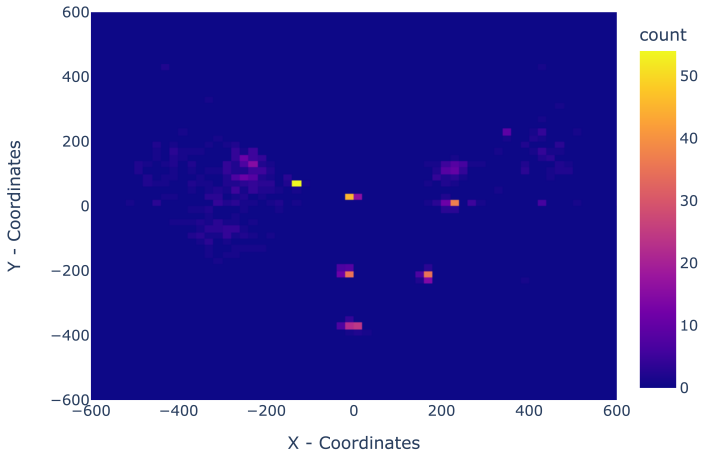

Figure 5.1: Heatmap of max throughput

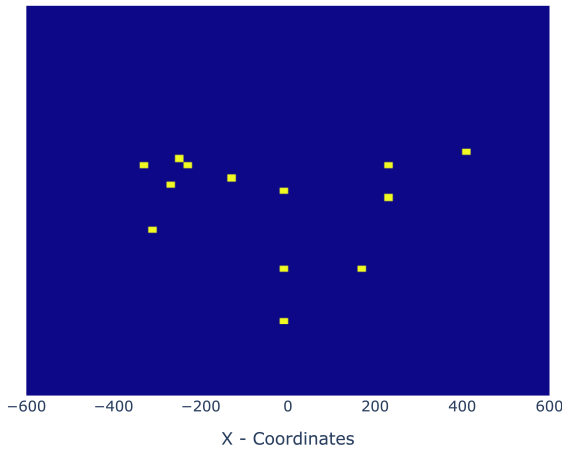

Figure 5.2: Mean of max throughput

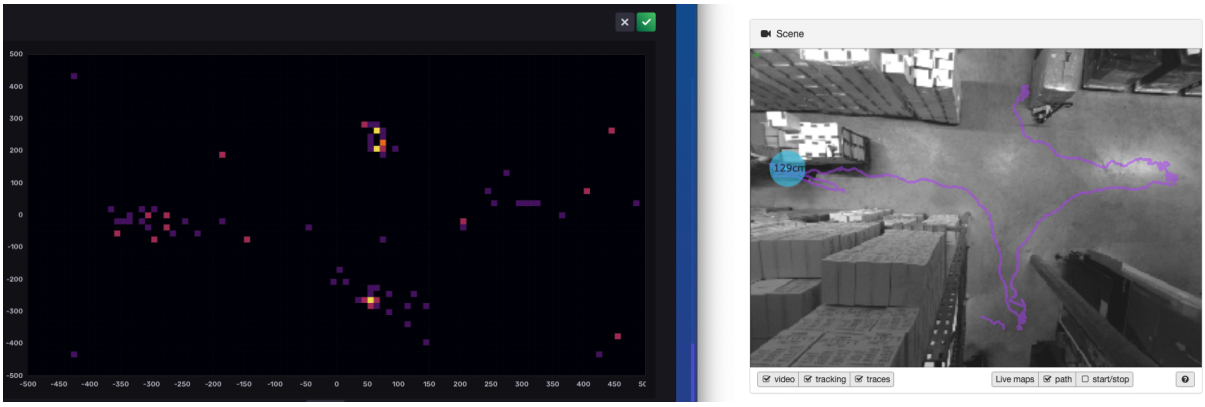

Figure 5.3: Frame of a recording used to manually evaluate the system.

5.2 Controlled real world test scenarios

The results of multiple real world scenarios will be presented and discussed in this section. This test cases were very helpful to ascertain where there are still some challenges to overcome when trying to match the data with the architecture and algorithm. In order to manually analyze the data after the experiment and be able to verify the result of the matching algorithm, all the experiments performed were recorded on a computer. Figure 5.3 shows a snapshot of a recorded video employed for this procedure. Moreover, this setup was used to asses the functionality of the system in real time and thus, be able to successfully perform the experiments.
Finally, to properly analyze the performance of the system and conduct testing, it was known beforehand which individual were wearing which RFID tag. Normally, this mapping is not provided because the system must remain GDPR compliant [40], and as such no mapping can be made between individuals and tags.

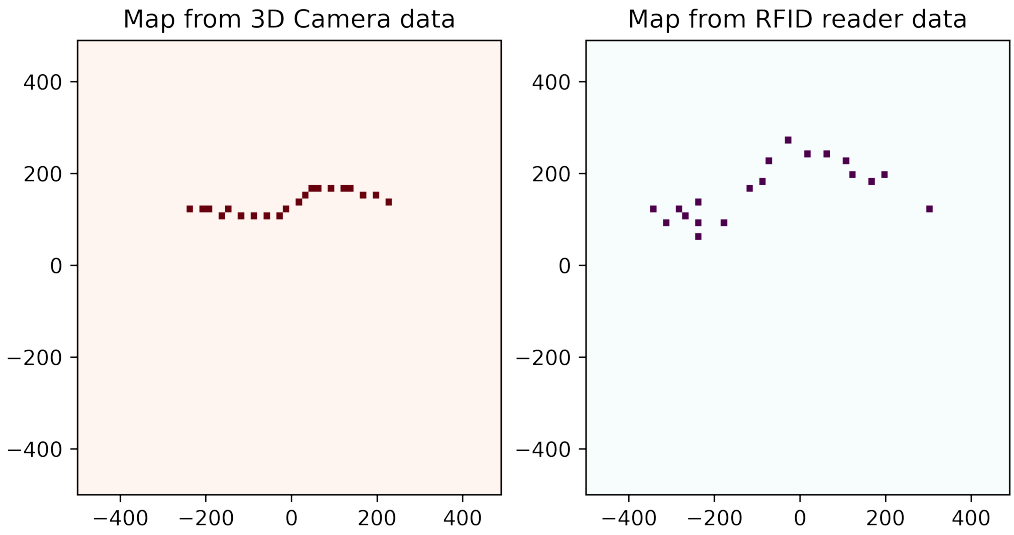

Figure 5.4: Maps from a 3D camera and a RFID reader of a single person walking in the scene.

5.2.1 Single RFID Tag

This scenario evaluates the accuracy and precision of the system when tracking a single individual wearing a RFID tag under the scene. It was found that the algorithm could identify the individual with 100% accuracy when the RFID reader was able to regularly capture the position of the RFID tag. The correlation between the data was higher than 70-80%. Figure 5.4 illustrates the data capture by a 3D camera and a RFID reader of a person wearing a RFID tag. In spite of the fact that the RFID tag's accuracy is not as high as the data from the 3D camera, there is a clear correlation between the two.

Move in/out and around the scene:

This variant of the scenario is very similar to the previous one, the only key difference is that the individual leaves the scene and upon returning, the 3D cameras assigns a new `ID`. The accuracy was found to be around 100% as well when the RFID reader was able to consistently capture the RFID tag position resulting in a correlation higher than 70-80% percent.

RFID Tag transferred to a partner without a tag:

The scenario presented here is one of the most challenging scenarios for the matching algorithm. Since one of the assumptions of this system is that each individual is assigned a personal and unique RFID tag, this scenario shall never happen. It has nevertheless been conducted for testing purposes in two variants:

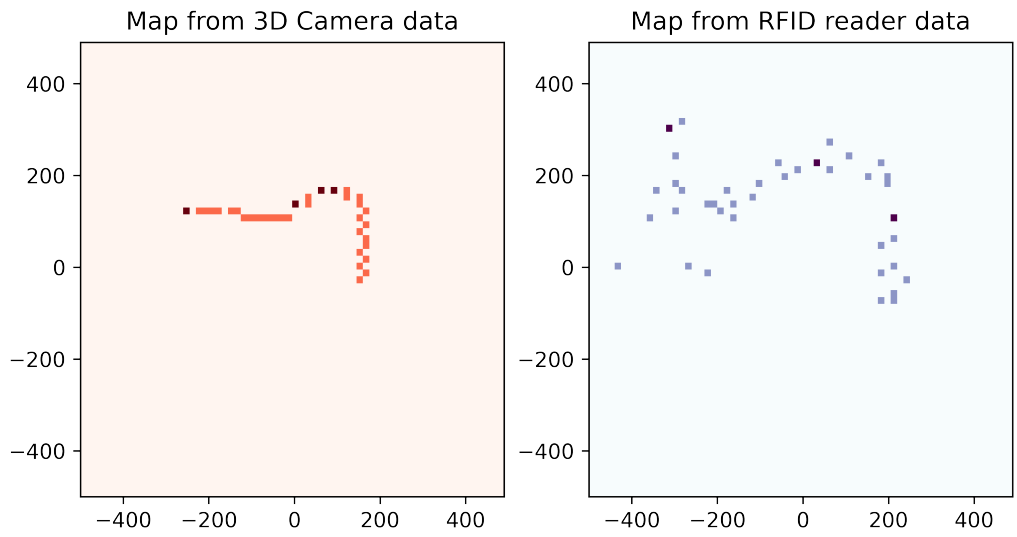

Figure 5.5: Maps from a 3D camera and a RFID reader of a single person wearing a RFID tag walking between a group of people.

- **Walking together on the same path and transferring the tag in the middle of the scene:** Due to the fact that the RFID tag was passed to a second individual somewhere in the middle of the scene, and because they were both walking in the same direction, the matching algorithm incorrectly attributes the entire path of the RFID tag to a single individual. It is possible to improve the algorithm by taking into account other metrics in addition to the correlation. This topic will be covered and discussed in section 5.3.3.

- **Crossing and transferring the tag in the middle of the scene:** Given that the two individuals were walking in opposite directions, the RFID tag transfer was not mistakenly associated with one individual. The correlation was insufficient due to the path not being fully covered. In this case the system successfully avoided to assign the RFID tag to a person.

Many people randomly walking but only a single individual is carrying a tag

Since the system is based on the assumption that each individual is assigned a personal and unique RFID tag, the scenario presented should never occur. In spite of this, it was conducted for testing the algorithm with some arbitrary noise in the data.
Several persons were randomly walking under the camera, but only one of them was carrying an RFID tag. Due to the fact that the individuals are moving in different directions, thus making it easier to find the right correlation for the algorithm, the tag was successfully assigned to the correct individual.A false positive rate of 0% was observed. Figure 5.5 shows the correct path of the single individual wearing the RFID tag.

5.2.2 Two RFID Tags

The objective of this scenario is to evaluate the accuracy and precision of the system when tracking two individuals wearing RFID tags and moving under a scene. It was found that the algorithm could not always identify the individuals, even though the RFID reader was able to regularly capture the position of the RFID tag and the correlation between the data was higher than 70-80%.

Crossing in the middle of the scene:

Using this variation of the scenario, the matching algorithm was able to successfully match an RFID tag to an individual captured by the 3D camera. The correlation between the data of an individual and that of another individual clearly differs, as their directions are not the same or even opposing, and therefore the algorithm had no difficulty assigning the RFID tag to that individual. It was not necessary to make any further adjustments to the matching algorithm.

Walking together on the same path:

This extended scenario proved to be the most challenging scenario for the system. Two individuals walking in the same direction and at the same time provide nearly identical data for the system. Consequently, in early versions of the algorithm, an individual may have been assigned to both RFID tags while the second may have been assigned to none. In order to resolve this issue, the matching algorithm had to be further fine-tuned and capable of handling such situations. As a result, although an individual can no longer be assigned to more than one RFID tag, the system is still capable of assigning the RFID tag to the wrong individual since the correlation and ratio are still high enough. Increasing the threshold for the ratio and correlation would solve this problem, but the algorithm would not be able to handle other scenarios where the correlation is not as high.

5.2.3 Multiple RFID Tags

A group of individuals wearing RFID tags under a crowded environment will be tracked in this scenario in order to assess the system's accuracy and precision. The test results indicated that the algorithm barely managed to assign the correct tags in each of the three test scenarios described below. Among the three scenarios tested, only the first one in which a single individual was walking had some success.

Single individual moving between others:

Every individual in this variant wore an RFID tag. Only one individual was walking around, however. It has been previously demonstrated that, with the presence of multiple

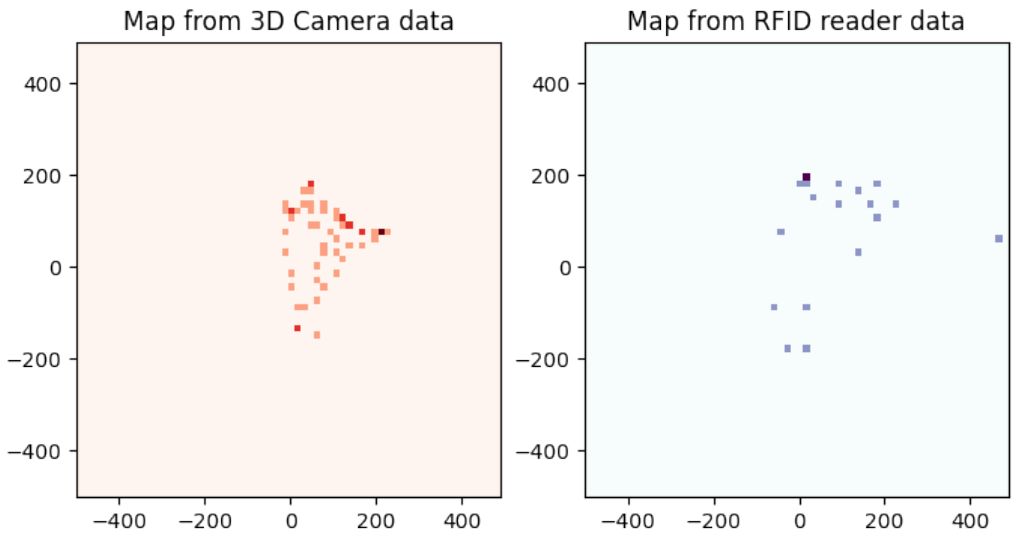

Figure 5.6: Example 1: Maps from a 3D camera and a RFID reader of a single person walking between a group of people where everybody is wearing a RFID tag.

tags, the inaccuracy of the RFID reader increases substantially, making it more challenging to match the ID of the 3D camera with the RFID tags. However, in some occasions, the algorithm was able to successfully find a match despite the noise. Figure 5.6 shows a path taken that was correctly assigned. As clearly seen, a lot of noise is present in the data coming from the RFID reader.

A group of individuals wearing a RFID tag and moving around randomly:

Under the given circumstances, the algorithm was unable to find a match, as noise levels are too high for the algorithm to find any correlation that meets the threshold established to match the data. Despite lowering the threshold significantly, no meaningful results were obtained. Further tests will be required in different environments and with different types of RFID tags, so that the algorithm can be fine tuned and made more effective. Moreover, different RFID readers and different configurations shall be thoroughly tested.

A group of individuals wearing a RFID tag and standing still:

Similarly to the previous scenario, this variant did not yield any meaningful results. The algorithm is designed to take only the correlation into account, which resulted in wrong results as shown in figure 5.7. A different metric, such as the `Euclidian distance`, would have prevented this wrong assignment from occurring.
This scenario is however extremely unlikely to occur, in contrast to the preceding one. Generally, before and after stopping under the scene, individuals are likely to enter and leave the scene, which facilitates finding a correlation between the data and correctly

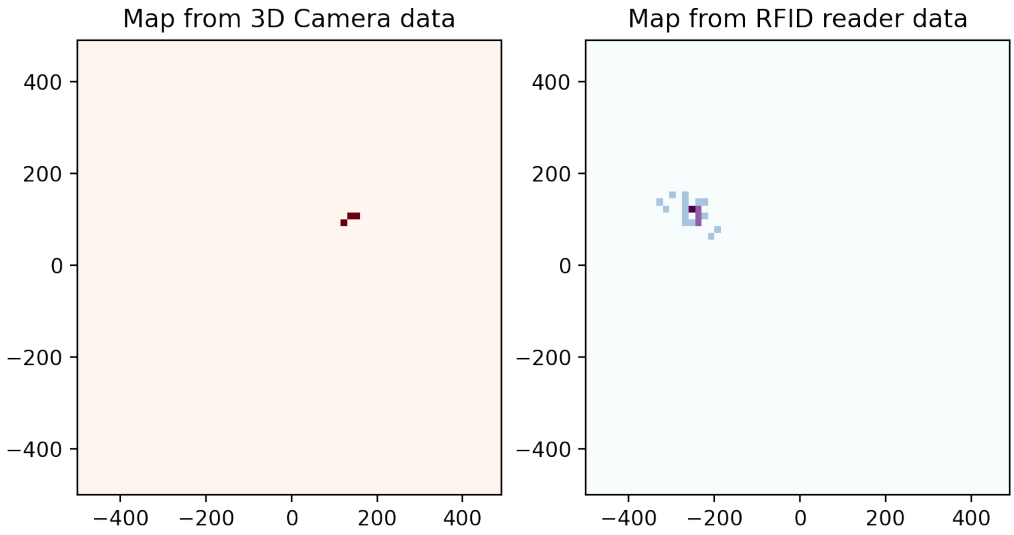

Figure 5.7: Wrong assignment of an RFID tag to a person standing still surrounded by a group of people.

identifying individuals. Further testing with multiple individuals shall be performed once the restrictions imposed by the COVID-19 pandemic are rescinded.

5.2.4 Visitor counts: 3D Camera vs. RFID Reader

As of now, the focus was on how to integrate and merge data from both sources. Data collected during the experiments can, however, be analyzed independently from each other and compared. In order to conduct this comparison, the visitor counts are grouped according to 1 minute intervals. The result of this operation is depicted in figure 5.8. In contrast to the 3D camera, which assigns different and random IDs to people entering and leaving the scene, the RFID tags, thanks to their EPC, are always uniquely identifiable by the RFID reader. In this manner, the data collected from RFID readers can be used to determine the unique number of visitors present at any given point in time. In this specific test scenario, there were two participants physically present, but the 3D camera has assigned up to 13 different identities within five minutes (when using a grouping based on 5 minutes intervals, the different IDs were found to be 35). On the other hand, the number of RFID tags did not exceed the actual count of RFID tags present, *i.e.,* 14.
In the period between `10:15 - 11:05 o'clock` the experiments were performed using two different RFID tags, and the participants entered and left the camera's field of view constantly, resulting in a large discrepancy between the two measurements.

In the time frame between `11:10 - 11:20 o'clock` the different Reader Modes of the RFID reader were tested, as already explained in section 5.1. In this test scenario no participants were present under the scene. Instead, RFID tags were statically placed under the camera, therefore, the camera's count is equal to 0, whereas the RFID count is almost

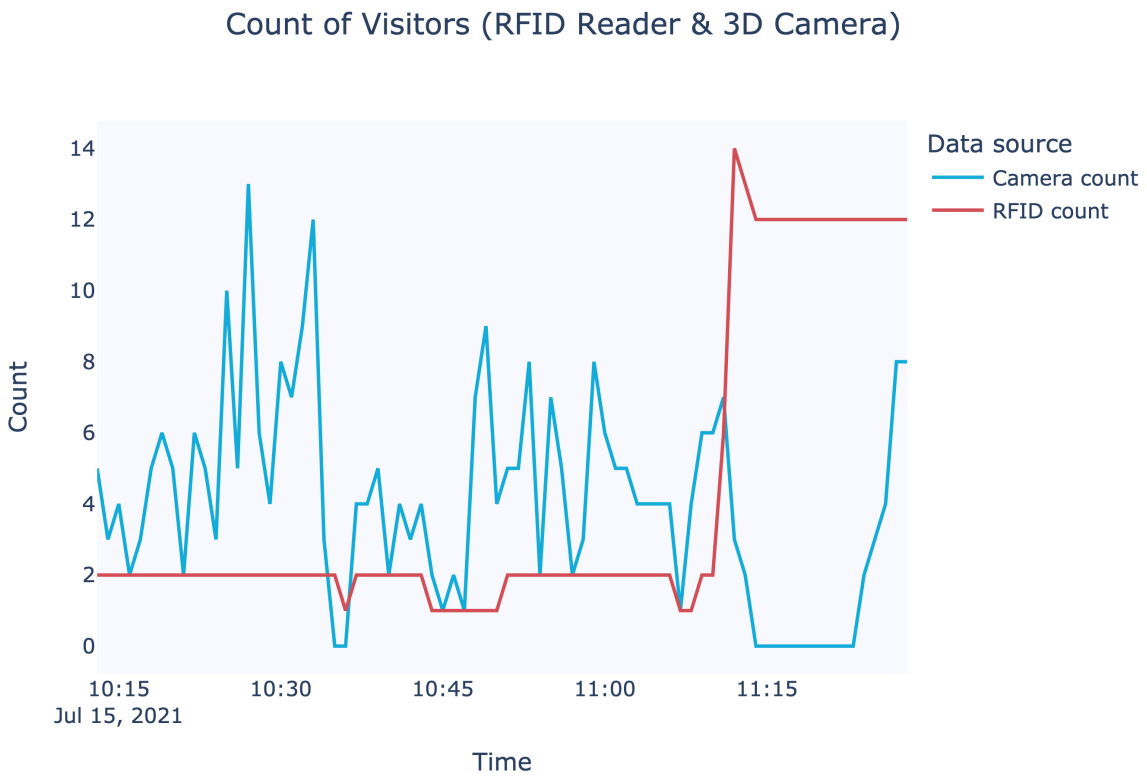

Figure 5.8: Count of visitors from 3D cameras and RFID readers during a test session

constant to 13. As a consequence, the number of visitors counts taken from RFID readers data can be directly interpreted as **unique** number of visitor counts.

5.3 Discussion and Findings

This section introduces a discussion of the evaluation results. After, it explores how much the implemented system adheres to the specifications identified as the main objectives of this study. Towards the end of this section, a number of limitations of the implemented solution and methods are discussed.

5.3.1 Matching solution and the RFID reader accuracy

In this study, the main objective was to match 3D camera data with RFID data. The solution provided here provides a stable pipeline and a system with almost no configuration effort required. A new event could be added and analyzed without significant configuration. The *C-Counter 2.0* has been designed so that it automatically detects the running RFID readers with the paired 3D camera and it runs the matching algorithm against it. The algorithm has already been proven to provide a high level of accuracy in identifying individuals wearing RFID tags as long as the RFID readers provides qualitatively good data. However, the accuracy of the matches decreases significantly when the location of multiple RFID Tags has to be calculated at the same time. With such a configuration and circumstances, the RFID reader is unable to keep up with performance and precision. It is possible however to increase the accuracy of RFID reader by specifying a larger `Compute Window` as shown in figure 4.4 of section 4.1.2. Thanks to a larger `Computer Window`, or also known as smoothing window, the RFID reader performs an average of the location estimates and therefore improves the accuracy. As a consequence, if specifying a `Compute Window` of 2 seconds, the 3D camera would have to sample data at the same frequency as the RFID reader. A decrease in data quantity would result in both *C-Counter 2.0* and *C-Counter 1.0*. The final data may be cleaner, but with fewer samples and therefore, to determine the best tradeoff between data quantity and quality, further research is necessary.

5.3.2 Synchronization Offset on the RFID Reader

Speedway's RFID reader features a built-in clock that keeps track of date and time information. Since it is subject to drift like every clock, it may need to be synchronized periodically. There are two main approaches found in [62] to accomplishing this:

- **Automatic Synchronization Using Network Time Protocol (NTP):**
  In the event that the Speedway Reader is connected to the Internet via a router and is configured with the Dynamic Host Configuration Protocol (DHCP), the router will normally have a NTP server configured and will synchronize the reader's date and time.

- **Manually Configure NTP:**
  The Speedway Reader can be configured manually to point to an NTP server by entering the command "config network ntp add [ntp server name or address]" in a Rshell. To accomplish this, the NTP protocol must first be disabled.

However, despite testing and configuring these two approaches, the data from the RFID reader contained a wrong timestamp. The timestamp was usually between two and fifteen minutes ahead of the actual time. In light of such a wide difference in time, the matching algorithm was unable to successfully match the data based on the timestamp. Therefore, the timestamp coming from the RFID reader is at the moment ignored and assigned by the Influx Database at the time of storing the data.

Workarounds such as this work as long as the small time difference is considered. Furthermore, if the system becomes overloaded, the actual timestamp may differ even more from the one assigned. It is possible to improve the accuracy of the matching algorithm by fixing the timestamp on the RFID reader, since the first step of the algorithm is to compare both sources of data based on the timestamps.

5.3.3 Measuring Statistical Dependence

The Pearson correlation has been employed in this study as a tool to determine whether there is any statistical evidence for a linear relationship among variable pairs such as coordinates. The Pearson coefficient, despite some errors sometimes present, has proven to be a very reliable measurement for comparing data. As shown in [14], the squared `Euclidian Distance` is in fact equal to a distance based on the `Pearson Correlation`. Due to the fact that Pearson Correlation does not require normalization of the data, unlike the Euclidian distance, this result is of particular interest and makes the implementation easier. In order to normalize the data for RFID readers, each time the boundaries of the 3D camera need to be manually mapped into the RFID reader's coordinate system. Whenever a change is made to the configuration of the RFID reader, such as changing the antenna power or moving the reader, these values must be reassessed. In addition, the matching algorithm should be able to read the values for each RFID reader and apply the normalization. There is no easy way to automate this process. Therefore, the `Pearson Correlation` has been employed throughout this study.

Nonetheless, other metrics should be considered in order to improve *C-Counter 2.0*'s accuracy and reliability. As an additional layer of matching, Manhattan distance, cosine similarity, Jaccard or even the Euclidian Distance similarity can also be used.

Chapter 6

Final Considerations

In this work, a system that is capable of collecting, storing, processing and matching data from 3D cameras from third parties with data from RFID readers was developed. To ensure acceptable social distance as well as provide a strategic and near-real-time business intelligence application, the tracking accuracy of *C-Counter 1.0* has been improved. This enhancement has been accomplished by analyzing camera path information in combination with tracking information provided by RFID tags. In an innovative and easy-to-use manner, RFID tags and tracked objects can be linked in an easy manner to form unique identification links. Furthermore, the approach conforms to GDPR regulations, maintaining privacy by not reporting any personally identifiable information to RFID tags.

A detailed description of both the high-level design and the technical implementation of *C-Counter 2.0* has been put together, covering all relevant elements. Furthermore, each aspect has been discussed to enable the reader to gain a better understanding of the development process involved. Throughout the implementation phase, the system has been continually refined several times in order to provide a robust pipeline and to minimize the need for manual configuration in the set-up and data analysis processes.

The system has been implemented in such a way that it can automatically detect whether a 3D camera has been paired with an RFID reader or not. This means that the effort to create a new event is almost nonexistent, and no configuration needs to be created or stored in the system. In assessing the convenience and practicality of a solution of this kind, it is important to take this point into account.

Experimental results showed that *C-Counter 2.0* is indeed capable of identifying people with enough accuracy as long as the data provided by RFID reader is accurate, thus only when a few people are being captured by the system at the same time. Nonetheless, even without matching the data from 3D cameras, the system would be able to identify at any given time the number of unique visitors since more than the 90% of RFID tags are captured.

Ultimately, the development of the solution encountered a number of challenges. On the one hand, RFID readers were not always reliable and consistent in providing data pushes. There has been considerable effort allocated to setting up a proper infrastructure and

to taking steps to ensure that the pipeline would finally be stable in order to conduct experiments. On the other hand, as outlined in section 5.3.2, the architecture had to be adapted in order to make the system work, as the data was not always accurate.

6.1 Future Work

The effects of integrating other RFID readers based on the same or different technologies should be considered in future research. Furthermore, infrared sensors, such as LIDAR sensors, could be incorporated into the system as well. This would potentially result in an improvement to the accuracy of *C-Counter 2.0*'s measurements and the flexibility of the overall approach. Through such integration, it is also possible to gain new insights into visitor flow and behavior.

Generally, the read rate of a tag is determined primarily by the amount of RF power that it can receive. As a result, RFID tags with a low reading rate are often located at the edge of the reading zone, which lies beyond the field of view of the 3D camera. As a consequence, such tags can be excluded automatically since they do not qualify as matches. Therefore, future studies should examine the benefits and costs associated with incorporating more data pre-processing steps into the pipeline, as well as employing additional solutions for determining statistical dependencies, as discussed in section 5.3.3. Moreover, existing solutions, like *C-Counter 1.0* could be migrated to utilizing a time series database such as InfluxDB to improve the performance and achieve a homogeneous environment through the different proposed solutions suited for the integration of other technologies.

As a final point, the set of functionalities provided by *C-Counter 2.0* could also be expanded. In fact, by utilizing an Influx database, export functions, such as exporting the processed information as excel sheets and .csv files, could easily be implemented, as well as an automatic report generation.

Bibliography

[1] Myrvoll, Tor A and Håkegård, Jan E and Matsui, Tomoko and Septier, François. *Counting public transport passenger using WiFi signatures of mobile devices.*2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). 2017.

[2] Kalogianni, Eftychia and Sileryte, R and Lam, Marco and Zhou, Kaixuan and Van der Ham, Martijn and Van der Spek, S and Verbree, E. *Passive wifi monitoring of the rhythm of the campus.* Proceedings of The 18th AGILE International Conference on Geographic Information Science. 2005.

[3] Vaccari, Andrea and Samouhos, Stephen and Glicksman, Leon and Ratti, Carlo. *MIT enernet: correlating WiFi activity to human occupancy.* Proceedings of Healthy Buildings. 2009.

[4] B. Rodrigues, C. Halter, M. Franco, E. J. Scheid, C. Killer and B. Stiller, "BluePIL: a Bluetooth-based PassIve Localization Method," 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2021, pp. 28-36.

[5] R. H. Ribeiro et al., "ASIMOV: a Fully Passive WiFi Device Tracking," 2021 IFIP Networking Conference (IFIP Networking), 2021, pp. 1-3, doi: 10.23919/IFIPNetworking52078.2021.9472786.

[6] Bocek, Thomas, et al. "Blockchains everywhere-a use-case of blockchains in the pharma supply-chain." 2017 IFIP/IEEE symposium on integrated network and service management (IM). IEEE, 2017.

[7] Vu, Long and Nahrstedt, Klara and Retika, Samuel and Gupta, Indranil. *Joint bluetooth/wifi scanning framework for characterizing and leveraging people movement in university campus.* Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems. 2010.

[8] Betzing, Jan Hendrik. *Beacon-based customer tracking across the high street: perspectives for location-based smart services in retail.* 2018.

[9] Zambrano, Brian: *Serverless Design Patterns and Best Practices: Build, secure, and deploy enterprise ready serverless applications with AWS to improve developer productivity. 2018.*

[10] Li, Hanchuan and Zhang, Peijin and Al Moubayed, Samer and Patel, Shwetak N. and Sample, Alanson P.: *ID-Match: A Hybrid Computer Vision and RFID System for Recognizing Individuals in Groups. 2016.*

[11] Wang, Ching-Sheng and Cheng, Li-Chieh: *RFID & vision based indoor positioning and identification system. 2011.*

[12] Sample, Alanson P. and Macomber, Craig and Jiang, Liang-Ting and Smith, Joshua R.: *Optical Localization of Passive UHF RFID Tags with Integrated LEDs. 2012.*

[13] Wang, Zhongqin and Xu, Min and Ye, Ning and Xiao, Fu and Wang, Ruchuan and Huang, Haiping: *Computer Vision-Assisted 3D Object Localization via COTS RFID Devices and a Monocular Camera. 2019.*

[14] Berthold, MR and Höppner, F.: *On Clustering Time Series Using Euclidean Distance and Pearson Correlation. 2016.*

[15] K. V. Seshagiri, Pavel V. NIkiting and Sander F. Lam: *Antenna Design for UH RFID Tags: A Review and a Practical Application. 2009.*

[16] AWS Serverless, `https://aws.amazon.com/serverless/?nc1=h_ls`, accessed April 2021

[17] Amazon API Gateway, `https://aws.amazon.com/api-gateway/?nc1=h_ls`, accessed April 2021

[18] AWS Lambda, `https://aws.amazon.com/lambda/?nc1=h_ls`, accessed July 2021

[19] AWS CloudWatch, `https://aws.amazon.com/cloudwatch/`, accessed July 2021

[20] Amazon RDS, `https://aws.amazon.com/rds/?nc1=h_ls`, accessed March 2021

[21] AWS S3, `https://aws.amazon.com/s3/?nc1=h_ls`, accessed July 2021

[22] Installing the AWS CLI, `https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html`, accessed March 2010

[23] AWS Lambda Limits, `https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html`, accessed April 2021

[24] AWS CloudFormation, `https://docs.aws.amazon.com/cloudformation/`, accessed April 2021

[25] AWS CloudFormation Limits, `https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-limits.html`, accessed July 2021

[26] Xovis: Guaranteed data privacy, `https://www.xovis.com/de/technology/detail/guaranteed-data-privacy/`, accessed April 2021

[27] Xovis AG Website, `https://www.xovis.com/en/home`, accessed April 2021

[28] AWS Pricing, `https://aws.amazon.com/pricing/`, accessed April 2021

[29] Sensalytics: solutions, https://www.sensalytics.net/en/solutions, accessed April 2021

[30] Ipsos: solutions, https://www.ipsos-retailperformance.com/en/services, accessed April 2021

[31] Axper: People Counting & Tracking Solutions, https://axper.com/people_counter/, accessed April 2021

[32] Xovis 3D Technology, https://www.xovis.com/en/technology/detail/xovis-3d-technology/, accessed Mai 2021

[33] VCARE: Manage Your Occupancy Limits in Real-time, https://v-count.com/solutions/realtime-occupancy/, accessed Mai 2021

[34] Mandigo: Live-Besucherstrom-Tracking fÃ¼r Veranstaltungen und Messen, https://www.mandigo.de/, accessed June 2021

[35] Mandigo RFID-Development,https://www.mandigo.de/innovations, accessed June 2021

[36] Aucxis Rfid Solutions, https://www.aucxis.com/en/rfid/rfid-technology, accessed Mai 2021

[37] Understanding RFID and RFID operating ranges, https://blog.acdist.com/understanding-rfid-and-rfid-operating-ranges, accessed July 2021

[38] Hisotry of RFID Technology, https://www.trace-id.com/history-rfid-technology/, accessed July 2021

[39] Speedway Connect Capabilities, https://support.impinj.com/hc/en-us/articles/360000044699-Speedway-Connect-Capabilities, accessed July 2021

[40] Gruhler, Andreas, Bruno Rodrigues, and Burkhard Stiller. "A reputation scheme for a blockchain-based network cooperative defense." 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2019.

[41] Reader Modes Made Easy, https://support.impinj.com/hc/en-us/articles/360000046899-Reader-Modes-Made-Easy accessed July 2021

[42] InfluxDB data elements, https://docs.influxdata.com/influxdb/v2.0/reference/key-concepts/data-elements/, accessed July 2021

[43] AWS SDK for JavaScript, https://aws.amazon.com/sdk-for-javascript/, accessed July 2021

[44] Node.JS, https://nodejs.org/en/, accessed July 2021

[45] What is Python?, https://www.python.org/doc/essays/blurb/, accessed July 2021

[46] Why Should You Learn Python For Data Science?, https://analyticsindiamag.com/why-should-you-learn-python-for-data-science/, accessed July 2021

[47] Pandas, https://pandas.pydata.org/, accessed July 2021

[48] NumPy, https://numpy.org/, accessed July 2021

[49] SciPy, https://www.scipy.org/, accessed July 2021

[50] Query and code together with Flux, https://www.influxdata.com/products/flux/, accessed July 2021

[51] SQL (Structured Query Language), https://searchdatamanagement.techtarget.com/definition/SQL, accessed July 2021

[52] Monolith vs Microservices vs Serverless and what to choose for your business needs, https://medium.com/ni-tech-talk/monolith-vs-microservices-vs-serverless-and-what-to-choose, accessed July 2021

[53] AWS Lambda Use Cases: When to use Lambda layers, https://lumigo.io/blog/lambda-layers-when-to-use-it/, accessed July 2021

[54] Get started with Flux, https://docs.influxdata.com/influxdb/v2.0/query-data/get-started/, accessed July 2021

[55] InfluxDB Client JS, https://github.com/influxdata/influxdb-client-js, accessed July 2021

[56] Write data to InfluxDB, https://docs.influxdata.com/influxdb/v2.0/write-data/, accessed July 2021

[57] SAM CLI, https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-start-api.html, accessed July 2021

[58] Pandas DataFrame corrWith, https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corrwith.html, accessed July 2021

[59] Pearson Correlation Coefficient, https://www.spss-tutorials.com/pearson-correlation-coefficient/, accessed July 2021

[60] Kendall Rank Correlation Explained, https://towardsdatascience.com/kendall-rank-correlation-explained-dee01d99c535, accessed July 2021

[61] Spearman's Rank-Order Correlation, https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php, accessed July 2021

[62] Synchronize and Set the Clock on Speedway RAIN RFID Readers, https://support.impinj.com/hc/en-us/articles/202756558-Synchronize-and-Set-the-Clock-on-Speedway-RAIN-RFID-Readers, accessed July 2021

Abbreviations

API	Application Programming Interface
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
EM	Electromagnetic
EPC	Electronic Product Code)
ETL	Extract Transform Load
GDPR	General Data Protection Regulation
HF	High Frequency
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
LF	Low Frequency
RFID	Radio-Frequency Identification
UHF	Ultra High Frequency
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
RDS	Relation Database Service
S3	Simple Storage Service
SQL	Structured Query Language

List of Figures

2.1 Mandigo: example of visitor flow measurement [34] 8

2.2 VCARE occupancy recommendation [33] 9

2.3 Sensalytics path analytics [29] 10

2.4 Active and passive RFID tags [36] 12

2.5 RFID Far-Field and Near-Field. 13

2.6 RFID frequencies [36] 14

2.7 Passive RFID tag backscatter [15] 14

2.8 Impinj xArray R680 16

2.9 xArray beam pattern 16

2.10 Xovis PC2 3D camera 18

2.11 Milesight IoT UR32 and UR35 routers 18

3.1 An illustration of a typical implementation of C-Counter 2.0 consisting of a 3D camera and a RFID reader 21

3.2 Architecture overview of *C-Counter 2.0* 22

3.3 Data Collection Overview of *C-Counter 2.0* 23

4.1 DeviceHub portal 26

4.2 Ursalink port mapping 26

4.3 Impinj xArray gateway 27

4.4 Reader profile & location configuration 29

4.5 SWC: Reader modes data output 29

4.6 SWC: output connection 29

4.7 Levels of modulation taken from [41] 30

4.8 Location mode recomendation from [41] 31

4.9 The TICK stack 34

4.10 Typical serverless architecture taken from [52] 39

4.11 RFID readers and 3D cameras reference systems. 44

4.12 3D Camera and RFID reader data matching algorithm explained with a sequence diagram 47

4.13 AWS CloudWatch lambda function's metrics 50

4.14 Dashboard control panel with variables. 50

4.15 Scatter plot of last known EPC's position. 51

4.16 Heatmap of the recorded EPC's position. 52

4.17 Dashboard control panel with variables. 52

5.1 Heatmap of max throughput 55

5.2 Mean of max throughput 55

5.3 Frame of a recording used to manually evaluate the system. 55

5.4 Maps from a 3D camera and a RFID reader of a single person walking in the scene. 56

5.5 Maps from a 3D camera and a RFID reader of a single person wearing a RFID tag walking between a group of people. 57

5.6 Example 1: Maps from a 3D camera and a RFID reader of a single person walking between a group of people where everybody is wearing a RFID tag. 59

5.7 Wrong assignment of an RFID tag to a person standing still surrounded by a group of people. 60

5.8 Count of visitors from 3D cameras and RFID readers during a test session 61

A.1 Heatmap of auto static dense reader 82

A.2 Mean of autoset static dense reader 82

A.3 Heatmap of autoset static fast 82

A.4 Mean of auto static fast 82

A.5 Heatmap of auto set custom 82

A.6 Mean of auto set custom 82
A.7 Heatmap of auto set dense reader deepscan 83
A.8 Mean of auto set dense reader deepscan 83
A.9 Heatmap of dense reader M4 83
A.10 Mean of dense reader M4 83
A.11 Heatmap of dense reader M8 83
A.12 Mean of dense reader M8 83
A.13 Heatmap of hybrid M2 84
A.14 Mean of hybrid M2 84
A.15 Heatmap of nax niller 84
A.16 Mean of nax niller 84

B.1 Terminal after InfluxDB has been started 85

List of Tables

4.1 Impinj Speedway RAIN RFID reader modes 31
4.2 RFID data example 33
4.3 Example of a 3D camera dataset 45
4.4 Example of a RFID reader dataset 45
4.5 Merged dataframes with tolerance of 1 second 46
4.6 Merged dataframes with tolerance of 1 second and with NaN values 46
4.7 Final dataset containing a match between a personId and a RFID tag . . . 46

5.1 Average count of entries and number of EPCs found for each reader mode 54

Listings

4.1 Flux query to get the lastSeen timestamp of all the RFID readers 36
4.2 Xovis 3D camera data format and RFID reader data format in location mode 40
4.3 Parse RFID reader request in a lambda function 41
4.4 Store RFID reader data in InfluxDB 42

Appendix A

Evaluation Results of Reader Modes

This section contains additional evaluation results. Figure A.1 to figure A.16 illustrate the evaluation performed between different reader modes offered by the Impinj xArray R680 RFID Reader. A detailed comparison has been given in chapter 5.

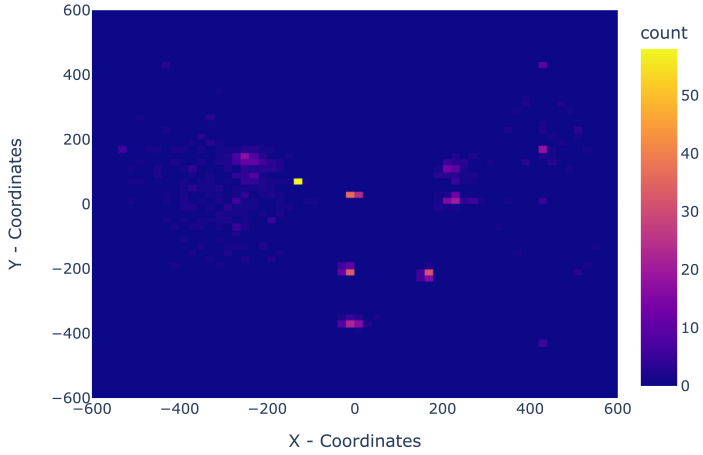

Figure A.1: Heatmap of auto static dense reader

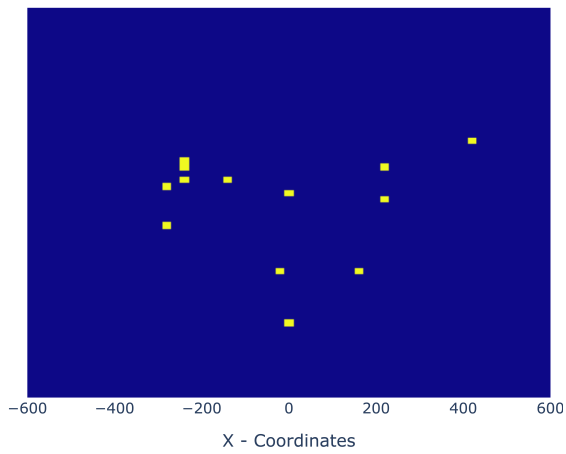

Figure A.2: Mean of autoset static dense reader

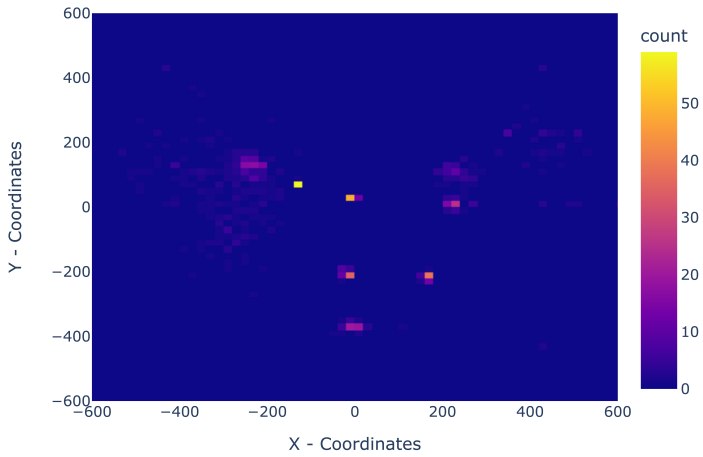

Figure A.3: Heatmap of autoset static fast

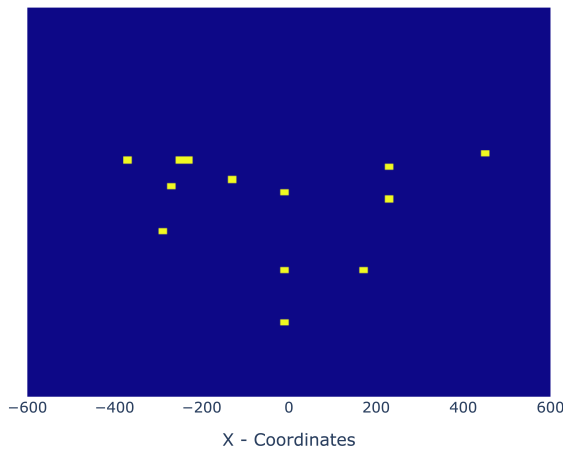

Figure A.4: Mean of auto static fast

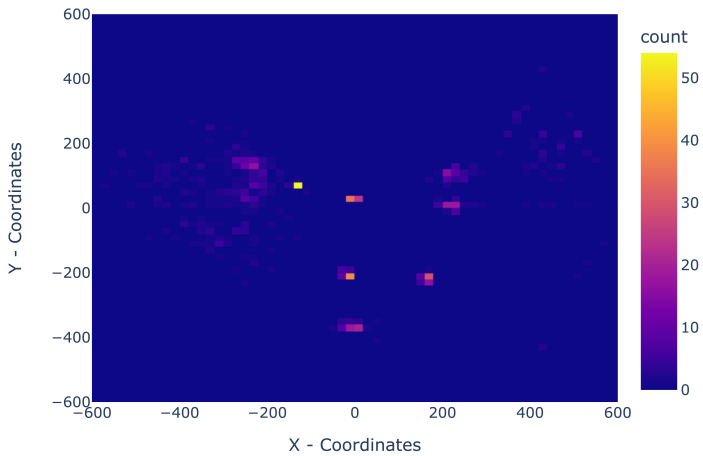

Figure A.5: Heatmap of auto set custom

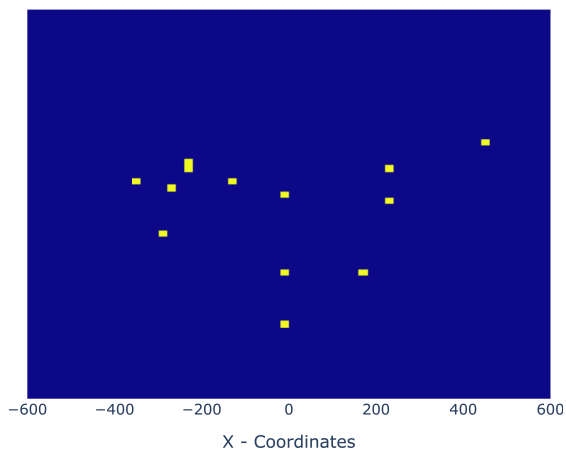

Figure A.6: Mean of auto set custom

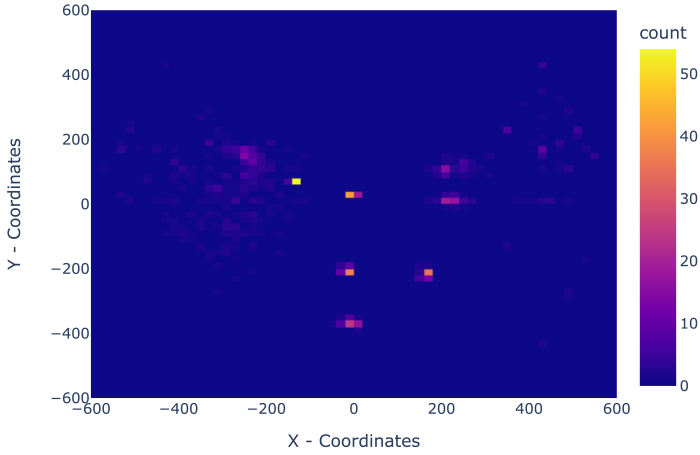

Figure A.7: Heatmap of auto set dense reader deepscan

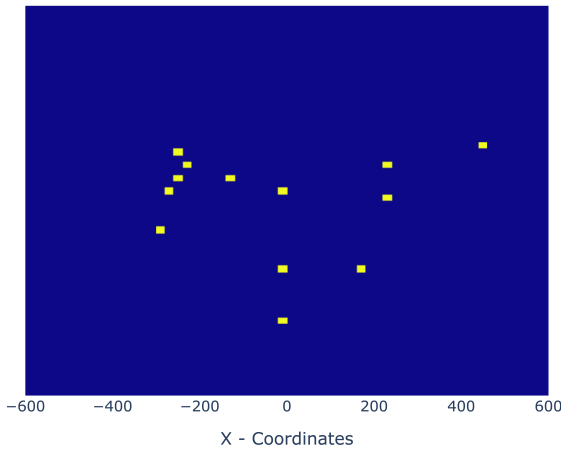

Figure A.8: Mean of auto set dense reader deepscan

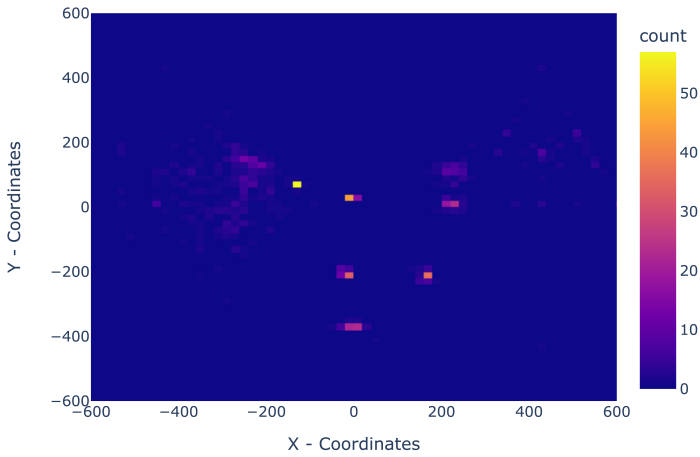

Figure A.9: Heatmap of dense reader M4

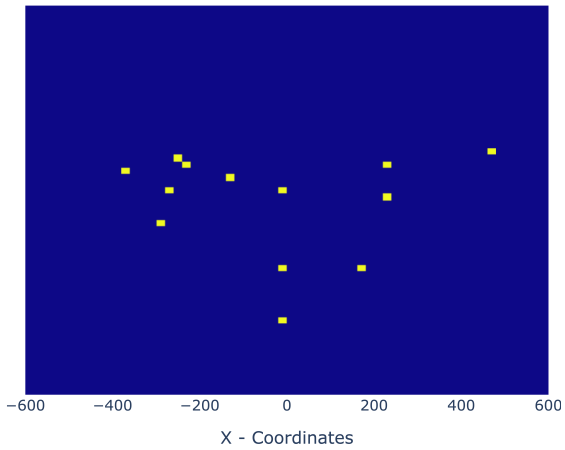

Figure A.10: Mean of dense reader M4

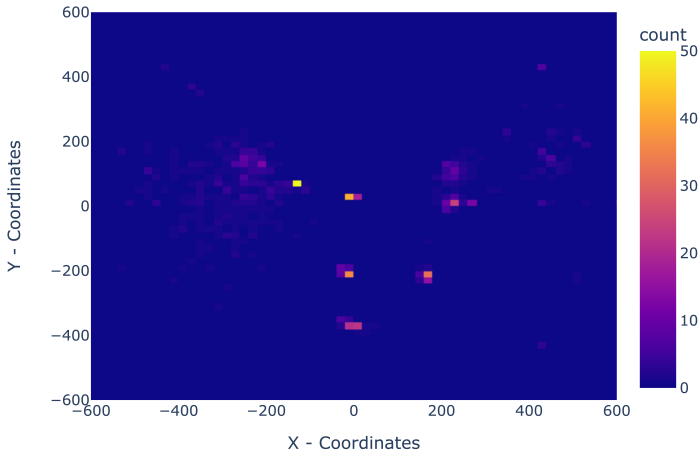

Figure A.11: Heatmap of dense reader M8

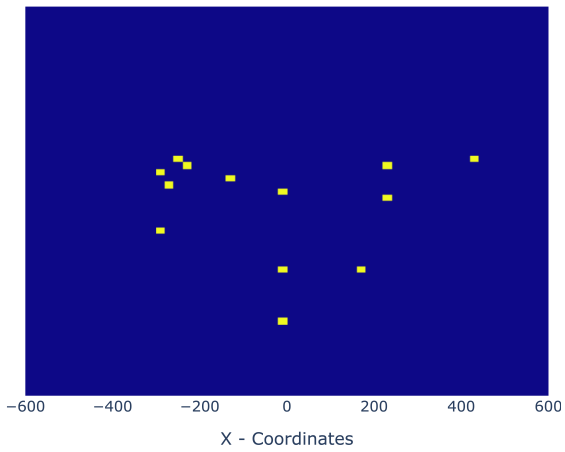

Figure A.12: Mean of dense reader M8

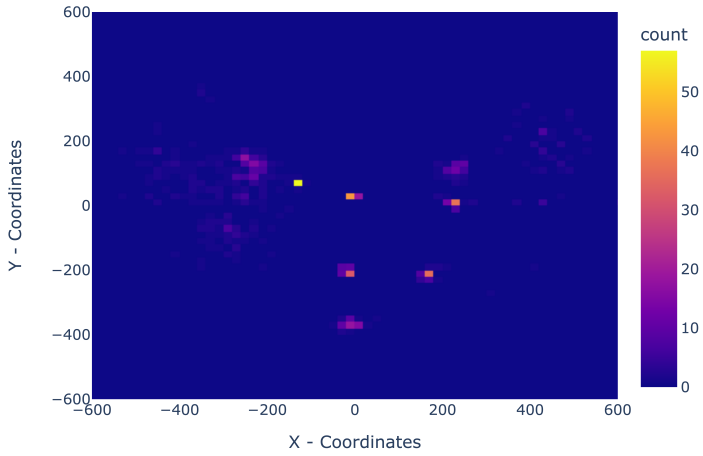

Figure A.13: Heatmap of hybrid M2

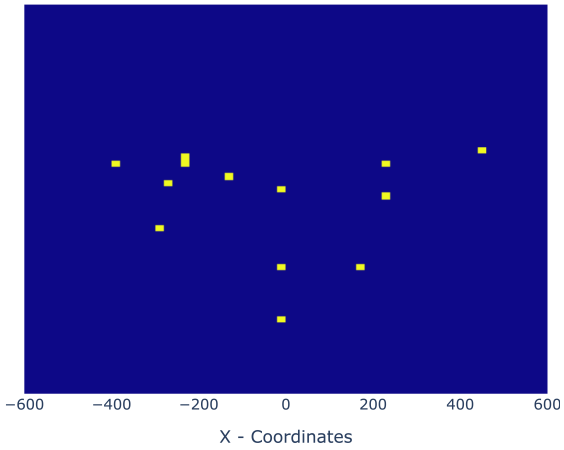

Figure A.14: Mean of hybrid M2

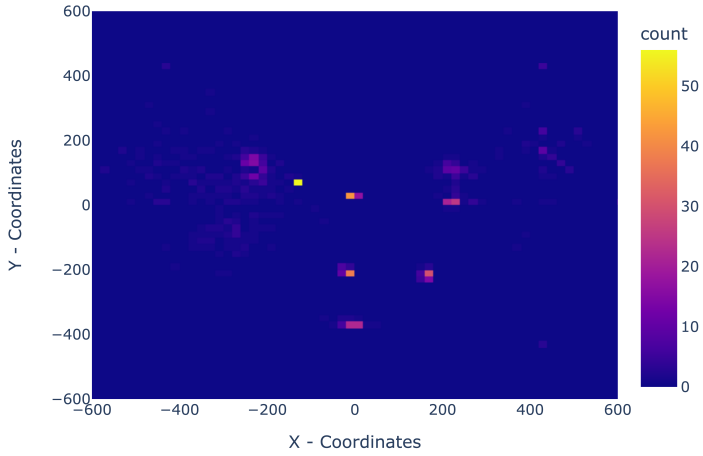

Figure A.15: Heatmap of nax niller

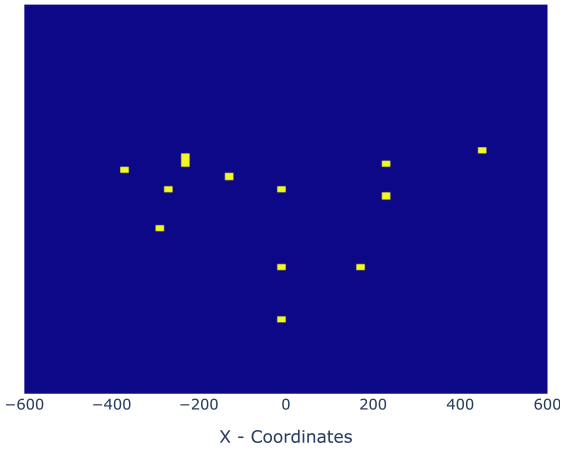

Figure A.16: Mean of nax niller

Appendix B

Installation Guidelines

This installation guideline is based on a MacOS system and may differ from the setup on a Windows or Linux system.

1. **Initial setup for InfluxDB**

   (a) First of all, download the latest version of Docker and start it: `https://docs.docker.com/docker-for-mac/install/` (MacOS), `https://docs.docker.com/docker-for-windows/install/` (Windows), `https://docs.docker.com/install/linux/docker-ce/ubuntu/` (Linux).

   (b) Once Docker has been successfully started, a local instance of InfluxDB can be created with the following command in the terminal:

   ```
   $ docker run -p 8086:8086 -v influxdb2:/var/lib/influxdb2 influxdb:2.0
   ```

   The docker version of InfluxDB 2.0 will be pulled and started, the output on the terminal should look as follows:

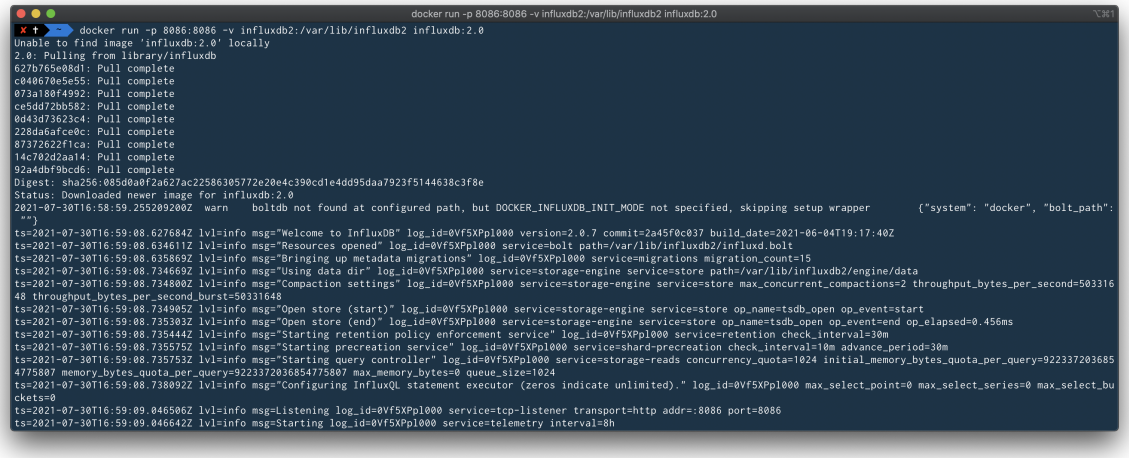

Figure B.1: Terminal after InfluxDB has been started

2. **Initial setup for AWS**
   (a) Install AWS CLI
   (b) Install the AWS SAM CLI
   (c) add your AWS credentials to ‘ /.aws/credentials‘
   (d) add the region and default to ‘ /.aws/config‘
3. **Clone Github Repository**
   (a) Download the project repository hosted on github: `https://github.com/ilecipi/cloud-counter-2.0` (for the credentials ask to Ile Cepilov).
4. **Deploying the application on AWS**
   (a) Navigate into the root of the project and install all the dependencies of each microservice with the following command:

```
$ npm run installAll
```

   (b) Navigate and open the following file `InfluxDB/app.js`
   (c) Insert your InfluxDB credentials: `token`, `org`, `bucket` and `url`.
   (d) Run the following command to deploy a new stack containing the application to AWS, once the stack has been deployed, you can get the IP address on the AWS console and use it:

```
$ npm run build && npm run cloudcounter
```

   (e) Run the following command to run the backend locally:

```
$ npm run local
```

Appendix C

Contents of the CD

- Abstract and Zusammenfassung:
  - *Abstract.txt* & *Zusfsg.txt*
- Thesis (PDF and LaTeX source code):
  - *MA_Ile_Cepilov.pdf* & *MA_Ile_Cepilov.zip*
- Midterm presentation slides:
  - *MA_Ile_Cepilov_midterm_presentation.pptx*
- Application source code:
  - *Cloud-Counter-2.zip*