INSTITUTO
UNIVERSITÁRIO
DE LISBOA

IoT system for the validation of conditions in shipping couriers

Pedro Miguel Moreira Filipe

Master in Telecommunications and Computer Engineering

Supervisor:
Doutor Carlos Eduardo Dias Coutinho, Assistant Professor
Iscte - Instituto Universitário de Lisboa

November, 2022

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

IoT system for the validation of conditions in shipping couriers

Master in Telecommunications and Computer Engineering

Pedro Miguel Moreira Filipe

Supervisor:
Doutor Carlos Eduardo Dias Coutinho, Assistant Professor
Iscte - Instituto Universitário de Lisboa

November, 2022

Agradecimentos

First, I want to thank my family for their unwavering support throughout my academic years, and for making it possible for me to pursue my academic goals. If not for their support and encouragement, it wouldn't have possible for me to reach the point where I am today.

I want to thank my friends for offering their help and ideas in moments where I couldn't progress during the dissertation. I also want to thank their companionship in my personal life, especially in the tougher times.

To my supervisor, Carlos Coutinho, I want to thank the time invested in me, the help provided with structuring this dissertation, ideas suggested and for making me pursue academic publications.

Finally, I want to thank Bruno Mataloto. Bruno acted as a co-supervisor all throughout the dissertation, offering his help by helping me organize and proofreading the thesis. He also offered his knowledge with the hardware and software used, clarifying any questions I had.

Resumo

O aumento da atividade online nos dias de hoje tem causado um aumento no número de lojas de comércio online. À medida que a experiência de compra se torna menos pessoal, algumas pessoas tentam abusar negócios online com fraudes de devolução. O número de dispositivos IoT também tem crescido devido à quarta revolução industrial, ou Indústria 4.0, que consiste na automação de processos e troca de dados na indústria através de IoT e *machine learning*.

Esta dissertação inclui questões de investigação, hipóteses, objetivos e uma metodologia para o desenvolvimento de um sistema que integra IoT para resolver o problema de devoluções fraudulentas.

O protótipo desenvolvido monitoriza quando encomendas saem das instalações e quando são entregues, para além de permitir a verificação da integridade física das encomendas nestes mesmos momentos. Isto é alcançado com o uso de um Raspberry Pi com uma câmara associada e um ESP32 com um sensor de movimento, conectados entre si através do protocolo MQTT e Node-RED.

Para avaliar o sistema, vários testes foram criados de forma a simular um cenário de aplicação no mundo real. Os resultados obtidos permitem concluir que o desenvolvimento foi um sucesso e que o protótipo pode ser usado em empresas de logística, distribuição e transporte para evitar fraudes de devolução.

Palavras-chave: IoT, Serviços de logística, Monitorização de encomendas, Integridade de encomendas, Indústria 4.0

Abstract

The growth in online activity these days has caused an increase in the number of online shopping businesses. As the shopping experience becomes less personal, some people try to abuse online businesses with return fraud. The number of IoT devices has also experienced growth due to the fourth industrial revolution, or Industry 4.0, which consists of process automation and data exchange in the industry through IoT and machine learning.

This dissertation includes research questions, hypotheses, objectives, and a methodology for the development of a system that integrates IoT to solve the problem of fraudulent returns.

The prototype developed monitors when packages leave the facilities and when they are delivered, in addition to allowing to check upon the packages' integrity at these same moments. This is achieved by using a Raspberry Pi with a camera attached and an ESP32 with a motion sensor, connected via the MQTT and Node-RED protocol.

To evaluate the system, several tests were created to simulate a scenario of real-world application. With the results obtained, it is possible to conclude that the development was a success and that the prototype can be used in logistics business to prevent return fraud.

Keywords: IoT, Logistics services, Package monitoring, Package integrity, Industry 4.0

Contents

List of Figures

List of Tables

Glossary

API - Application Programming Interfaces

CSI - Camera Serial Interface

CSS - Cascading Style Sheet

CPU - Central Processing Unit

DSRM - Designated Science Research Methodology

DS - Design Science

EPC - Evolved Packet Core

FL - Forward Logistics

GPRS - General Packet Radio Service

GPIO - General Purpose Input/Output

GPS - Global Position System

GSM - Global System for Mobile

HTML - HyperText Markup Language

IS - Information Systems

IDE - Integrated Development Environment

IoT - Internet of Things

IP - Internet Protocol

JMS - Java Message Service

LCD - Liquid Crystal Display

MQTT - Message Queuing Telemetry Transport

MIPI - Mobile Industry Processor Interface

NFC - Near Field Communication

OpenCV - Open Computer Vision

OCR - Optical Character Recognition

PIR - Passive Infrared

PIC - Peripheral Interface Controller

PRISMA - Preferred Reporting Items for Systematic Reviews and Meta-Analyses

RFID - Radiofrequency Identification

RMI - Remote Method Invocation

RL - Reverse Logistics

SOAP - Simple Object Access Protocol

SQL - Structured Query Language

SCM - Supply Chain Management

TCP - Transmission Control Protocol

TCP/IP - Transmission Control Protocol/Internet Protocol

UAV - Unmanned Aerial Vehicles

WLAN - Wireless Local Area Network

WSN - Wireless Sensor Network

WTB - Where's The Bear

CHAPTER 1

Introduction

In today's era, online engagement has grown significantly [1] and has become a staple in modern society. Online classes have higher enrollment numbers [2] due to hybrid solutions between online and offline learning. Online dating has become a common practice [3], with 30% of single adults of the United States and 25% of young Europeans reporting using dating applications. E-commerce, defined as the exchange of tangible and intangible goods which includes all aspects of the trade process such as online marketing, ordering, payment, and delivery [4], has grown gradually over the years [5], making online shopping increasingly common [6].

Online shopping businesses rely on delivery services to distribute goods to their customers. The delivery services industry represents $1.4 trillion, or nearly 11% [7], of the United States (US) Gross Domestic Product (GDP), where GDP is a measure of the monetary value of goods and services generated in a country over a period [8]. A good shipping service is critical for a consumer, as evidenced by research [9][10], which found that shipping is the most significant part of online shopping for customer satisfaction. Within the delivery process, package handling/condition is a make-or-break factor for online retailers. Study [11] confirms that the condition of the package delivered, this is, if it is damaged or not, is of utmost importance for customer satisfaction. Furthermore, research [12] of users' review comments revealed that inadequate handling of goods is the most frequent topic.

Online engagement has driven the number of Internet of Things (IoT) devices up as well, as a projected growth of 150% from 2017 to 2020 shows [13]. IoT is a global network according to standard and interoperable communications protocols, in which devices, physical or virtual, are connected to each other and integrated into the information network [14]. IoT has various applications in everyday life like cities [15], emergencies

[16], agriculture [17][18], homes [19][20], healthcare [21][22], retail and logistics [23], and much more. IoT has a lot of potential to create solutions to problems never fixed before, like poor handling of delivered goods and improving quality of life.

1.1. Motivation

When a product delivered is reported damaged by a customer, and the customer wishes to return it, it triggers a multistep process of handling said product, also known as Reverse Logistics (RL) [24], in the Supply Chain Management (SCM).

SCM [25][26] refers to the coordination of material and information flows between installations of suppliers, manufacturers, and distribution. These three installations also compose the main stages of SCM: supplying, manufacturing, and distribution. Additionally, customers can also be considered as a stage since information can flow in the form of customer feedback. RL is the backwards flow of materials and information in SCM, to a point of recovery or suitable disposal [27], as opposed to Forwards Logistics (FL). FL is the forward flow of materials and information in SCM, to a point of delivery to a customer. Fig. 1 illustrates an example of a supply chain and the processes FL and RL.

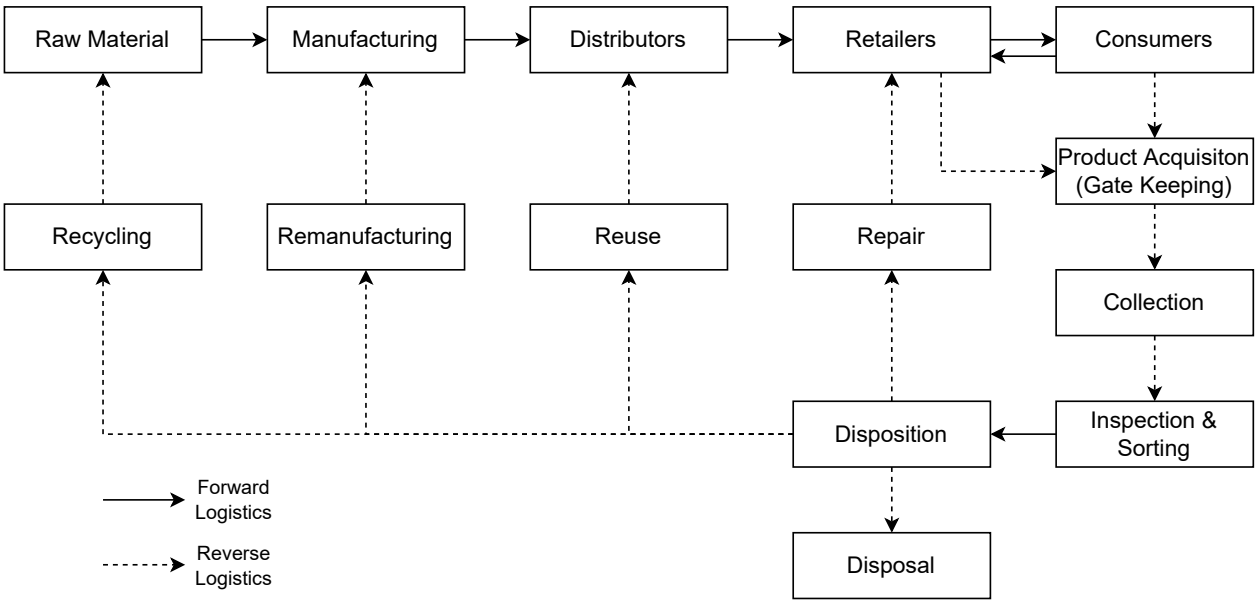

Figure 1. Example of a supply chain and their FL and RL processes (adapted from [24])

Damage caused to a product can occur anywhere along the FL line, however for courier services their responsibility for the package lies within the distribution stage. For a delivery service to be liable for package damage, it means that the package was damaged during delivery. Considering that the package wasn't damaged before distribution, it leaves only two possibilities: The delivery driver damaged it, or the customer did. This becomes a case of good faith, and without any compromisable proof to settle the dispute, return policies cannot be enforced, leaving the business with a tough decision: deny the return and leave the customer dissatisfied, or allow it and lose out on revenue. This is a global issue; research studies demonstrate that in the clothing industry 50% of returns are fraudulent, and 8% of all returns in North America are fraudulent, rendering return policies useless [28].

IoT is utilized to improved various aspects of modern life and can be very important for the development of solutions to unresolved problems in a variety of fields, including retail and logistics. A system based on IoT could be crucial to support delivery businesses with fraud returns, or at very least validate the transportation conditions under which products are delivered. This system could prove beneficial for both consumers and businesses.

From a business standpoint, courier services are the ones that stand to benefit the most from the system previously described. The system can offer non-repudiation as it validates the state of the product before it is delivered to the customer. This means that if the customer attempts to return a product due to mishandling, the system can provide proof to enforce return policies. With this improvement, courier businesses may reduce revenue lost on returns while maintaining customer satisfaction. Aside from that, data collected by the system can serve as a monitoring tool. For example, if a driver has a high rate of damaged packages, it may indicate poor driving. Given the number of packages delivered per day by courier services [29], that data can be used as a measure of a driver's performance.

The system's non-repudiation also extends to customers. For example, if a product is delivered damaged, the system will recognize it as such, meaning that the customer would be in their right to return the item. The monitoring of drivers also benefits the

client indirectly since it assures proper delivery by serving as a deterrent to improper driving.

1.2. Context

Delivery companies' success relies on their image, and ultimately, on their customers. Developing ideas that improve delivery, correlates directly with customer satisfaction, which correlates with the success of the company. Some studies have been conducted to improve logistic services employing IoT.

Study [30] employed REDTags, an IoT-enabled device network that allows delivery service personnel to verify the status of a product at each step of the delivery process. A framework was built to track and monitor the items, with back-end features for smart data transmission, administration, storage, and analytics. The framework provided a dynamic view that was suited to individual stakeholders, allowing for instant feedback and model changes. Machine learning was used to predict potential damage of items packaged, yielding good results. However, attaching these tags to each package can be time consuming, and verification in bundled packages may not be possible.

Another study [31] used a programmable IoT system with a decentralized based block-chain platform to track logistics in real-time and assure package safety and quality. The system was equipped with various sensors that recorded temperature, light, humidity, vibration, and geo-position. Radiofrequency identification (RFID) and QR code technologies were also utilized for stakeholders to interact with parcels in transit. By adding more sensors, the platform could be expanded to transport new data, and an open data API was proposed to provide customers with product history.

These studies highlight the importance of IoT for the development of new solutions in the logistics services industry. These two topics, IoT and delivery services, are the ones covered in this dissertation.

1.3. Research Questions

It is critical to detect whether a package has been mishandled before delivering it to a customer to ensure that all parties are treated fairly. During the development of an IoT

system that validates conditions in shipping couriers, the following questions are to be answered:

- How can the system verify the quality of the package before delivery?
- How effective is the system at detecting package damage?
- What further applications are there for the data acquired by the system?

Questions one and two are crucial for the fulfillment of the objectives of the paper. The first question is asked to determine what configuration of the IoT system can be used to verify package conditions. After concluding what configuration to employ, the system's success rate at detecting damage must be evaluated, which is why the second question is proposed.

The third question proposed isn't as important as the previous two. The system developed will acquire a large amount of data regarding parcel conditions, allowing the data to be used for purposes other than determining whether a package is damaged. This research question is complementary since determining applications for the data may improve the system created.

1.4. Hypotheses Definition

Hypotheses are defined as predictive answers to the research questions. With this in mind, three hypotheses were defined to answer the research questions defined in the previous section.

Regarding the first research question, to achieve verification of package conditions before delivering to clients, the system configuration should include a camera so that it can take pictures before delivery. The camera is connected to a microcontroller that can be activated/deactivated it to save battery life. This microcontroller is paired to a network module to transfer the information collected to a database. The database is necessary to store the data and monitor the deliveries. Besides that, the microcontroller should include sensors for tracking.

For the second question, the system aims to correctly identify and verify the conditions of all packages transported, which translates to an efficiency of 100%.

Finally, concerning the third question, the data acquired is going to be utilized as an evaluation tool for the transport trucks drivers' ability. If the system detects too many occurrences of damage, it could indicate poor driving by the person behind the wheel.

1.5. Objectives

The main goal of this study is to implement a system that can validate the conditions of packages into courier services, using IoT. The system can then be integrated into transportation trucks of delivery businesses, helping the business reduce fraudulent returns and, at the same time helping customers ensure a return if they are properly entitled to.

A complementary goal for this study is to use the data acquired by the IoT system as a monitoring tool for driving quality during transportation. Too many damaged packages might be an indicator for poor driving; by identifying this behavior, businesses can limit the number of damaged deliveries and inform drivers of the problem.

1.6. Methodology

The methodology "Designated Science Research Methodology (DSRM)" by Peffers at al. [32] was followed for the development of the suggested work. DSRM provides a framework for design science (DS) research as well as a mental approach for assessment and presentation of DS research in information systems (IS). With it, it is possible to create a solution to an identified problem using an artifact. Peffers et al. [32] described six stages for DSRM:

- Identifying problems and motivation;
- Definition of objectives for a solution;
- Design and development;
- Demonstration;
- Evaluation;
- Communication.

For the application of DSRM, an entry point must first be identified. There are four approaches:

- ”Problem-Centered” - Entry point at stage 1;
- ”Objective-Centered Solution” - Entry point at stage 2;
- ”Design and Development-Centered” - Entry point at stage 3;
- ”Client/Context Initiation Solution” - Entry point at stage 4.

Fig. 2 shows a visual representation of the DSRM process.

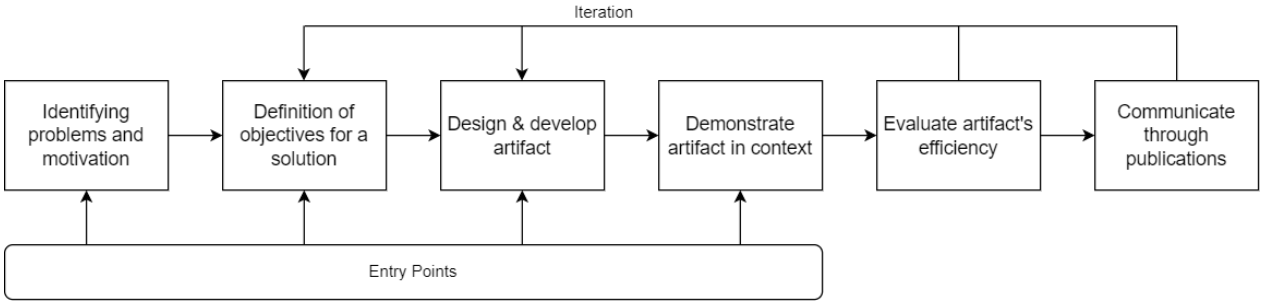

Figure 2. DSRM process [32]

Since the study is based on the problem of fraudulent returns, the entry point for this dissertation is in the first stage.

The first stage of the process was already complete in the section 1.1, where problems were identified, and motivation was given.

The formulation of objectives for a solution, which makes up the second stage, is declared in sections 1.3 and 1.5.

The next stage is designing and developing the artifact. The architecture and functionality shall be decided in this stage, after which the solution will be developed depending on the established objectives. For this dissertation, this phase will consist of designing the IoT architecture and functionalities that will be required to validate package conditions, which is described in chapter 3.

Once developed, the solution will be put to the test. In the fourth stage, a demonstration will be performed by the artifact to solve one or more instances of the problem. For this study, a series of tests will be conducted on IoT system. The tests performed are explained in chapter 4.

Evaluation of the artifact’s efficiency also happens in the chapter 4. The previous stage’s results are analyzed considering the objectives formulated in the second stage.

The suggested solution in this work will be evaluated based on whether it can verify package conditions, as well as its consistency. The artifact's results can also be compared to artifacts of related works.

The final stage consists of communicating the problem and its importance, as well as the artifact via academic publications such as dissertations or professional publications.

Stages two and three can be iterated through to implement a new solution in case of sub-par results obtained in stage five, or new additions for publications in stage six.

1.7. Dissertation Structure

The structure of this dissertation is organized as follows:

- The first chapter introduces the study, its motivation, the study's research questions, hypothesis, objectives, and the methodology used in the dissertation's scope;
- The second chapter is a literature review that describes current state-of-the-art projects in the field of IoT and delivery business. The chapter will contain the methodology applied for selecting studies, a short description of each, and a summary of the research done;
- The third chapter covers the design and implementation of the IoT system utilized to solve the challenge in this study. It will also include the explanation of the components that will be employed throughout the design and implementation of the solution;
- The fourth chapter will test the solution implemented. The findings will be discussed, and the system will be tweaked based on it;
- The fifth and final chapter is composed of the dissertation's conclusions and possible future.

CHAPTER 2

Literature Review

This chapter focuses on reviewing studies related to the topic of the dissertation. The review will provide information on relevant methodologies as well as what has and hasn't been done in this field of expertise. The chapter is structured into four sections:

- Methodological approach and review criteria;
- Related Work;
- Image Processing and IoT;
- Summary.

First, in section 2.1, the approach for finding research related to the dissertation's topic, and the criteria followed to critically analyze and determine if bodies of work are relevant to this dissertation will be explained.

Following that, in section 2.2, studies discovered by the search previously conducted will be meticulously described. This section will provide knowledge about where innovation can be achieved.

Furthermore, projects involving IoT, and image processing are briefly detailed in 2.3. These techniques may prove useful in the dissertation's solution, so a review of publications that incorporate it may prove beneficial.

Lastly, a summary of the studies encountered and reviewed is present in 2.4, to organize all the techniques, protocols, equipment, and so on in the studies reviewed.

2.1. Methodological approach and review criteria

A systematic literature review was conducted using the "Preferred Reporting Items for Systematic Reviews and Meta-Analyses" (PRISMA) [33] statement to find state-of-the-art research in the field of logistics and IoT.

In order to find papers connected to the dissertation's topic, a literature search was undertaken in the Scopus database. This database allows users to search for papers using

queries; this feature was used to improve efficiency in the search for studies. The query executed was the following: *"packages" OR "parcels" OR "goods" OR "merchandise" OR "wares" OR "products" AND "monitoring" OR "tracking" OR "overlooking" OR "supervising" AND "sensors" OR "esp32" OR "raspberry" OR "surveillance" OR "conditions" OR "damage" OR "delivery" OR "iot" OR "internet of things"*.

The articles researched also had to respect the following conditions:

- Language must be English or Portuguese;
- Year of the studies must be between 2016 and 2021;
- In the final stage of publication;
- Must be open access;
- Not related to the subject areas of medicine, chemistry, earth and planetary sciences, biochemistry, genetics and molecular biology, agricultural and biological sciences, and mathematics.

The combination of the search query and the restrictions imposed resulted in 1,148 studies found. From this point, the PRISMA approach was adapted and is described in the flowchart in Fig. 3.

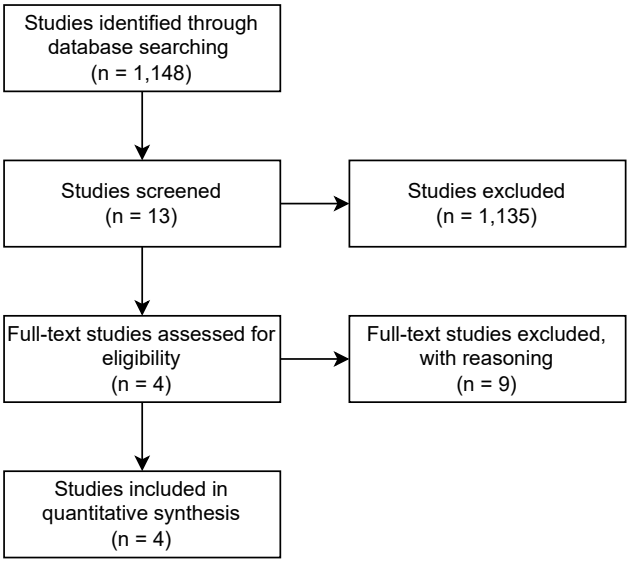

Figure 3. Flowchart for the adapted PRISMA approach [33]

The relevance of the titles and abstracts of the articles found in the Scopus database served as the basis for the study screening criteria. There were 13 studies left after the screening. The remaining papers were examined and assessed for quantitative synthesis suitability. It resulted in the exclusion of nine papers for failing to contribute to the dissertation's theme.

Few studies in the subject of IoT and logistics were discovered after conducting a thorough literature review. The issue was that the keywords selected were insufficiently relevant to include research related to the matter.

To correct this, the systematic literature search was done through Google Scholar. Using this search engine, searches for "package monitoring using rfid" and "supply chain monitoring" were performed. Doing so improved future searches since new keywords were found in the articles from the first searches.

2.2. Related Work

This section describes studies linked to the dissertation's theme, which is the usage of IoT for the improvement of cargo conditions in logistics services.

2.2.1. REDTag: A Predictive Maintenance Framework for Parcel Delivery Services

REDTag Service [30] presents an IoT-based, big data analytics prediction framework adapted to logistics services, to track the condition of transported goods and to early predict potential damages. It uses REDTags, which are smart ad-hoc hardware tags, attached to each package.

The tag is an IoT device containing sensors, batteries, memories, a processor, and a networking module to communicate with external mobile devices, using Near Field Communication (NFC). The device can be fitted with a variety of sensors, such as temperature, humidity, and location, to log a variety of events, like falls or collisions.

The framework consists of front-end and back-end services to store, manage, and analyze REDTag data. It relies on Big Data architecture to store data, as well as a machine learning element to predict potential breaks. Fig. 4 illustrates the proposed framework architecture.

The data and predicted results are shown on dynamic and customizable dashboards, allowing monitorization of packages and quick intervention. Clients can also be alerted of where, when, and who caused damage to packages, motivating drivers to be careful.

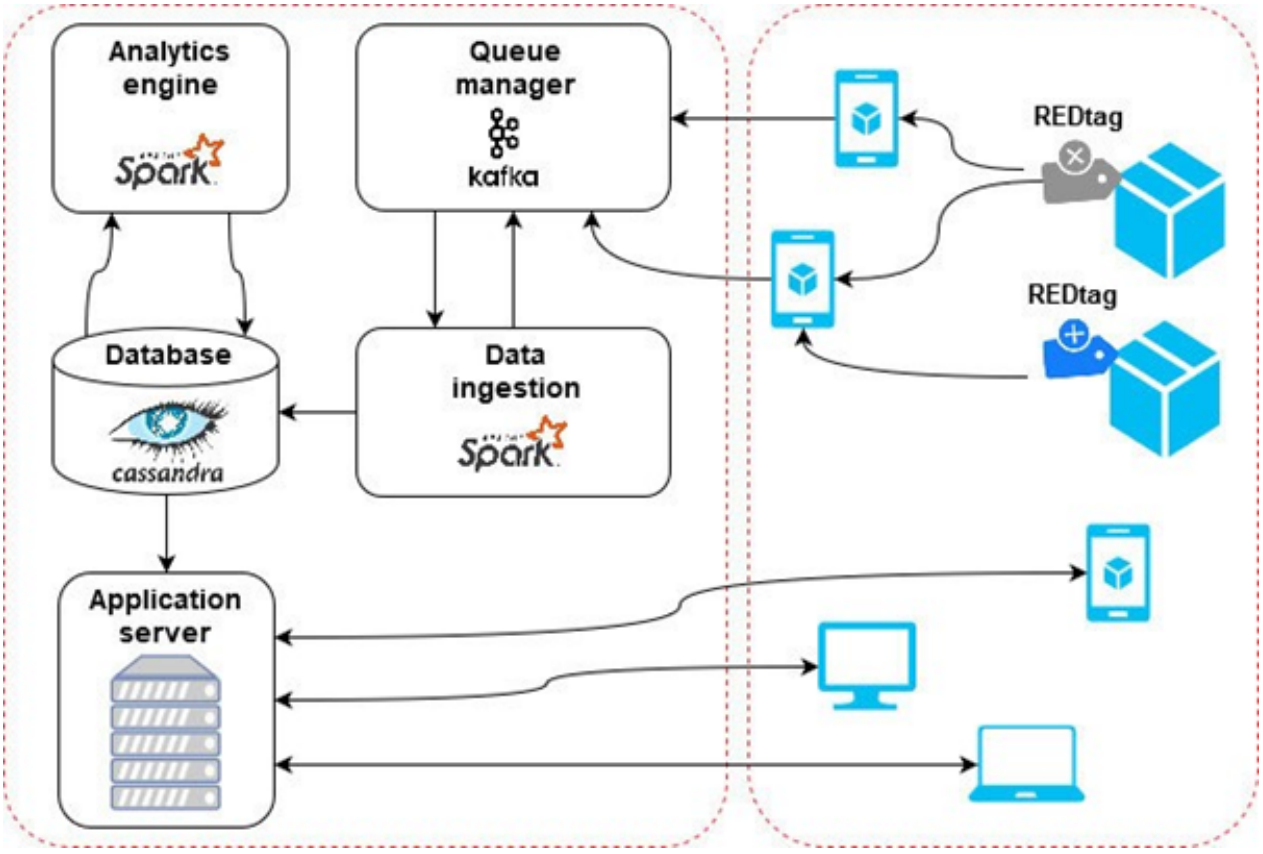

Figure 4. Framework architecture of REDTag Service [30]

2.2.2. Logistics tracking system based on decentralized IoT and blockchain platform

Midaoui et al. [31] provides a solution for a logistics tracking system that combines IoT and blockchain technologies. IoT is utilized as a means to monitor the quality of the products, and the blockchain service is used for data storage and transmission.

The IoT architecture employed in this research is made up of three important modules: Sensor, Connectivity, and Processor. The sensor module contains the light, humidity, temperature, and vibration sensors required in the architecture. To scan RFID tags and identify packages, the connectivity module includes an RFID RC522 tag writer and RFID tag, as well as a 3G modem for communication with the blockchain platform.

The processing module is a Raspberry Pi 4 Model B that connects the two modules and processes information.

The sensors enabled and their respective thresholds are based on the goods transported and obtained by the Raspberry Pi, that fetches a configuration from the blockchain server. These configurations are agreed upon by the logistics services. The system transfers data when a threshold is surpassed, or a certain amount of time has elapsed.

The blockchain platform collects and stores the information received by the IoT system, so it may be displayed in the future. This technology is used due to being a transparent and secure service that ensures data trust and immutability. Fig. 5 shows the architecture suggested in the research.

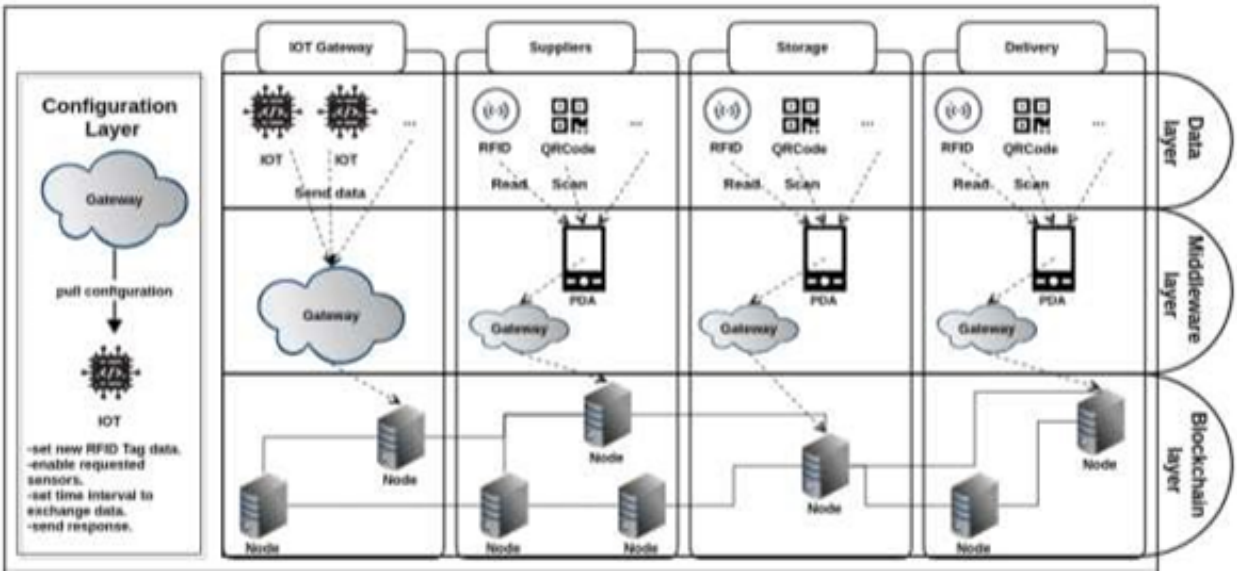

Figure 5. Architecture based in IoT and blockchain technology [31]

2.2.3. Real-Time Monitoring via Patch-Type Piezoelectric Force Sensors for Internet of Things Based Logistics

Chuang et al. [34] suggests the usage of patch-type piezoelectric force sensors that can determine if a parcel has been opened and/or damaged. These sensors release electric charges when mechanical tension is applied. If the parcel is opened, tensive stress is produced, and compressive stress if the parcel collides.

A prototype featuring a microcontroller unit was created so that the analog signals from the sensors can be transformed into digital signals. The digital signals are transmitted via Bluetooth by the microcontroller. Bluetooth was adopted because it is low-cost, low-power, low-interference, and a protocol easy to implement.

Transportation vehicles receive the sent signal through their infotainment system using an application developed, iTape. iTape, shown in Fig. 6, analyzes and monitors the data so that it can be uploaded to a cloud database via 3G/4G communication. The database, presented in Fig. 7 keeps note of the occurrence's time, type and location of the occurrence, so that it may be monitored in real time.

Figure 6. iTape [34]

Sensor Number	Time	Event	GPS
dsfafas132	2015-11-28 20:18:09	Tear Off	
dsfafas250	2015-11-27 18:34:50	Tear Off	
dsfafas186	2015-11-27 16:58:40	Fall Down	
dsfafas943	2015-11-26 19:48:27	Tear Off	
dsfafas751	2015-11-25 11:17:35	Fall Down	
dsfafas483	2015-11-25 09:23:45	Fall Down	

Figure 7. Cloud database [34]

2.2.4. SafeTrack, an intelligent model for logistics management

SafeTrack [35] is a logistics management model that uses geofencing algorithms, RFID technology, and commercial mobile devices. SafeTrack automates: delivery management without user interaction, detection of irregularities in real-time, such as erroneous deliveries or cargo thefts, and monitorization of detours in planned routes. To do this, a component is attached to the rear of the truck with an RFID shield that detects when parcels pass over the RFID reader.

The model provides a website that authorizes administrative functionalities as well as view features for monitored mobile devices movements. It also enables real-time monitoring of ongoing travels and the occurrence of alerts due to detours, low battery, and delivery status, as shown in Fig. 8.

	ID	Travel	Driver	Alarm	Date	Severity
View	45	Palmeira-POA	Rodrigo Remor Oliveira	WARNING!! The device's battery is low !	16/12/2011 16:09:00	Low
View	45	Palmeira-POA	Rodrigo Remor Oliveira	WARNING!! The device is leaving the planned route!	24/11/2011 23:14:20	High
View	45	Palmeira-POA	Rosana Salete	WARNING!! The device is leaving the planned route!	24/11/2011 20:10:40	High
View	45	Palmeira-POA	Rodrigo Remor Oliveira	WARNING!! The device's battery is low !	23/11/2011 23:14:20	Low
View	45	Palmeira-POA	Carlos Emir	WARNING!! The device is leaving the planned route!	23/11/2011 18:30:16	High
View	45	Palmeira-POA	Rodrigo Remor Oliveira	WARNING!! The device's battery is low !	23/11/2011 14:10:30	Low
View	45	Palmeira-POA	Rosana Salete	The device is shutting down	23/11/2011 12:40:10	High
View	45	Palmeira-POA	Carlos Emir	WARNING!! The travel was initialized out of the estimated date!	22/11/2011 16:10:00	Medium
View	45	Palmeira-POA	Rosana Salete	The device is shutting down	20/11/2011 17:09:00	High

Figure 8. Website Monitoring and Management [35]

Package monitoring is accomplished by implementing the *SafeDuino* component, RFID tags and the *SafeTrack Mobile* application in vehicles, as shown in Fig. 9. RFID tags are placed in packages so that they can be traced, and in trucks so that they can be tracked as they enter or depart an installation. The vehicles also signal their position via the Global Position System (GPS) component.

Figure 9. Carrier station architecture for the deployment of *SafeTrack* [35]

SafeDuino is made up of an Arduino, a Bluetooth module, an RFID reader, and a rechargeable battery that is recharged using the vehicle's 12 volts sockets. This component reads RFID tags on packages, but it can't tell whether they're being delivered or deposited. The *SafeTrack Mobile* app solves this problem. The Bluetooth module is used to communicate between the app and the *SafeDuino*.

The *SafeTrack Mobile* app keeps track of shipments and determines if they are being loaded or unloaded. This application operates in the background on a mobile device and feeds information to the *SafeTrack Server* which is done by General Packet Radio Service (GPRS) using Simple Object Access Protocol (SOAP).

The app has two approaches for the loading/unloading problem depending on the number of antennas used. For the single antenna approach, the app recognizes entering or leaving by verifying if the RFID tag has been entered before. For the two antenna approach, one of the antennas is positioned farther inside the truck. The order which by the antennas scan the RFID tags determines the loading or unloading of the package.

2.2.5. RFID GPS and GSM-based logistics vehicle load balancing and tracking mechanism

Prasanna and Hemalatha [36] propose a system that prevents truck overload, tracks deliveries, and detects incorrect delivery of goods. This is achieved with a Peripheral Interface Controller (PIC) microcontroller that contains the followings components: RFID, GPS, Global System for Mobile (GSM), Weight Sensor, Keypad, Alarm, and Liquid Crystal Display (LCD).

The products in the vehicle are tracked and monitored using RFID. Each package is equipped with an RFID tag and the transport truck is equipped with an RFID reader. The driver must read the tag and enter a code into the keypad before delivering it to a customer. If the code is correct, the package can be unloaded; otherwise, the alarm is triggered, and the LCD displays a message.

Vehicle load management is accomplished through the usage of a weight sensor. The sensor registers the weight inside the transport vehicle during loading, and if the

threshold value is exceeded, the alarm in the microcontroller buzzes. The threshold value is assigned previously in the microcontroller. GPS and GSM are used for tracking.

2.2.6. Hybrid Cargo-Level Tracking System for Logistics

Yang et al. [37] developed a hybrid system that combines various types of positioning and communication technologies, and infrastructure-based and infrastructure-less position methods. This system improved availability, accuracy, and overall cost.

The hybrid system offers infrastructures for data transferring and positioning to the tracking devices. These devices acquire data for position computation and environmental sensing, and then transmit the data to a server through the infrastructure provided. After that, the server determines the position of the devices and delivers it to the tracking applications.

The hybrid system offers two types of hybrids. The first hybrid employs GPS, WiFi, ZigBee and RFID for wireless positioning, while GSM, WiFi and Zigbee are used for communicating. The second hybrid leverages existing infrastructure-based/less position technologies in a complementary manner to reduce expenses.

Two types of tracking devices are used in the system. A full-function device includes GPS, WiFi, motion sensor, ZigBee and an RFID tag, allowing infrastructure-based positioning and also infrastructure-less positioning as an anchor node. A dummy device is equipped with a motion sensor and ZigBee,and only supports infrastructure-less positioning; thus, tracking can only be performed with the help of a full-function device.

The intelligent tracking devices have a modular design that allow greater versatility. By removing modules from a full-function device, it can be turned into a dummy one. The devices work for weeks and even months due to power saving of the hybrid design. Furthermore, the devices only update their position when motion is detected.

The server offers services related to the development of smart logistic tracking apps, allowing customers to develop tracking apps that match their needs.

2.2.7. eTracer: An Innovative Near Real-Time Track-and-Trace Platform

eTracer [38] is a platform designed specifically for courier companies that allows near real-time end-to-end monitoring, detection, and tracking of products in transit. It works

with a variety of carriers and forms of transport, improving real-time management in the SCM.

eTracer combines RFID tags with telematics and wireless technologies to collect a variety of information. To obtain this information, eTracer employs various technologies, such as GPRS, GPS and RFID, paired with current protocols and standards, like Evolved Packet Core (EPC), Transmission Control Protocol/Internet Protocol (TCP/IP), Bluetooth and Wireless Local Area Network (WLAN).

RFID tags are attached to parcels. The tags are configured with an RFID reader and a eTracer specific portable equipment. A GPS receiver in the equipment gathers and transfers necessary data for the package's position via GPRS. Vehicles are equipped with RFID readers to detect cargo during loading. These cars also have a GPS receiver and GPRS wireless transmitter to always broadcast their whereabouts.

Logistic facilities have RFID readers positioned in entrances so they may detect packages being moved within the facilities. The data obtained is stored in a local web server through WLAN, which is transferred afterward to the eTracer main server. This server also receives data acquired by the portable equipment through GSM or GPRS.

eTracer stores the data in a relational database so that authorized eTracer users may obtain real-time, automatically updated, and organized information about the timing and position of their deliveries. The eTracer architecture is represented in Fig. 10.

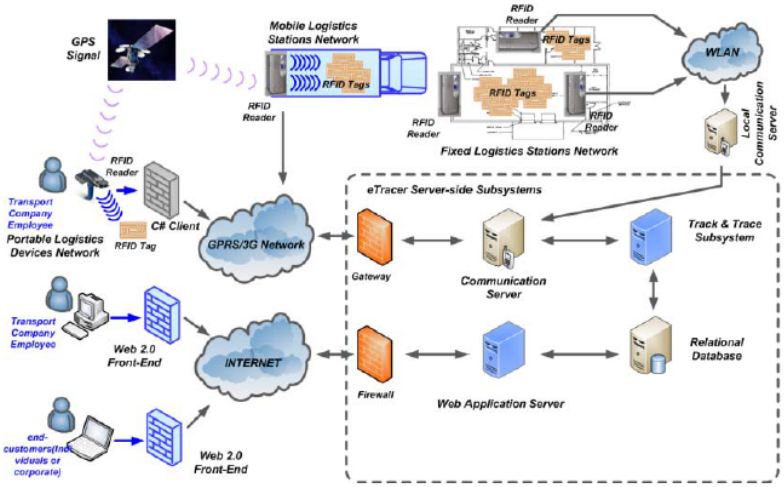

Figure 10. eTracer platform architecture [38]

2.2.8. An Intelligent Logistics Tracking System Based on Wireless Sensor Network

Wang [39] created a system capable of tracking and monitoring parcels within logistics services. There are two layers to this system: server and application. The server layer gathers information about deliveries through various sensors. The application layer converts the information collected into events and stores it in a database.

The server is composed of modules: Sensors, application engine, and communication. The sensor module contains a bar code reader, an RFID reader, terminal mobile devices, wireless sensors, and other sensors that acquire essential data for the application engine. The application engine filters out useless information from the data acquired, creating relevant events for logistical services. The communication module transmits the events to the application layer, utilizing the Transmission Control Protocol (TCP) and SOAP, and Remote Method Invocation (RMI) and Java Message Service (JMS) technologies, where they are stored in a database for further use.

Customers and staff of courier companies can access information on their parcels via text messages and e-mail, as well as geographical information. The study doesn't specify how the sensors are utilized for detecting abnormal behavior nor how the information collected is used to form relevant events.

2.2.9. A Remote Monitoring System of Logistics Carrier Based on Wireless Sensor Network

Wang [40] developed a wireless sensor network (WSN) and GPRS communication-based remote monitoring system. The system is composed of three parts: Wireless sensor network, GPRS transmitter, and a monitoring center. The system's architecture is represented in Fig. 11.

The WSN is mounted in transportation vehicles, and is composed of a ZigBee microcontroller unit, wireless sensor nodes, RFID readers, and a GPRS wireless communication module. Sensors are utilized to gather data about the conditions inside the vehicle and to monitor driving, as ways to ensure package safety. RFID tags are attached to the packages and are scanned automatically by the RFID readers. The information acquired by the sensors is transmitted via ZigBee and then transmitted to the monitoring center

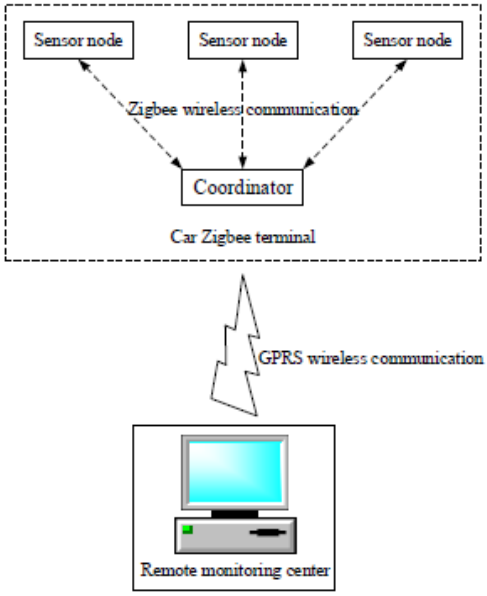

Figure 11. Remote monitoring system architecture [40]

via GPRS communication. The monitoring center analyzes the data received and sends instructions to the vehicle, monitoring it in real-time.

To guarantee optimal conditions during transportation and ensure package safety, the following sensors are utilized: temperature, humidity, brightness, and infrared. Temperature and humidity sensors are employed to collect data on the conditions within the vehicles. The brightness sensor determines if the truck's door has been opened, and the infrared sensor detects whether someone is inside.

2.3. Image Processing and IoT

This section reviews research regarding the use of image processing techniques in the area of IoT.

Dubey and Damke [20] created a non-contact baby monitoring device that detects crying, movement, and position of the infant. A Raspberry Pi 3 Model B+ was used, together with a sound detection sensor module and image processing, to achieve this. To implement image processing the Raspberry Pi camera must be setup so that the baby's head and body are centered. Furthermore, the raspberry operating system and the packages Open Computer Vision (OpenCV), Numpy and a virtual environment must be

installed. The system was trained using a Haar cascade classifier paired with a dataset of positive and negative images of a newborn doll. When the system detects the infant's motion, it sends an alert via e-mail to the system's user.

Where's The Bear? (WTB) [41] is a system that combines cameras, edge clouds, and a back-end cloud to automatically categorize animals in images captured by the system using machine learning-based image processing. WTB employs techniques to decrease resource waste. First, animal classification is done on-site, which reduces the amount of bandwidth required. Second, training data was created by combining backdrop pictures taken at the reserve with images of animals obtained from Google Images, decreasing bandwidth requirements once again. Third, the model is trained on the cloud before transferring it to the on-site system, requiring less processing power. Animal classification was accomplished using Google TensorFlow and OpenCV Optical Character Recognition (OCR), which correctly categorized 87% of produced photos and 94% of real images with a 0.90 certainty or better.

Kapoor et al. [42] describe a method for determining variables that hinder plant growth with the combination of IoT and image processing. This work utilized soil moisture, temperature, and humidity sensors paired with a Serial JPEG camera module and an SD card assembled on an Arduino UNO. The camera snaps photos of crops at regular intervals, which are analyzed to identify morphological changes, and the farmer is notified if they are hazardous to the plant's growth. The image is processed by MATLAB algorithms that are trained with a dataset of pictures previously taken. The algorithms consider not just the picture, but also the environmental conditions and fertilizers.

A very similar study [43] also employed MATLAB algorithms to detect changes in plants. In this study, the IoT system incorporates sensors and a camera assembled in an Arduino. Factors like temperature, humidity, and color, shape, texture, and area of the plant are all considered. MATLAB segments the image, performs edge detection, and analyzes the plant for illnesses. Based on the analysis given by the algorithms, pesticides may be applied by the IoT system in according to the disease diagnosed.

A smart city solution to overcome the impact of traffic congestion is suggested in [44]. It proposes an IoT system that can measure traffic density based on real-time

video and image processing techniques. It employs a Raspberry Pi, a cloud server, and MATLAB. The objective is to determine traffic density on both sides of the road and give the user the ability to control the traffic signal. The solution was achieved by calculating the difference between a reference image, that corresponded to an image of the road without traffic, and images in real-time. This calculation was performed by the Raspberry Pi using image processing techniques available in MATLAB. Furthermore, since color is not important for this process, the image calculated is converted into grayscale and binarized to improve performance.

Sharma et al. [45] suggest a smart city system based on WSN, Unmanned Aerial Vehicles (UAV), IoT, and image processing for early fire detection. Sensors are placed in regions of interest to gather environmental data. Temperature, humidity, light intensity, and smoke information are all collected in real-time by the sensors. Radio Frequency is used to communicate between sensors and to transfer data to the central node, while GPRS is utilized for communication between the central node and a cloud platform. The gathered data is stored in the cloud, where it is displayed and examined by MATLAB for abnormal events. In the occurrence of an abnormal event, the control station is notified, which triggers the UAV to acquire photos of the location. The sensor's GPS coordinates are used to determine the location. The image taken is processed to detect potential fires.

Rane et al. [46] designed a system to validate and manage parking lots. Utilizing a BeagleBone Black microcontroller attached with a Raspberry Pi camera in the entrance, the system validates the entrance to parking spaces. A picture of the license plate is snapped, and the number is converted into text via OpenCV's OCR technique. The converted license plate number is sent to the server, where it is compared with the employees' registered license plate numbers. The server validates the entry in the parking lot, permitting it or denying it. An application is made available so that users can mark where they have parked their cars, and visualize spots taken.

2.4. Summary

The previous studies will be summarized in this section. To do so three tables were created. Table 1 summarizes the key features of the publications discussed in section 2.2. It covers sensors utilized, equipment used, and how communication was established.

Table 1. Summary of the studies reviewed in 2.2

Ref.	Sensors	Equipment	Communication
[30]	Temperature, Humidity, Location	REDTag	NFC
[31]	Temperature, Humidity, Light, Accelerometer, RFID	Raspberry Pi 4 Model B	3G
[34]	Piezoelectric force	Microcontroller unit, iTape	3G, 4G
[35]	RFID, GPS	Arduino	GPRS, Bluetooth
[36]	RFID, GPS, Weight	Keypad, Alarm, LCD, PIC microcontroller	GSM
[37]	RFID, GPS, Motion	Microcontroller	WiFi, ZigBee, GSM
[38]	RFID, GPS	In-house portable equipment	GSM, GPRS, Bluetooth
[39]	Barcode reader, RFID	Not specified	Not specified
[40]	Temperature, Humidity, Brightness, Infrared, RFID	ZigBee microcontroller	GPRS, ZigBee

Table analysis reveals that RFID is very abundant in this sector. In the SCM process, RFID is routinely used to identify packages. Equipment and communication-wise presents more variety on what is used.

The tools used to implement image processing techniques are listed in Table 2. All projects developed either use OpenCV or MATLAB for their image processing needs. On the one hand, most studies that integrated data sensing with image processing, such as [42], [43], [45], utilized the MATLAB software. On the other hand, studies that utilized only images for the development of their solution used OpenCV.

Table 2. Summary of the tools needed for the studies reviewed in 2.3

Ref.	Tools employed
[20]	OpenCV, Numpy, Raspberry operating system
[41]	OpenCV, Tensorflow
[42]	MATLAB
[43]	MATLAB
[44]	MATLAB
[45]	MATLAB
[46]	OpenCV

Finally, table 3 states the main contributions of the studies described in the section 2.2.

Table 3. Main contribution of the studies reviewed in 2.2

Ref.	Contribution
[30]	Package break prediction based on machine learning, data-driven solution with integration of IoT and Big Data frameworks
[31]	Decentralized IoT/Blockchain-based platform to assure transparency and traceability to stakeholders in SCM
[34]	Usage of piezoelectric patch type force sensors that can detect opening or falling of packages to improve real-time monitoring
[35]	Automatic delivery management and an inconsistency detection mechanism in real-time
[36]	Avoiding weight overload of transport trucks and misplacement of packages
[37]	Hybrid system that utilizes infrastructure-based and infrastructure-less positioning techniques for cargo-level tracking
[38]	Providing courier businesses real-time end-to-end tracking of parcels
[39]	Track, visualize and automatically manage information in the delivery process
[40]	Remote supervision of delivery vehicles and transported goods

Many advances in the field of IoT-based cargo monitoring in logistics services have been developed, but few attempt to identify damage to carried products. Study [30] predicts potential damage based on data collected, and [34] recognizes whether a package has been opened or damaged. Other studies focus on tracking, and/or ensuring safe conditions during transportation with sensors.

CHAPTER 3

System Design and Architecture

The solution proposed in this research aims to assist logistics companies in preventing return fraud. This chapter explains the IoT system's design and architecture, which is based on the objectives proposed in 1.5. The chapter is structured in the following way:

- System Requirements;
- System Design and Architecture;
- System Components;

The system's requirements are going to be defined in section 3.1. With established requirements, the system and its architecture can be designed. Section 3.2 contains all of the necessary equipment and applications to fulfill the proposed requirements, and address the issues identified in the preceding section.

Afterward, in section 3.3, a more in-depth description of the equipment and applications are provided, to better understand how the system works.

3.1. System Requirements

The proposed IoT system has two objectives to achieve. The first is to check the state of packages transported by courier services, which technology can be integrated into delivery trucks, benefiting both delivery companies and customers by determining who's responsible for package damage and enforcing return policies.

Regarding this objective, it is important to understand that a great number of packages are transported every day, and if the system keeps track of information on each package, a large quantity of data will be generated. This makes it more difficult to visualize data in the future. Additionally, packages may sustain damage at any stage during delivery.

The second objective is to employ the acquired data as a monitoring tool for driving quality. By recognizing drivers that drive poorly, it may be possible to reduce the number of damaged shipments.

In light of this, the following requirements are essential to complete the first objective:

- Guarantee that all packages are identified and information is stored for future reference;
- Information retrieval about a parcel must be simple;
- It must be possible to check if a delivery was or was not damaged during the delivery process.

For the second objective, the following requirements were identified:

- Package data must be paired with the driver responsible for delivering it;
- Delivery information must be able to be categorized into damaged or not damaged.

3.1.1. Santos e Vale - Logistics, Distribution and Transportation

Santos e Vale [47] is a Portuguese logistics, distribution and transportation company that delivers over 8,500 packages per day, moving over 750,000 tons per year. The business brought up the issue of customers claiming that packages were damaged during shipping when they weren't. The project developed in the dissertation was based on solving this problem to implement it in the corporation's vehicles.

Given the enterprises' involvement in the development of the system, and the goal to integrate it into Santos e Vale's vehicles, the following extra requirements were requested to achieve a successful implementation:

- The system's operation must be autonomous, meaning it cannot interfere with employees' daily activities;
- Package detection is accomplished through QR codes.

Both parties benefit from this partnership. If the development of the system is successful, Santos e Vale can incorporate the system and reduce the number of false package

damage claims. For the researchers, the partnership provides a practical application and confirmation that the system can be used in a real-world setting.

3.2. System Design and Architecture

With the requirements established, the design and architecture of the IoT system can now be defined. The architecture of the system is split into four layers, as shown in Fig. 12: Hardware, Network, Data and Application Layer.

- Hardware Layer - Two sets of components are used as hardware: an ESP32 with a motion sensor and a buzzer, and a Raspberry Pi with a camera module. The motion sensor can detect when the truck cargo door is opened, and with this information the system knows that cargo is going to be loaded/unloaded. Knowing that cargo is going to be transported, the Raspberry Pi, which is the system's most important component, activates the camera so it can detect QR codes and record the loading/unloading process. When a QR code is identified, the buzzer sounds off indicating that video recording has begun. The driver then shows the package to the camera until the buzzer sounds again, indicating that the video recording has finished.
- Network Layer - The Raspberry Pi hosts the main constituent of the network layer, the Node-RED server. The server activates the Message Queuing Telemetry Transport (MQTT) broker, which acts as a mediator for the exchange of information between the ESP32 and the Raspberry Pi. Beyond that, the server stores the information obtained by the QR codes in a database. This information is paired with information about the cargo door state and is displayed to the user via dashboards. The flow of information is guaranteed by the Internet Protocol (IP) and the MQTT protocol. Both the ESP32 and the Raspberry Pi are connected to the truck's WiFi.
- Data Layer - Two elements incorporate the data layer, an SQLite database, and two Python scripts that employ the OpenCV library. The scripts are responsible for detecting QR codes and recording the loading/unloading process. In the case of the QR code detection script, the information obtained is stored in a

database. This database stores information about the identifier of the package, time of detection, type of operation (loading or unloading), and classification of the package (damaged or not damaged). The time of detection gives a rough estimate of when the package was loaded/unloaded. The type of operation is determined using logic, explained at the end of subsection 3.3.6.

- Application Layer - This layer represents the front-end of the system. Node-RED is equipped with tools capable of creating web pages to show all types of information. In this case, two pages are available, one with the information stored in the database and another with information regarding the cargo door.

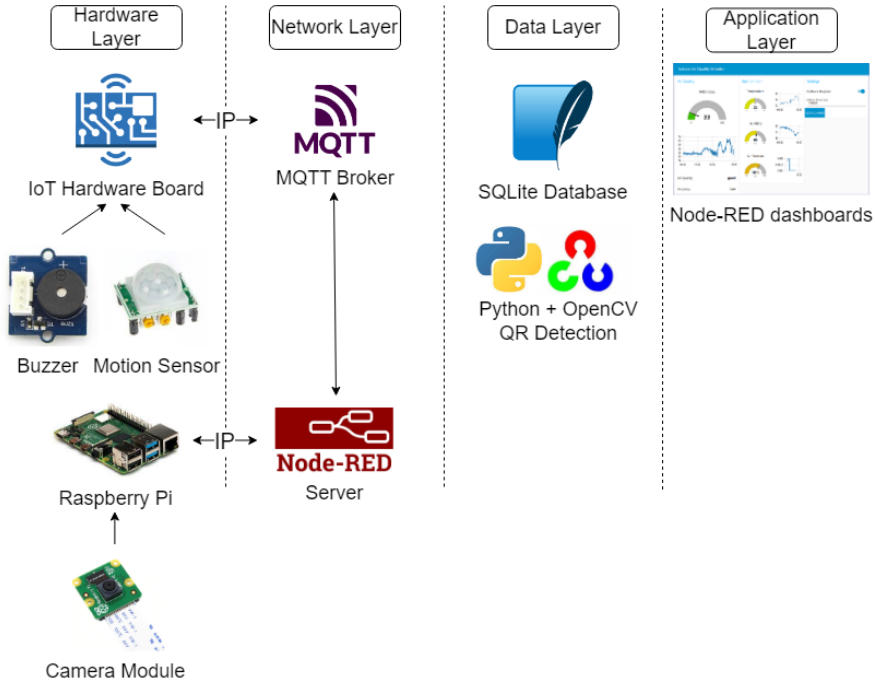

Figure 12. System's architecture and layers

3.3. System Components

This section provides some insight into the components used to build the IoT system in this dissertation. Each component will include a brief description and an explanation of how it fits into the overall system.

For the development of the system, it was established that two devices were needed, a main one and a secondary one. The main device stores data, detects QR codes and

records the loading process, while the secondary device recognizes when the loading process is occurring.

With this arrangement between devices, resource management can be done on the main device. Since the secondary device recognizes when loading has finished, it can relay this information to the main device to interrupt the processes related to QR detection and video recording since it is no longer needed.

Taking this into consideration, two sets of components were picked: a Raspberry Pi with a camera module to take the role of main device and an ESP32 with a motion sensor module and a buzzer as a secondary microcontroller.

3.3.1. Hardware Layer: Raspberry Pi and Camera Module

The choice of hardware for the main device was based on the following factors:

(1) Compactness - How much space it occupies;
(2) Ease of set up - How simple it is to setup;

Regarding factors (1) and (2), since the goal is to integrate the system in delivery trucks, it is advantageous that the components take as little space as possible and that its setup is straightforward so that anyone can perform it. Also, given that communication between the two devices is important to save resources and necessary for the system to work, the hardware choice must be able to connect to other devices.

With this in consideration, the device chosen was a Raspberry Pi 4 Model B. The Raspberry Pi is a pocket-sized computer capable of performing all the tasks that a desktop computer can. Some of the model's key features are its 64-bit quad-core processor, 4K display support, WiFi, Bluetooth, Gigabit Ethernet and H264 encoding up to 1080p 60 fps [48]. Raspberry configuration is also fairly easy, all it needs is a microSD card with the Raspberry operating system installed and it is ready to work with.

Since it works just like a desktop computer, Raspberry Pi has access to most tools that desktop computers have. One of them is Node-RED, which within its many features, supports accessing and storing data in an SQLite database. With this application and database, package data can now be saved for future consultation, however, on its own the Raspberry Pi cannot recognize QR codes and record the loading procedure.

This is where one of the Raspberry Pi's major strengths, its modularity, is convenient. The microcomputer is equipped with a 40-pin General Purpose Input/Output (GPIO) header and a two-lane Mobile Industry Processor Interface (MIPI) Camera Serial Interface (CSI) camera port. This camera port is very helpful since it enables a camera module to be connected to the Raspberry and operate it via Raspberry applications.

To perform QR code recognition and capture the loading process, a camera module was connected to the Raspberry Pi. The camera that used is an official Raspberry Pi Camera Module V2.1, a very small, lightweight camera with eight megapixels of resolution and 1080p video capture. Using Python scripts and the OpenCV library the camera module is able to detect QR codes in packages and film the loading/unloading process.

With this pairing, the main device fulfills all the requirements needed. The device was fitted inside a case for protection and can be seen in Fig. 13.

Figure 13. Raspberry Pi with camera module setup

3.3.2. Hardware Layer: ESP32, Motion Sensor and Buzzer

Following up the choice of the main device is the choice of a secondary device to monitor the status of a delivery truck's cargo door. Knowing when the cargo door is open or shut leads to saving resources on the Raspberry Pi, since the device doesn't have to run the scripts for QR detection and video recording when the door is closed.

The criteria for picking the secondary device were the same as the main one: compactness and ease of set up. Considering this, the ESP32-WROOM-32D microcontroller was chosen for the development of the system. The ESP32 is a powerful, small microcontroller device, with WiFi and Bluetooth modules that can be utilized for a variety of

applications, such as a low-power sensor network. It has a customizable central processing unit (CPU) clock frequency, that may go up to 240 MHz, and a sleep current of 5 μA making it appropriate for battery-powered applications.

These characteristics make the ESP32 suitable for the secondary device. The WiFi module is essential to relay the information about the cargo door status to the Raspberry Pi, however, a motion sensor is required to obtain this data.

The motion sensors researched were Passive Infrared (PIR) sensors. These sensors consist of two parts: a pyroelectric sensor and a Fresnel lens. The Fresnel lens focuses the infrared signals onto the pyroelectric sensor, which in turn detects changes on the infrared levels from the signals received [49].

Study [50] employed a motion sensor in their work and compared the performance of two PIR sensors, SB00422A-1, and HC-SR501. Results showed that the HC-SR501 produced fewer false negatives which led to its choice for this IoT system.

Initially, the system didn't have a buzzer incorporated, but given the results obtained in section 4.2.2 it became clear that it was necessary. The buzzer sounds on two different occasions, when a QR code is successfully scanned so the driver displays the package for the recording, and after the recording stops.

Together, the ESP32, HC-SR501 PIR sensor and buzzer recognize whether the cargo door is opened or closed, relays this information to the Raspberry Pi so it can execute the Python scripts, and informs the driver when it can showcase the package for the video recording. Given that the code is identical and only has to be altered if the GPIO pins vary from setup to setup, the assembly and configuration of this pair is simple. The end product assembled can be seen in Fig. 14.

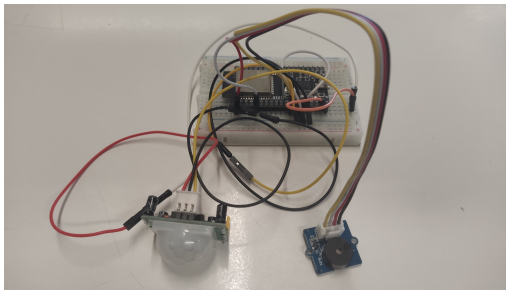

Figure 14. ESP32, PIR motion sensor and buzzer assembly

3.3.3. Network Layer: Internet Protocol and MQTT - Information Exchange

All the information exchanged by the devices is done through IP and MQTT. MQTT is a standard messaging protocol for IoT devices, that runs one layer above IP. Its popularity is due to its small, minimal data packets and ease of implementation. This makes it ideal for connecting remote devices with low bandwidth, low power usage, high latency, low connection costs, and variable availability [51].

This protocol uses a publish/subscribe architecture to exchange information. MQTT clients send a subscribe message to a MQTT broker with the topic they want to subscribe to. Afterward, when a client sends a publish message, it is relayed by the broker to all the clients subscribed to the topic in the publish message. The protocol is also represented in Fig. 15.

In this system's case, two topics are employed, "Motion" and "Buzzer". In the "Motion" topic, messages regarding the movement detected by the motion sensor are sent to the broker, while the "Buzzer" topic sends messages to indicate when to sound the buzzer. The broker then relays the messages to all the devices subscribed to the topics. The "Buzzer" topic sends information from the Raspberry to the ESP32, while the "Motion" topic sends it in the opposite direction, from the ESP32 to the Raspberry.

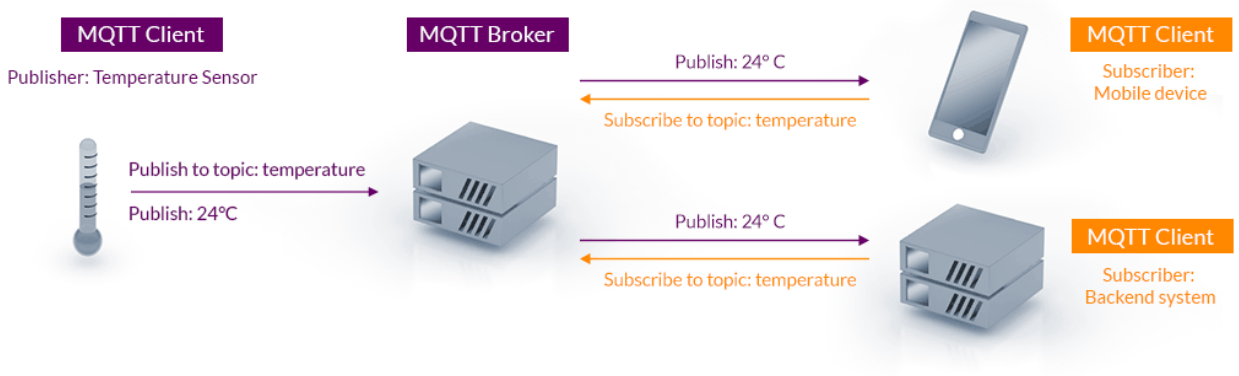

Figure 15. MQTT Protocol [52]

Utilizing these two protocols, a link between the two devices is established where the MQTT broker acts as a mediator.

3.3.4. Network Layer: Node-RED server

Node-RED is an open-source software that connects physical devices, Application Programming Interfaces (API), and web services built on Node.js. It offers a browser-based flow programming editor with various nodes that allow programming without writing code. JavaScript functions are also included for more complex implementations [53].

The Node-RED server was utilized as the main piece of the network layer for two reasons: The simplicity of the flow-based programming and the compatibility with other hardware and software. An example of the Node-RED flow programming is shown in Fig. 16.

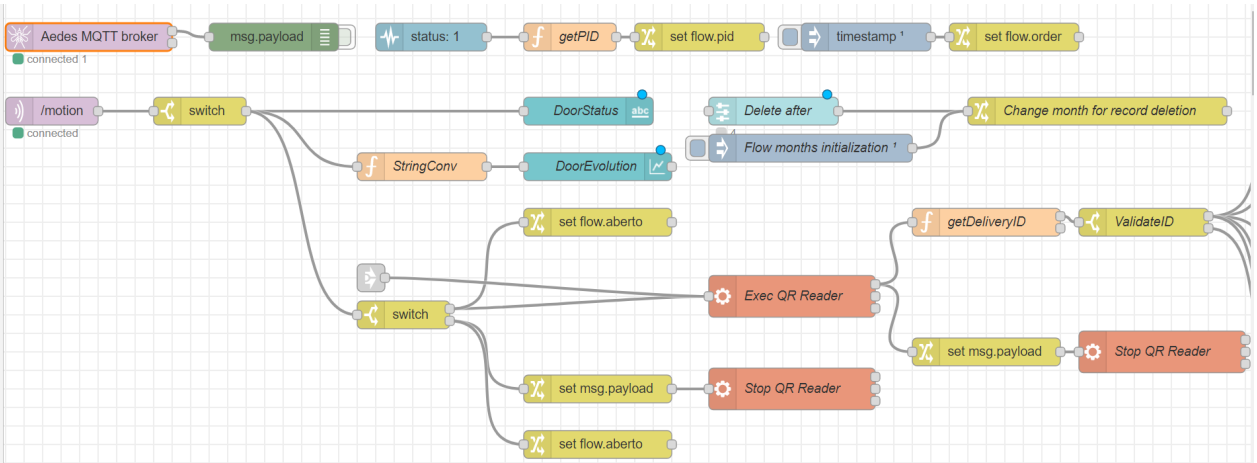

Figure 16. Example of the Node-RED back-end

The server is the backbone of the system; it bridges the ESP32, the Raspberry Pi and the database hosted in the microcomputer. Node-RED interprets the messages sent by ESP32, executes the Python scripts associated with QR recognition and video recording, and stores the information gathered in the database.

A big advantage of Node-RED is that the flow-based programming simplicity isn't back-end restricted. Creating dashboards to visualize data flowing through the server is straightforward with the usage of nodes already included, but HyperText Markup Language (HTML) and Cascading Style Sheets (CSS) are also available to create more complex displays. This advantage is employed for another system feature using Node-RED server: The implementation of two dashboards for data visualization.

3.3.5. Data Layer: Python scripts

Python is an open-source programming language, easy to use with access to various third-party libraries that allows integrating systems more effectively. The Raspberry Pi also comes with an Integrated Development Environment (IDE) of Python pre-installed, Thonny.

Using Python, the camera module can be programmed to be used without human input and instead, be activated when certain conditions are met. The IoT system was developed so that two Python scripts wait to be executed when the necessary conditions are met.

The first script scans the camera's field of view for QR codes. When detected, the information is decoded and submitted to the Node-RED server for validation (If data read on the QR code is a number and bigger than 0). If the validation is successful, the information is stored in the SQLite database for future use.

The second script records a short five second video after the QR code is detected and verified. With the help of the buzzer, the delivery driver knows when the QR code was scanned successfully and should proceed to display the package, so the video recording can capture the package's integrity from all angles. The video is then stored in the Raspberry Pi. The full code of both scripts is located in Appendix A.

The conditions that trigger the execution of these scripts work in tandem. The first execution of the QR detection script is when the cargo door is opened, and the final interruption of the script is when the cargo door is closed. The script's subsequent executions and interruptions are dependent on the number of QR codes identified.

When a QR code is detected and the information is validated, the QR code detection script is interrupted for the second script to execute. The validation of the data is important because if a QR code is not decoded correctly and fails the validation, the code is still running, and it can decode the QR properly.

After the second script is finished, the first one is executed again so it can look for possible QR codes. The flow of scripts execution can be visualized in Fig. 17.

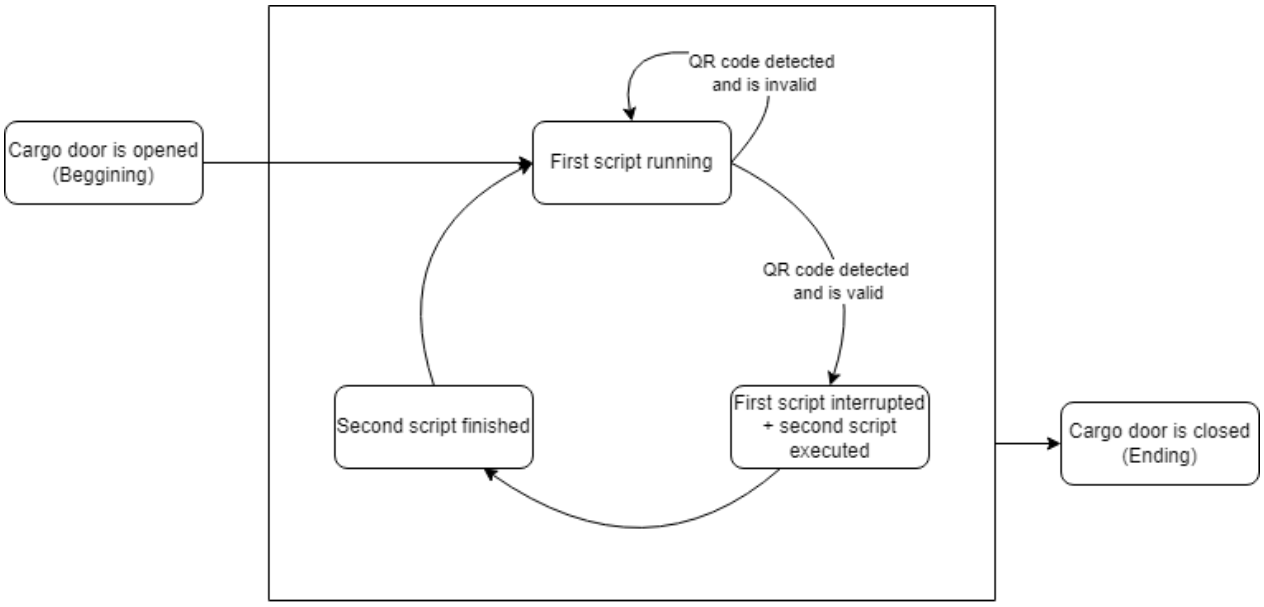

Figure 17. Script execution process

3.3.6. Data Layer: SQLite database

A database was required to store the data collected by the scanned QR codes, so a database was created using SQLite. SQLite is an easy-to-use, lightweight, fast, and reliable SQL database engine [54]. It is suited for embedded and IoT devices due to its small size, and data analysis is simple using Structured Query Language (SQL) commands.

SQLite excels in offering local data storage for individual applications and devices [55]. This makes SQLite a viable contender for the database needed. Pairing this with the fact that the data gathered by QR codes is minimal, and that the database is hosted in the Raspberry Pi, SQLite is the ideal database for this project. Its lightweight design benefits the implementation in a Raspberry Pi and its speed makes it user-friendly. Additionally, utilizing SQL commands allows the user to access information more effectively.

The database used in the system is made up of a table with 5 columns:

- Unique ID - The table's primary key for identifying each entry;
- ID - Identification linked with the data gathered from the QR code;
- Date - Date and time at which the QR code was scanned;
- Operation Type - Type of operation associated with the detection of QR code, can be "load" or "delivery";

- Classification - Categorization of the package integrity, can be ”not classified”, ”not damaged, or ”damaged”.

By using SQL queries, data can be ordered by date or ID, or even searched for a specific package. To determine what type of operation is being performed, simple logic was applied: When a package has an even number of entries or none at all, it is being loaded; when it has an uneven number of entries, it is being delivered.

3.3.7. Application Layer: Node-RED Dashboards

All the data collected by the system is presented to the user via Node-RED dashboards. As previously stated, one of the major benefits of Node-RED is how easy it is to display data structured in node-RED flows using built-in nodes.

Two dashboards were created for the user experience. The first dashboard displays information regarding the status of the truck’s cargo door to the user. It informs the user whether the door is opened or not and its evolution over time through a graph. Fig. 18 depicts the dashboard.

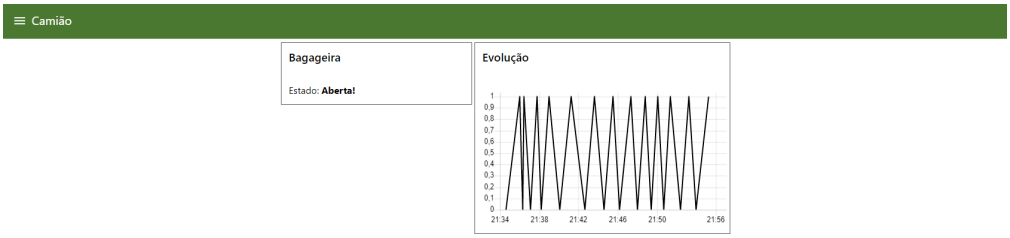

Figure 18. Node-RED dashboard with information regarding the truck’s cargo door

The second dashboard includes the information stored in the SQLite database. Using an HTML table, the columns ”ID”, ”Date and Time”, ”Operation Type”, and ”Classification” from the SQLite table are displayed in this dashboard, which was created using HTML. The ”Classification” field can be changed directly on the dashboard to allow users to quickly categorize the physical state of the package after examining the videos. Furthermore, there is a hyperlink that, when clicked, downloads the video file related to the operation performed, and a slider to choose the number of months elapsed to delete records from the database. The dashboard is visible in Fig. 19.

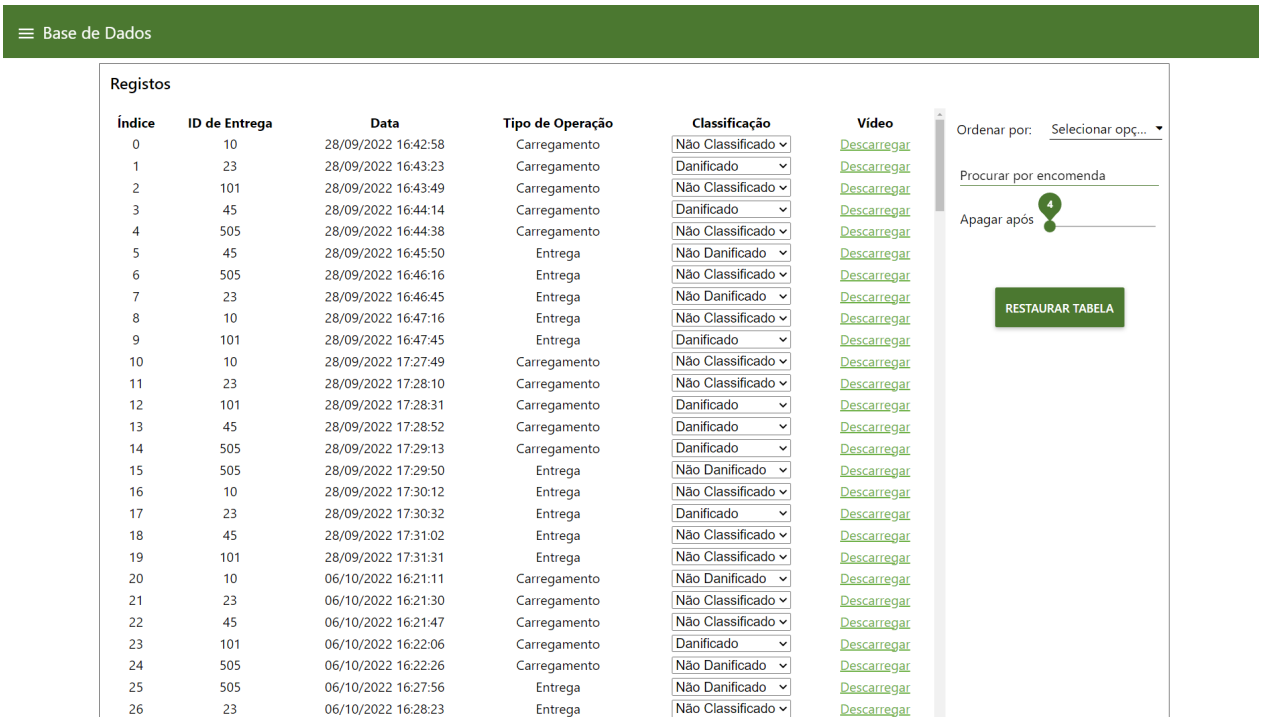

Figure 19. Node-RED dashboard with database information

CHAPTER 4

Implementation and Results

This chapter covers the implementation and the results obtained by the testing procedures performed on the system developed. The goal is to simulate a real-world scenario of package delivery, so that the results obtained from implementing the system and testing are as comparable as possible to an implementation inside of a transport truck.

A description of the real-world scenario simulated and the system's implementation inside the transport truck is provided in section 4.1. Section 4.2 includes all tests performed and their reasoning (4.2.1, 4.2.3, 4.2.5 and 4.2.7), a discussion and analysis of the results (4.2.2, 4.2.4, 4.2.6 and 4.2.8), and a report on the file and database storage (4.2.9). Finally, section 4.3 hypothesizes about a possible real-case implementation.

4.1. Implementation

To replicate the real-world scenario, the system was implemented inside a classroom that represents the cargo space inside a transport truck. Packages with QR codes placed on top were used as the deliveries that had to be made, with each QR code being unique and can be seen in Fig. 20. The packages were brought in and out of the classroom to simulate the loading and unloading process.

Figure 20. Packages used for testing

In terms of the system itself, the motion sensing ESP32 was positioned at the end of the room, distanced from the Raspberry Pi to ensure that the Python scripts were up and running by the time the QR codes crossed the camera's field of view. Initially, the Raspberry was mounted halfway and high above inside the room to get a top-down view of the cargo, but the results in section 4.2.4 reveal that packages are not entirely visible with this arrangement, thus the Raspberry Pi was instead positioned on a table at waist level, so the driver can display it to the camera during recording. The implementation can be seen in Fig. 21, where "A" is the position of the ESP32, "B" is the initial position of the Raspberry Pi, and "C" is the final position.

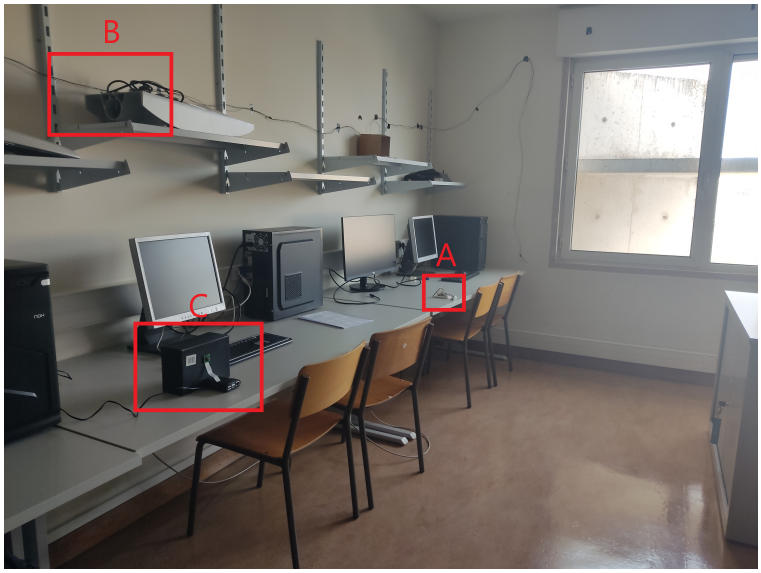

Figure 21. Implementation in the classroom

4.2. Tests and Results

Throughout this section the tests conducted to evaluate the system will be detailed, along with their purpose and results. The first one is to measure the detection range of the QR code Python script. To determine if a package was damaged during the loading/unloading procedure, the camera must have visibility over the procedure.

If the QR code can only be read at close distances, the video recorded won't be enough to analyze potential damages, thus rendering the system useless. For testing

purposes, different sizes of QR codes will be used to determine if there's any significant change to the detection range.

The second test was developed to confirm the consistency of the QR code detection and video recording scripts. This test resembled the loading process; numerous packages were stored in the classroom, and after a period of time, the database was consulted for any missed QR codes and/or missing videos. The loading and unloading process was longer due to the system's implementation, thus the test was timed to see how much longer it took.

Following that, the third test incorporates damaged packages to verify if the videos recorded during the loading/unloading procedure can be used identify damage caused to packages.

The last test determines whether daylight has an impact on the system's behavior, more precisely whether it influences the QR code detection and lessens the visibility of the video recording. During the unloading process, the sun may reflect on surfaces rendering the QR code illegible or making it impossible to verify damages in the video.

Finishing the section is a report on the size occupied by the video recordings and SQLite. Finally, the last section includes the hypothesized implementation for a real-life application of the system.

4.2.1. QR code detection range

This test was performed to measure the maximum detection range of the QR code detection script. Using printed QR codes with different sizes, this trial studied if the size difference affected QR detection.

The script developed can read QR codes with ease, however it was not tested in circumstances comparable to a real case scenario. In order to work in a real case scenario, the detection range has to be enough so that it can identify the QR code and record the entirety of the package in its field of view.

For this test, three QR codes were printed on A3, A4, and A5-sized sheets of paper. The paper's width was completely filled with the QR codes centered, as seen in Fig. 22. Following that, the farthest distance at which the QR code could be read was measured.

Figure 22. A3, A4, A5 QR codes

4.2.2. QR code detection range - Results

The results gathered from the QR code detection range test are presented in table 4. Results show that the bigger the QR code size, the greater the detection range is.

Table 4. Maximum detection range per QR code size

Size	Maximum Distance [cm]
A3	181
A4	128
A5	96

For the system, these findings imply that in order to achieve the full autonomy, as mentioned in section 3.1.1, the bigger the QR code size the better. However, the benefit of having a longer detection range involves a few drawbacks.

One, it becomes more difficult or perhaps impossible to apply a QR code on a package the bigger the QR code is. Since an A3-sized QR code is 29.7 cm in width and 42 cm in length, it won't fit on most packages. This can be observed in Fig. 20 since the smallest package used in testing could only fit an A5 QR code.

Second, bigger QR codes result in more expensive deployment. For instance, Santos e Vale distributes more than 8,500 packages per day that require a QR code for identification. This means that a slight reduction in deployment costs through the use of smaller QR codes would result in significant long-term cost savings.

It is important to know that the maximum detection range is influenced by the angle at which the camera's line of sight intersects the QR code. If the QR code is exactly

vertical and forms a 90º angle with the camera's line of sight, the distance measured corresponds to the maximum possible; however, if the QR code is slightly tilted, the distance is reduced. This situation is represented in Fig. 23. For the maximum distance measured in table 4, the QR codes formed the 90º angle, therefore in a day-to-day situation, the distance is often less.

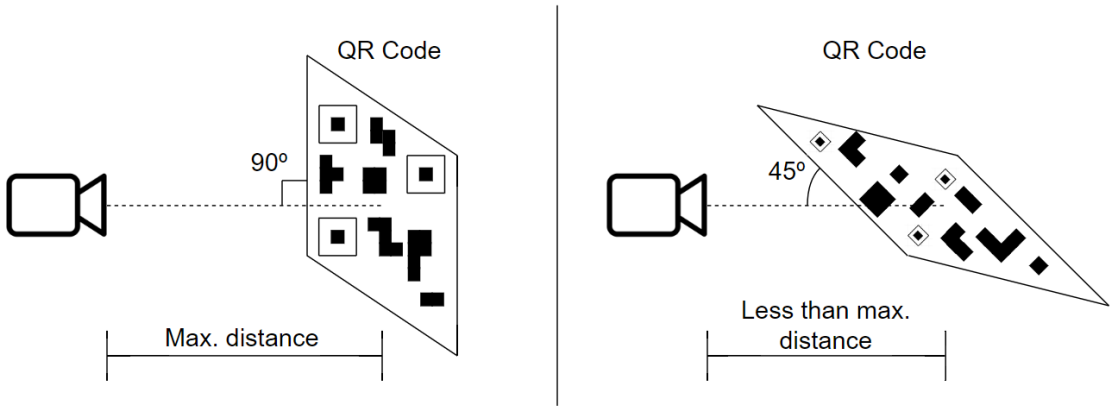

Figure 23. Influence of QR code angle in detection

Something else that may hinder the maximum distance is the camera module used. The camera has a still resolution of eight megapixels, higher quality cameras have better resolution and may provide better QR detection results, without mentioning the benefit of more detailed video recordings.

Due to the shortcomings mentioned and to have a standard QR code size that fits all of the packages, the A5 sized QR code was picked for the rest of the tests performed.

The system was also modified to include a buzzer connected to the ESP32. This buzzer sounds off when a QR code has successfully been scanned to indicate the driver that the package has been identified and may be stored or delivered. With the usage of the smaller, A5 sized QR code the detection range is only 96 centimeters, and the modification prevents situations where the QR code cannot be read due to the distance.

4.2.3. QR code detection fidelity

The goal of the second test is to assess the fidelity of the two Python scripts executed by the Raspberry Pi. To achieve this, the loading and unloading procedures were simulated using five packages with distinct QR codes. The simulation occurred has follows:

(1) ESP32 detects cargo door opening - QR code detection script is executed;
(2) Package is brought in and the respective QR code is identified - Code is validated, QR code detection script interrupted, buzzer sounds and video recording script is executed;
(3) Recording ends and package is stored - QR code detection script executed again;
(4) Steps two and three are iterated through for the remaining packages;
(5) ESP32 detects cargo door closing - QR code detection script is shutdown.

This procedure represents the loading process of delivery. The same is also done for the unloading process with the exception that step four is skipped because it is expected that each package has a different delivery location. This means that steps one, two, three and five are repeated five times in total for unloading all packages.

After simulating both loading and unloading procedures there should be a total of 10 QR codes identified, and 10 videos recorded. Afterward, the database containing the QR code entries and the Raspberry Pi folder that stores the videos recorded are going to be checked for any missing elements. This whole process is going to be repeated 10 times to ensure that the results are reliable.

As a consequence, the results gathered and alterations made from the test conducted in section 4.2.5 and 4.2.6, this test is going to include a second attempt. The differences from each attempt are the number of repetitions which were reduced to five, and the recording duration that was increased by five seconds, totaling at 10, from the first attempt to the second.

If the results reveal that any entries or videos are missing, it means that either the system or the scripts malfunctioned, and it must be addressed so that all packages are identified, and damages can be detected. Besides evaluating the efficiency of the scripts, this test also allows to extract other results.

One of them is the extra time needed for package identification and recording. To do this, the simulation from before will be timed, and by accessing the system's logs, the amount of time spent running the scripts can be determined. The measured time represents how long it takes for the system to validate the QR code, record the loading/unloading procedure and save the video. Determining the time spent by the system is very important because if it is too long there is a negative impact on the service.

The other result that can be obtained from this test is the visibility granted by the video recording. Since it is the first time that videos are going to be recorded, the collected videos can be used to evaluate the visibility provided by the implemented setup.

4.2.4. QR code detection fidelity - Results

The results obtained from running the test described in the previous section can be found in table 5. More detailed values can be found in the appendix B.

Table 5. QR code detection fidelity results

	1st Attempt	2nd Attempt
Codes Scanned	100	50
Codes Missed [%]	0	0
Videos Missed [%]	0	0
Average Loading Time [s]	87.8	99.8
Average Unloading Time [s]	119.1	119
Average Time Spent Per Package [s]	7.71	12.6

The fact that no QR codes and videos were missed means that the scripts are running perfectly. These results demonstrate that the ESP32 is placed distant enough to activate the QR code detection script on time to detect packages being loaded or delivered and that the Node-RED script setup is working as intended.

The unloading time is greater than the loading time due to the different way the simulations are treated. As previously stated, the simulation of the unloading phase has to activate and deactivate the ESP32 five times, opposed to just once during the loading phase. This is done to replicate the different locations where the packages are

delivered, and causes the 31.3 and 19.2 second difference between both times, in the first and second attempt respectively.

Regarding the first attempt, since the video recording itself takes five seconds, analyzing the ”Average Time Spent Per Package” value reveals that it takes 2.71 seconds to authenticate the QR code and save the video file. As a result, if another package is to be loaded or unloaded, it must be done roughly three seconds after the recording of the previous one. As for the second attempt, subtracting the 10 second period needed for recording yields a time of 2.6 seconds for video storage and QR authentication, suggesting that the time had no impact on time required to save the video and validate the QR code. Both these times are low enough to not warrant any changes to the system.

Finally, playing back the recorded videos highlights an issue with the way the Raspberry Pi and the camera module were setup. Although the top-down view of the cargo makes it easier to film the entire loading/unloading procedure, the package is not entirely visible. It is difficult to properly inspect the package for damage without the use of a second camera.

Changing the camera location proved to be a solution for the issue without altering the system’s core design. Positioning the camera at waist level allows workers to display the package during the five seconds of the video recording. With this adjustment, packages can be examined for damages, but the system is slightly less autonomous as a result. To further aid the drivers with displaying the package, the buzzer also sounds off when the video recording has ended.

4.2.5. Video recording quality

The packages utilized in the prior test will now be used to evaluate the video recording’s quality. The purpose of this test is to determine if the quality of the videos recorded by the camera module is sufficient for manual inspection of damages.

As previously stated, the used camera has an eight-megapixel resolution and can capture video up to 1080p, however, recording a higher quality video comes at the cost of a bigger file. In order to find the optimal ratio between file size and video resolution,

so that it is possible to identify damages, the smallest resolution will be tested first, and if necessary, it will be incrementally increased.

For the test itself, packages with small holes are going to be identified and recorded by the system. The videos stored are going to be watched to check whether the damages are visible. If the damages are not visible, it means that the quality is insufficient to achieve the objective of verifying package integrity.

4.2.6. Video recording quality - Results

Since the results are not quantifiable, a scale was defined to convey them. The scale has three classifications based on whether the damages can be seen through the videos that are: ”Visible”, ”Barely Visible”, and ”Not Visible”. The packages are labeled from one to five, following the order from smallest size to biggest. With a scale defined to evaluate the video quality and the packages labeled, the smallest resolution of 640 x 480 was put the test, and its results are presented in table 6.

Table 6. Video recording quality results for 640 x 480 resolution

Packages	Not Visible	Barely Visible	Visible
1	-	-	✓
2	-	-	✓
3	-	-	✓
4	-	✓	-
5	✓	-	-

Analyzing the table, packages four and five show unsatisfactory results. However, the video resolution is not at fault for these results, as can be seen in Fig. 24, but the recording duration is. As the bigger packages are put through the system, the five second period to display the package for the video recording is insufficient to show all sides. For the case of package four, the last side is not displayed with enough quality, and for package five the final side is not displayed at all, both problems due to time constraint.

Given that damages are identified in the smaller packages, increasing video resolution isn’t necessary. The employee could speed up the displaying process, although this could decrease video quality due to the package not being still enough time. The other alternative is to extend the recording duration of the second script to 10 seconds,

Figure 24. Damage inspected in the video recording of package 1

which allows drivers to calmly show all sides of the package without worrying with time constraints. This alternative was the chosen one.

As a result of this change, all the damages on packages are now visible, making the verification of package integrity during delivery possible. This change does come with the trade-off of a bigger file size and longer ”Average Time Spent Per Package”. Due to this, the experiment ran in sections 4.2.3and 4.2.4 was repeated to analyze the changes caused by the increase in recording time.

4.2.7. Lighting impact

This last test consists on determining if lighting has an impact on the system’s normal behavior. In real case scenario, light could reflect or shine in a way that affects the system in a negative manner. Since the implementation was done in a controlled environment, this test was devised to study the impact of lighting conditions that could occur in a non-controllable environment.

The system’s setup was moved around in favor of the natural sunlight that entered the classroom. Changing the system’s location, more specifically the Raspberry with the camera module, allowed for testing the light’s influence on the QR detection script. Five angles, labeled from ”A” to ”E”, were tested for the maximum distance at which a QR code could be identified. Angle ”A” corresponds to when light shines on the QR code, and angle ”E” is the opposite angle, corresponding to when light shines on the camera module. The rest of the angles are 45° shifts, starting from angle ”A” and ending with ”E”. These angles can be visualized in Fig. 25.

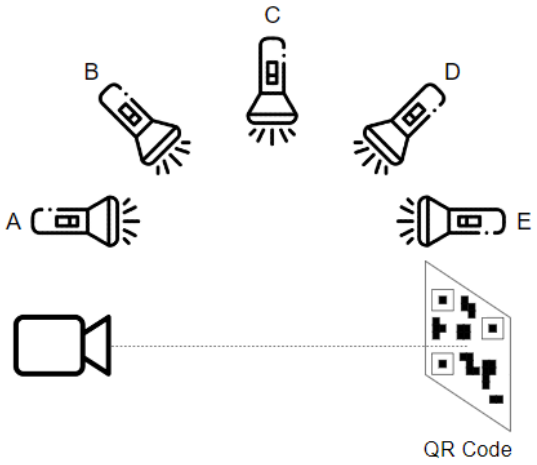

Figure 25. Angles tested for lighting impact

Understanding how light could affect the system is crucial since it facilitates deployment in real life. With the help of the test results, it is possible to determine the necessary precautions to avoid lighting issues when placing the system in the cargo space.

4.2.8. Lighting impact - Results

In order to analyze the effect of light on the system, the variable tested was the detection range of the QR detection script. This script is fundamental for the system, and if it malfunctions due to light, the system cannot function. For each angle where light shined from, the maximum detection range was measured and noted on table 7, the test was done in two separate days at the same time, where one day was sunny and the other cloudy.

Table 7. Light effect on the maximum detection range

Angle	Detection Distance (Cloudy) [cm]	Detection Distance (Sunny) [cm]
A	80	75
B	96	96
C	88	85
D	45	30
E	17	8

The results obtained indicate that the detection range is significantly decreased when light is in the camera's field of vision. For angles "D" and "E" the detection range

decreased roughly between 53%-69% and 82%-92%, respectively. The range limitation highlights how crucial the system's placement inside the cargo space is. The camera must be placed such that no ambient nor artificial light shines directly, or even at an angle, into its field of view.

As with the remaining angles, the light's influence is not as impactful. Angles "A" and "C" had decreases between 17%-22% and 8%-12%, respectively, whilst angle "B" registered no decrease. The decrease measured for angle "A" is a consequence of the light reflecting on the QR code. The reflection caused the QR code to brighten, reducing the detection distance.

The differences between angles on different days suggest that the detection range is dependent on the weather. The test was conducted on two separate afternoons, one with cloudy weather and another with sunny weather. Consequently, the distance measured varied slightly on the same angles. The same may be said for the time of day, because the angle at which light shines on the camera changes during the day due to the sun's position in the sky. Based on the results gathered, it can be generalized that the ideal implementation to reduce the effect of light on the system is to mount the camera perpendicularly to the cargo door. This causes the camera's line of sight to be perpendicular to the cargo door, where light comes from, preventing situations when light shines straight on the camera.

4.2.9. Report on file and database storage

For each QR code identified during the tests conducted, an entry was created in the SQLite table and a video was recorded. According to Santos e Vale's website statistics [47], the company completes 8,500 deliveries per day with a fleet of 500 vehicles, averaging 17 packages delivered per vehicle.

The Raspberry Pi's storage space is limited, so it is important to estimate how long it will take to reach maximum storage space in order to establish a file migration schedule. To accomplish this, the average size of each video and SQLite entry was calculated, and afterward, the number of days needed to use up all of the Raspberry Pi's storage was

determined. The schedule can then be defined so that it prevents the Raspberry from running out of space.

The calculations performed and the respective values can be consulted on table 8. It provides the total number of entries and videos recorded, as well as their size on the Raspberry, and the average size per entry and video.

Table 8. Storage report

Entries recorded	186
Database file size [KBytes]	24,576
Average entry size [Bytes]	132.13
Five second videos recorded	132
Five second videos total file size [KBytes]	49,540
Average five second video size [KBytes]	375.33
Ten second videos recorded	54
Ten second videos total file size [KBytes]	40,128
Average ten second video size [KBytes]	743.11

Each vehicle transports 17 deliveries per day, translating into 34 QR codes identified by the system, one for loading and one for unloading, which means that every day 34 entries and videos are stored in the Raspberry Pi. Pairing this information with the average file size for a ten second video and an entry on the SQLite database, the storage size occupied per day can be calculated using the equation 4.1.

$$SizeOccupied_{day} = (Video_{avg} + Entry_{avg}) * Packages_{day} * 2 \tag{4.1}$$

Applying the equation for Santos e Vale's statistics results in a storage size occupation of 24.68 Megabytes each day. Determining this value is fundamental to determine a migration schedule. The migration should be performed before the Raspberry Pi runs out of storage space, so calculating how long this takes is important. This time is dependent on two variables, the memory card storage space and the storage size occupied per day and can be calculated using the equation 4.2.

$$DaysToFillStorage = \frac{MemoryCardStorage}{SizeOccupied_{day}} \tag{4.2}$$

For this implementation a 16 Gigabyte memory card was employed, however some of the memory card's capacity is required for the Raspberry operating system and the tools and libraries necessary for the system to function. Acknowledging this important factor, it is assumed that the usable storage space on the memory card is of nine gigabytes. Taking into account this value and the storage size occupied per day estimated previously, the number of days to fill out the storage is 373. So, the migration period must be less than 373 days to avoid losing any information collected by the system.

4.3. Possible real-case implementation

This section reflects on a possible real-life implementation for the system developed throughout this dissertation. Fig. 26 displays the implementation hypothesized, where "A" is the position of the ESP32 and "B" is the position of the Raspberry Pi, note that the camera module is to be faced perpendicularly to the entrance. There can be many variations for the actual implementation, but they all need to account for the aspects highlighted by the results.

Figure 26. Hypothesis for implementation [56]

First, the distance between ESP32 and the Raspberry Pi. The ESP32 notifies the Raspberry Pi when motion is detected by the HC-SR501 movement sensor. This notification activates the QR detection script and given that communication and executing the script takes time, both components must be distanced enough so that the camera is ready to identify QR codes by the time the worker reaches the Raspberry Pi. Fig. 26 shows that

the ESP32 was positioned at the entrance of the truck, and the Raspberry Pi at the back to ensure that the distance is enough.

Second, the Raspberry Pi position and camera orientation. Lightning has an impact on how well the system performs, as demonstrated in 4.2.8; as a consequence, implementation should be done with concerns to the possible effects of lightning. As seen in Fig. 26, the Raspberry Pi is placed further into the cargo space, and the camera module is positioned perpendicular to the cargo door in order to minimize the exposure to natural lightning.

Not respecting these conditions could result in the system not working as intended and not performing up to the standards, such as not being able to identify certain packages, and consequently not recording the loading or unloading process.

CHAPTER 5

Conclusions and Future Work

The major goal for this dissertation was to help delivery businesses with fraud returns. To achieve this, it was established that the packages transported by these businesses needed to have their integrity verified, when loaded into the transport truck, and before being delivered to the customer. To do this, IoT was chosen as the base for the development of this solution. This choice was made due to the emergence of IoT solutions in retail and logistics business together with the existing potential of IoT sensor devices.

Even though the solution was originally conceptualized to help businesses with fraudulent returns, it proved beneficial in other ways. One was to provide assistance to customers by verifying the state in which their deliveries arrived, ensuring the customer the right to a return/refund. Besides that, the information collected by the system could also be used as a monitoring tool for drivers, analyzing which ones had a higher rate of damages packages.

To guide the development of the solution three research questions were designed, which are divided into the two established objectives. The first two research questions, that were ”How can the system verify the quality of the package before delivery?” and ”How effective is the system at detecting package damage?”, originated the first objective. This objective consisted of implementing an IoT system that can verify the integrity of packages, helping businesses with fraud returns. The second objective was to use the collected data as a monitoring tool for driving, which conceived with the third research question, ”What further applications are there for the data acquired by the system?”.

Before developing the system, a literature review was conducted. The research showed that only a few projects attempt to verify package conditions during delivery.

The system's design and architecture were defined using the information acquired from the literature review. For this, the requirements for the system needed to be established beforehand. Santos e Vale, a logistics, distribution, and transportation business had issues with fraudulent returns, and in the mutual interest, provided knowledge about the company's day-to-day for the implementation of the system developed in this dissertation, and also proposed additional requirements. Once the requirements were set, the system's architecture was created with three core components: an ESP32, a Raspberry Pi and Node-RED.

Once the system was developed it was time to test it. To ensure that the results were reliable, testing was performed in a simulated real-world scenario. Four tests were conducted to guarantee a proper real-life implementation, and the results were analyzed to calibrate the system and correct errors. A study was also performed to determine the best time for migrating the data stored by the system.

With the system completed, conclusions can be drawn about the system's ability to accomplish the objectives established and answer the research questions. The first question was how the system could check the conditions of packages before delivering. It was hypothesized that a camera could be used to take images before delivering the goods to a consumer. It can be concluded that the developed system solved this question, since the state of the package is verified before loading and when being delivered. Even if the verification is not done with images as originally planned, it is done with short videos.

The second question was proposed to assess the system's efficiency at detecting damages. The findings obtained in 4.2.6 prove that the system's is highly efficient. The purpose of the test was to evaluate the quality of the videos recorded by the system in order to use the videos to examine package damage. By watching the videos, all damages present on the packages tested were identified, showing that the system is highly efficient, only suffering from poor package placement or unfavorable lighting conditions.

As previously stated, the first objective arose from these first questions; therefore, given that the two questions were answered, it can be concluded that the first objective

was accomplished. Furthermore, each objective had a set of requirements associated to ensure that the objective could be achieved. All of these requirements were met because the system identified all packages, the information was easily accessible, and the videos enabled the detection of damages during delivery, indicating that the system was successful at achieving the objective.

For the final research question, it was expected to find other applications for the data gathered by the system. This defined the objective of using the data as a monitoring tool for driving quality during transportation. The system's recordings of the loading and delivery process makes it possible to identify damages, but as every truck may be driven by a different driver each day, the classification of a damaged item must be cross-referenced with the driver in charge of the vehicle that day. This wasn't accomplished, and as a result the objective wasn't fulfilled. Regardless, the question is answered, and monitoring driving quality is possible in the future.

As with the previous objective, there was also one requirement not entirely fulfilled by the system. Santos e Vale requested two extra requirements: the system had to operate autonomously without interfering with workers' tasks, and packages had to be detected using QR codes. Due to system's limitations, it was not possible for the system to operate completely autonomously; workers had to display the package for QR code identification and video recording. Additionally, the videos have to be manually reviewed to identify damages.

Although some aspects weren't accomplished, the project developed in this dissertation successfully accomplished initial objectives. The main goal for the system was to assist logistics services with fraudulent returns, and it was successful in doing so. The system's recordings and dashboards allowed users to view videos of the loading and delivering process, and correctly categorize packages as damaged or not damaged. The dashboards also allowed users to access database-stored information stored in a simple and easy process. The system can be applied in a real case scenario, such as small businesses or larger transport companies, given that testing was performed as a simulation of a real-world scenario.

The research conducted during the dissertation can be used as a blueprint for future works, even if the system isn't put to use in the actual world. It still acts as a proof of concept for any future projects that might be developed that are similar to this one.

5.1. Limitations

As stated in the conclusion, there are a few limitations that hinder the system's full potential. The most relevant is the detection range of the QR codes. When tested on A5 sized QR codes, the detection range was of 96 centimeters. In a real-life application, companies will look to use much smaller QR codes, certifying that the code fits on the package and cutting back on deployment costs. As a result of using smaller codes, the detection range will also reduce making the system harder to use.

Another drawback is the system's autonomy. The system was designed to be totally autonomous, as requested by Santos e Vale, and as demonstrated during testing, this is not achievable without the use of two cameras or the worker displaying the package. However, having two cameras would result in twice as many videos to examine, which implies more work for the reviewer.

The final limitation is that the system's hardware requires power and an internet connection to work. Since the system is to be implemented in transport trucks, and these are constantly moving, the power and internet connection must be provided by the truck. Due to a limitation of the ESP32 and the Raspberry Pi, it is not suggested to alter or turn off the network to which the hardware is connected. If the network changes, both components must be manually reconnected.

5.2. Future Work

This final section provides improvements for future implementations of the system developed in this dissertation. The recommendations given seek to improve the system created or even incorporate new functionalities. The limitations mentioned in the previous section will serve as foundation for future work suggestions.

The first improvement would be to improve the camera setup used. Using two higher quality cameras with the proper implementation would improve the system on two of its limitations. First, the usage of two cameras would remove the need for the worker to

display packages, since the packages can now be recorded in its entirety, improving the system's autonomy. Second, the higher quality of the camera could show improvements in the QR code detection range.

Upgrading the cameras presents an opportunity to employ machine learning. With higher resolution videos, machine learning could be used to automatically identify damages that have occurred in packages. This innovation would turn the system fully autonomous since it is no longer necessary human intervention to examine each video. Machine learning could also be applied to calculate the volume of each package. This would provide useful to determine how full each truck is and manage cargo between trucks.

5.3. Final Words

As an ending note, this dissertation was conducted during the period between September 2021 and November 2022 at ISCTE-IUL. Overall, working on the project was enjoyable and gratifying because it contributes to society moving forward to a more efficient and fairer future. Observing the success of the project in the end was very rewarding since it seemed like it was a difficult task in the beginning.

There were a few setbacks at times with setting up the equipment or hardware malfunctions, but these setbacks also were learning opportunities. Developing the system allowed me to gain knowledge in the logistics services area as I read related work , and learn even more programming languages, protocols, hardware, software, databases which were all used for the creation of the system.

Finally, the work done throughout the period mentioned previously resulted in two scientific papers being written. The first paper was written for the 28th IEEE ICE/ITMC & 31st IAMOT Conference IEEE which occurred between the 19th of June and 23rd of 2022, and the second paper was written for the " International Symposium on Sensing and Instrumentation in the 5G and IoT Era" conference that took place in November 17 and 18 of 2022. In this last conference, the paper written won the "Student Best Paper Award" in the topic of "Smart Sensors and Wireless Sensor Networks for IoT-1".

References

[1] S. Setti and A. Wanto, "Analysis of backpropagation algorithm in predicting the most number of internet users in the world," *Jurnal Online Informatika*, vol. 3, no. 2, pp. 110–115, 2019.

[2] B. Gemin, L. Pape, L. Vashaw, and J. Watson, "Keeping pace with k-12 digital learning: An annual review of policy and practice," *Evergreen Education Group*, 2015.

[3] L. Andrighetto, P. Riva, and A. Gabbiadini, "Lonely hearts and angry minds: Online dating rejection increases male (but not female) hostility," *Aggressive behavior*, vol. 45, no. 5, pp. 571–581, 2019.

[4] P. Timmers, "Business models for electronic markets," *Electronic markets*, vol. 8, no. 2, pp. 3–8, 1998.

[5] U. C. Bureau, *Quarterly retail e-commerce sales 3rd quarter 2021*, 2021.

[6] C. Chen, D. Zhang, X. Ma, *et al.*, "Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, 2016.

[7] S. Rajendran, "Improving the performance of global courier & delivery services industry by analyzing the voice of customers and employees using text analytics," *International Journal of Logistics Research and Applications*, vol. 24, no. 5, pp. 473–493, 2021.

[8] T. Callen, "Gross domestic product: An economy's all," *International Monetary Fund: Washington, DC, USA*, 2012.

[9] X. Liu, M. He, F. Gao, and P. Xie, "An empirical study of online shopping customer satisfaction in china: A holistic perspective," *International Journal of Retail & Distribution Management*, 2008.

[10] N. Vasić, M. Kilibarda, and T. Kaurin, "The influence of online shopping determinants on customer satisfaction in the serbian market," *Journal of theoretical and applied electronic commerce research*, vol. 14, no. 2, pp. 70–89, 2019.

[11] J. E. Collier and C. C. Bienstock, "Measuring service quality in e-retailing," *Journal of service research*, vol. 8, no. 3, pp. 260–275, 2006.

[12] A. Chopra, M. Arora, and S. Pandey, "Delivery issues identification from customer feedback data," *arXiv preprint arXiv:2112.13372*, 2021.

[13] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2018, pp. 29–35.

[14] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelfflé, *et al.*, "Vision and challenges for realising the internet of things," *Cluster of European research projects on the internet of things, European Commision*, vol. 3, no. 3, pp. 34–36, 2010.

[15] A. Khanna and R. Anand, "Iot based smart parking system," in *2016 International Conference on Internet of Things and Applications (IOTA)*, IEEE, 2016, pp. 266–270.

[16] A. Alphonsa and G. Ravi, "Earthquake early warning system by iot using wireless sensor networks," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, 2016, pp. 1201–1205.

[17] N. Gondchawar, R. Kawitkar, *et al.*, "Iot based smart agriculture," *International Journal of advanced research in Computer and Communication Engineering*, vol. 5, no. 6, pp. 838–842, 2016.

[18] Y.-D. Lee, "Implementation of greenhouse environment monitoring system based on wireless sensor networks," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 17, no. 11, pp. 2686–2692, 2013.

[19] Y. Jie, J. Y. Pei, L. Jun, G. Yun, and X. Wei, "Smart home system based on iot technologies," in *2013 International conference on computational and information sciences*, IEEE, 2013, pp. 1789–1791.

[20] Y. K. Dubey and S. Damke, "Baby monitoring system using image processing and iot," *International Journal of Engineering and Advanced Technology*, vol. 8, pp. 4961–64, 2019.

[21] M. W. Woo, J. Lee, and K. Park, "A reliable iot system for personal healthcare devices," *Future Generation Computer Systems*, vol. 78, pp. 626–640, 2018.

[22] P. M. Kumar, S. Lokesh, R. Varatharajan, G. C. Babu, and P. Parthasarathy, "Cloud and iot based disease prediction and diagnosis system for healthcare using fuzzy neural classifier," *Future Generation Computer Systems*, vol. 86, pp. 527–534, 2018.

[23] H.-E. Lin, R. Zito, M. Taylor, *et al.*, "A review of travel-time prediction in transport and logistics," in *Proceedings of the Eastern Asia Society for transportation studies*, Bangkok, Thailand, vol. 5, 2005, pp. 1433–1448.

[24] S. Agrawal, R. K. Singh, and Q. Murtaza, "A literature review and perspectives in reverse logistics," *Resources, Conservation and Recycling*, vol. 97, pp. 76–92, 2015.

[25] D. J. Thomas and P. M. Griffin, "Coordinated supply chain management," *European journal of operational research*, vol. 94, no. 1, pp. 1–15, 1996.

[26] H. Stadtler, "Supply chain management—an overview," *Supply chain management and advanced planning*, pp. 9–36, 2008.

[27] M. P. De Brito and R. Dekker, "A framework for reverse logistics," in *Reverse logistics*, Springer, 2004, pp. 3–27.

[28] D.-H. Shih, F.-C. Huang, C.-Y. Chieh, M.-H. Shih, and T.-W. Wu, "Preventing return fraud in reverse logistics—a case study of espres solution by ethereum," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 16, no. 6, pp. 2170–2191, 2021.

[29] E. Lopez-Santana, W. Rodrı¤guez-Vásquez, and G. Méndez-Giraldo, "A hybrid expert system, clustering and ant colony optimization approach for scheduling and routing problem in courier services," *International Journal of Industrial Engineering Computations*, vol. 9, no. 3, pp. 369–396, 2018.

[30] S. Proto, E. Di Corso, D. Apiletti, *et al.*, "Redtag: A predictive maintenance framework for parcel delivery services," *IEEE Access*, vol. 8, pp. 14 953–14 964, 2020.

[31] M. El Midaoui, M. Q. El Mehdi Ben Laoula, and K. Mansouri, “Logistics tracking system based on decentralized iot and blockchain platform,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 1, pp. 421–430, 2021.

[32] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[33] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, *et al.*, “Preferred reporting items for systematic reviews and meta-analyses: The prisma statement,” *Int J Surg*, vol. 8, no. 5, pp. 336–341, 2010.

[34] C.-H. Chuang, D.-H. Lee, W.-J. Chang, W.-C. Weng, M. O. Shaikh, and C.-L. Huang, “Real-time monitoring via patch-type piezoelectric force sensors for internet of things based logistics,” *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2498–2506, 2017.

[35] R. R. Oliveira, I. M. Cardoso, J. L. Barbosa, C. A. da Costa, and M. P. Prado, “An intelligent model for logistics management based on geofencing algorithms and rfid technology,” *Expert Systems with Applications*, vol. 42, no. 15-16, pp. 6082–6097, 2015.

[36] K. Prasanna and M. Hemalatha, “Rfid gps and gsm based logistics vehicle load balancing and tracking mechanism,” *Procedia Engineering*, vol. 30, pp. 726–729, 2012.

[37] G.-H. Yang, K. Xu, and V. O. Li, “Hybrid cargo-level tracking system for logistics,” in *2010 IEEE 71st Vehicular Technology Conference*, IEEE, 2010, pp. 1–5.

[38] E. Papatheocharous and P. Gouvas, “Etracer: An innovative near-real time track-and-trace platform,” in *2011 15th Panhellenic Conference on Informatics*, IEEE, 2011, pp. 282–286.

[39] J. Xing, “An intelligent logistics tracking system based on wireless sensor network,” *International Journal of Online Engineering*, vol. 14, no. 1, pp. 17–28, 2018.

[40] W. Wang, “Remote monitoring system of logistics carrier based on wireless sensor network.,” *International Journal of Online Engineering*, vol. 14, no. 1, 2018.

[41] A. R. Elias, N. Golubovic, C. Krintz, and R. Wolski, “Where’s the bear?-automating wildlife image processing using iot and edge cloud systems,” in *2017 IEEE/ACM*

Second International Conference on Internet-of-Things Design and Implementation (IoTDI), IEEE, 2017, pp. 247–258.

[42] A. Kapoor, S. I. Bhat, S. Shidnal, and A. Mehra, "Implementation of iot (internet of things) and image processing in smart agriculture," in *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, IEEE, 2016, pp. 21–26.

[43] K. Lakshmi and S. Gayathri, "Implementation of iot with image processing in plant growth monitoring system," *Journal of Scientific and Innovative Research*, vol. 6, no. 2, pp. 80–83, 2017.

[44] A. Frank, Y. S. K. Al Aamri, and A. Zayegh, "Iot based smart traffic density control using image processing," in *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*, IEEE, 2019, pp. 1–4.

[45] A. Sharma, P. K. Singh, and Y. Kumar, "An integrated fire detection system using iot and image processing technique for smart cities," *Sustainable Cities and Society*, vol. 61, p. 102 332, 2020.

[46] S. Rane, A. Dubey, and T. Parida, "Design of iot based intelligent parking system using image processing algorithms," in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, 2017, pp. 1049–1053.

[47] *Santos e Vale - Historia*. [Online]. Available: https://www.santosevale.pt/pt_pt/history.

[48] *Raspberry Pi 4 Computer Model B*. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf.

[49] *How HC-SR501 PIR Sensor Works & Interface It With Arduino*. [Online]. Available: https://lastminuteengineers.com/pir-sensor-arduino-tutorial/.

[50] B. M. G. Mataloto, "Iot*(ambisense): Smart environment monitoring using lora," Ph.D. dissertation, 2019.

[51] *Oasis MQTT - Overview*. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt#overview.

[52] *MQTT*. [Online]. Available: https://mqtt.org/.

[53] OpenJS Foundation & Contributors, *Node-RED*. [Online]. Available: https://nodered.org.

[54] R. D. Hipp, *SQLite As An Application File Format*. [Online]. Available: https://www.sqlite.org/aff_short.html.

[55] ——, *Appropriate Uses For SQLite*. [Online]. Available: https://www.sqlite.org/whentouse.html.

[56] Salavan, *Isolation truck on white background*. [Online]. Available: https://www.shutterstock.com/pt/image-illustration/isolation-truck-on-white-background-274410668.

APPENDIX A

Python scripts

The full python code for the QR code script is shown in Fig. 27.

```python
# Imports
import cv2
import signal
import sys

cap = cv2.VideoCapture(0) # Setup video capture

detector = cv2.QRCodeDetector() # QR Code detection

def signal_handler(signum,frame):
    cap.release()
    cv2.destroyAllWindows()
    sys.exit(0)

signal.signal(signal.SIGTERM, signal_handler)

while True: # Infinite loop so that the QR Code detection is constantly scanning for data

    _, img = cap.read() # Obtain frame for QR Code detection

    data, bbox, _ = detector.detectAndDecode(img) # Method that reads QR Codes by detecting bounding box coords and decoding the hidden QR data

    if(bbox is not None): # Draw blue box around QR Code and write data on top
        for i in range(len(bbox)):
            cv2.line(img, tuple(bbox[i][0]), tuple(bbox[(i+1) % len(bbox)][0]), color=(255,
                     0, 0), thickness=2)
        cv2.putText(img, data, (int(bbox[0][0][0]), int(bbox[0][0][1]) - 10), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (255, 250, 120), 2)

        if data: # Data found is printed
            print("data found", data)

    cv2.imshow("code detector", img) # Display of the live camera feed to the Desktop

    if(cv2.waitKey(1) == ord("q")): # 'Q' can be pressed to stop the program
        break
```

Ativar o Windows
Aceda a Definições par

Figure 27. QR code detection script

Fig. 28 displays the entire python code needed to record the 10 videos after a QR code is detected.

```python
import numpy as np
import cv2
import time
import signal
import sys

cap = cv2.VideoCapture(0)

fourcc = cv2.VideoWriter_fourcc('a','v','c','1') # Codec definition

fps = cap.get(5) # Definition of FPS

numOfSeconds = 10

numberOfSavedFrames = fps * numOfSeconds # SavedFrames = FPS * Nº of Seconds

out = cv2.VideoWriter('Desktop/videos/Carregamento.mp4', fourcc, fps, (640, 480))

i = 0

while(i < numberOfSavedFrames):
    ret, frame = cap.read()
    out.write(frame)
    i += 1

out.release()
cap.release()
cv2.destroyAllWindows()
```

Figure 28. Video recording script

APPENDIX B

QR code detection fidelity full results

This appendix shows all the results gathered to determine the values presented on table 4.2.4. The results are split between two tables: table 9 displays the data collected when the package recording period was five seconds. Table 10 displays the data collected when the package recording period was ten seconds.

Initially, the system would only record five second videos, but as the results obtained in 4.2.6 showed, it was necessary to increment the recording period to ten seconds. As a consequence, the QR code detection fidelity test had to be repeated. For better understanding of the tables, each value is explained:

- Attempt - Number of the attempt;
- Codes Scanned - Number of QR codes scanned on the attempt;
- Codes Missed - Number of QR codes missed by the system;
- Videos Missed - Number of videos not recorded by the system;
- Loading Time [s] - Time spent loading packages;
- Unloading Time [s] - Time spent unloading packages;
- Execution Loading Time (ESP32) [s] - Time spent loading packages measured by the ESP32;
- Execution Loading Time (Node-RED) [s] - Time spent loading packages measured by Node-RED;
- Execution Unloading Time (ESP32) [s] - Time spent unloading packages measured by the ESP32;
- Execution Unloading Time (Node-RED) [s] - Time spent unloading packages measured by Node-RED;
- Average Time Spent by Scripts (Loading) [s] - Time spent on average by scripts to detect a QR code and record one video during loading;

- Average Time Spent by Scripts (Unloading) [s] - Time spent on average by scripts to detect a QR code and record one video during unloading;
- Average Time Spent by Scripts (All) [s] - Average of both values measured on ”Average Time Spent by Scripts”.

The difference between ”Loading/Unloading Time” and ”Execution Loading/Unloading Time” is because the ”Loading/Unloading Time” was a measurement of the entire loading/unloading process, while the ”Execution Loading/Unloading Time” was a measurement from the first QR code scanned to the last video recorded saved on the Raspberry. Finally, the value ”Average Time Spent Per Package” from table 5 is obtained by calculating the average of the ”Average Time Spent By Scripts” row for each table.

Table 9. QR code detection fidelity results for five second videos

Attempt	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Codes Scanned	10	10	10	10	10	10	10	10	10	10
Codes Missed	0	0	0	0	0	0	0	0	0	0
Videos Missed	0	0	0	0	0	0	0	0	0	0
Loading Time [s]	103	90	88	89	87	85	76	92	90	78
Unloading Time [s]	132	129	117	120	112	122	120	110	120	109
Execution Loading Time (ESP32) [s]	84	73	73	74	74	70	76	78	74	66
Execution Loading Time (Node-RED) [s]	84	73	73	74	74	70	76	78	74	66
Execution Unloading Time (ESP32) [s]	116	115	103	104	96	106	103	96	108	95
Execution Unloading Time (Node-RED) [s]	116	115	103	104	96	106	103	96	108	95
Average Time Spent by Scripts (Loading) [s]	9.4	8	7.6	8.4	8	7.8	7.4	7	7.6	7.6
Average Time Spent by Scripts (Unloading) [s]	8	8	7.4	7	7	7.8	7.6	7.4	7.6	7.6
Average Time Spent by Scripts (All) [s]	8.7	8	7.5	7.7	7.5	7.8	7.5	7.2	7.6	7.6

Table 10. QR code detection fidelity results for ten second videos

Attempt	#1	#2	#3	#4	#5
Codes Scanned	10	10	10	10	10
Codes Missed	0	0	0	0	0
Videos Missed	0	0	0	0	0
Loading Time [s]	109	102	100	98	90
Unloading Time [s]	126	122	117	115	115
Execution Loading Time (Arduino) [s]	99	96	78	90	85
Execution Loading Time (Node-RED) [s]	99	96	78	90	85
Execution Unloading Time (Arduino) [s]	117	115	112	112	108
Execution Unloading Time (Node-RED) [s]	117	115	112	112	108
Average Time Spent by Scripts (Loading) [s]	12.4	12.8	12.6	12.6	12.8
Average Time Spent by Scripts (Unloading) [s]	12	12.4	13	12.4	13
Average Time Spent by Scripts (All) [s]	12.2	12.6	12.8	12.5	12.9