



Implementation of an IoT Architecture for Automated Garbage Segregation.

Angel Antonio Rodríguez Varela

Universidad de los Andes
Department of Systems and Computing Engineering
Systems and Computing Engineering
Bogotá, Colombia
2022

Implementation of an IoT Architecture for Automated Garbage Segregation.

Angel Antonio Rodríguez Varela

Project presented in partial fulfillment for the requirements to opt for the title of:
Systems and Computing Engineering

Director:

Ph.D. Carlos Andrés Lozano Garzón

Director:

Ph.D. German Adolfo Montoya

Investigation Group:

COMIT

Universidad de los Andes

Department of Systems and Computing Engineering

Systems and Computing Engineering

Bogotá, Colombia

2022

To my parents, that did everything within their power to provide me with all the tools needed for learning.

To my fiancée, for not letting me quit, ever.

To my thesis directors for giving me the greatest challenge of my career.

Abstract

Recycling can be up to six times cheaper for a nation than not doing proper waste management at all. That is why the government of Colombia set out to take advantage of over 40% of the waste produced in the country by 2030. Therefore, it is very important to develop tools and mechanisms that enable proper recycling. The effort of this project resulted in an IoT architecture that allows for the identification of users and separation of bags while enabling the analysis of the whole process. This was achieved by using RFID tags, for user recognition, object detection models, for classifying the bags, bands with ultrasound sensors, a robotic arm and a cloud analytics system.

Keywords: Architecture, *IoT*, Detection, Cloud, Analytics, Robotic.

Resumen

Reciclar puede resultar hasta seis veces más caro para una nación, que hacer la gestión apropiada de residuos. Por lo tanto, el Gobierno de Colombia ha propuesto como meta aprovechar más del 40% de la basura producida en el país para el 2030. Es por esto que resulta crucial desarrollar herramientas y mecanismos que hagan posible un reciclaje adecuado. El esfuerzo de este proyecto resultó en la implementación de una arquitectura de IoT que permite la identificación de usuarios y la separación de basuras mientras, simultáneamente, habilita el análisis de todo el proceso. Esto se logró usando tags RFID, para el reconocimiento de usuarios, modelos de detección de objetos, para la clasificación de basuras, bandas con sensores de ultrasonido, un brazo robótico y un sistema de analítica en la nube.

Palabras Clave: Arquitectura, *IoT*, Detección, Nube, Analítica, Robótico.

Content

	Pp.
Abstract.....	VII
List of Figures.....	X
List of Tables	XI
Introduction	1
1. Problem Statement.....	3
1.1. Objectives	
1.1.1. General Objective	
1.1.2. Specific Objectives	
1.2. Theoretical Framework	
1.2.1. Internet of Things	
1.2.2. Object Detection	
2. Architecture Design	5
3. Results	8
4. Conclusions and Future Work.....	19
Bibliography	23

List of Figures

	Pág.
Figure 1. Architecture Design Representation.....	5
Figure 2. Pseudocode for Nano_Jetson_Final_Integration.....	9
Figure 3. Docker Container with PyTorch Model.....	10
Figure 4. Pseudocode for the RFID Server.....	11
Figure 5. Bands Pseudocode.....	12
Figure 6. Execution Result of RFID Communication.....	13
Figure 7. Execution result of the Nano Jetson server.....	13
Figure 8. Report for Analysis of the Waste Management Process.....	14
Figure 9. Three Moments of the Deposit Process.....	15
Figure 10. Results of the Object Detection Model.....	15
Figure 11. Nano Jetson server identifying the runing.....	16
Figure 12. Execution inside the Docker container.....	17
Figure 13. Status of the Bands for Execution of Figure 14.....	17
Figure 14. Model Performance Evidence in the Architecture.....	18

List of Tables

Pág.

Table 1. Configuration and Performance Results of the Object Detection Model...16

Introduction

Colombia is responsible for producing over one million tons of garbage every year. From those million tons, 85% of these are deposited, without any major separation, in huge landfills or open sky communal dumps. In retrospective, this implies that only 15% of the waste generated is being used for recycling [10]. This becomes a problem, not only because landfills are responsible for 8% of the greenhouse effect, but because discarding waste, regardless of the new use that it can be given, turns out to be over six times more expensive for a country (as stated previously on the abstract) than doing the proper management of garbage [12]. That is why the National Government of the country established, since January 1 of 2021, a unique code for classifying and separating waste in the whole territory. This was intended to be done from the source; every household in the country should be responsible for following the recycling laws, in order to start generating the consciousness needed to solve the problem of waste management [8]. However, having control over the source renders itself useless if the process is not coherent from start to finish and everything is just thrown into a communal dump.

In Bogotá alone, the Doña Juana landfill reaches over six thousand tons of garbage every year [10]. This is where the problem of controlling the entire process begins to take its shape. The model of waste management for an apartment complex goes as follows: every tower has its own garbage shut. When a family needs to deposit a garbage bag, they just go to the common trash shut and dump the bag right in there. There are no particular bins for each color that the government established (black, white or green). Some administrations define that the garbage that can be recycled (white) needs to be left to the side for “especial pickup” but, then again, there is no actual guarantee that those bags will get the proper treatment, or that even if they are separated, at first, that they will not end up in the same communal dump. Given that over 60% of households in the capital of Colombia live in apartments and that, as stated previously, there is no control over recycling after the garbage leaves a home for these forms of housing, there is no actual guarantee

that this will be useful throughout the whole lifecycle of waste management, if everything just goes into the same container. Even if there are penalties for households that do not comply with recycling laws (penalties that can go up to sixteen minimum wages), this type of reinforcement will not have an effect, either, because there is no actual way of telling who is doing the proper separation of garbage or when in the process is the recycling lost to remedy the situation [11].

For 2030, the Government of Colombia established as a goal to take advantage of more than 40% of the waste produced. This comes as no surprise, because, other than being cost efficient for the nation, it is also one of the main components of what is known as “circular economy” for the optimized exploitation of resources that can go back into the value chain and the productive lifecycles of incorporations.

Recycling also allows for, among other things, reducing the energy consumption. For example, for the production of aluminum, more than 90% of the energy needed can be saved when discarded aluminum is used again, as compared to working with the metal element from scratch. Just the recycling of a glass bottle alone represents the saving of energy needed to light up an old lightbulb for more than four hours, and to light up a new LED one even more. Besides, it is also important to note that, for example, paper and Wood production from exploitable waste implies less deforestation, contributing, this way to the health of the environment. Lastly, the task of recycling, and reach the target established by the government, can increase the number of jobs generated for the citizens, this is critical given that for 2021 the unemployment rate was over 10,5%, and one of the reasons for riots and protests in the country [13].

The project presented here posits a solution for classifying and separating bags effectively, that could potentially be implemented in apartment complexes or even industrial buildings. It leverages a MobileNet V1 SSD object detection model, manually trained (in a Nano Jetson computer for faster AI processing), and a UR3 robotic arm to deposit bags according to their color, along with band mechanisms and other sensors (such as ultrasound and RFID antennas) that complement the architecture and makes for a complete and robust system, able to identify users and mobilize bags. Finally, a visualization dashboard was built on Data Studio for analyzing the process in terms of trends for colors, days and users.

1. Problem Statement

1.1 Objectives

1.1.1 General Objective

Implement an IoT architecture that allows for the automation and analysis of the process for separating and depositing garbage bags.

1.1.2 Specific Objectives

- Define the problem requirements.
- Design the IoT architecture for garbage bags separation.
- Design an object detection model that classifies the bags according to their color for disposal.
- Implement the IoT architecture so that it integrates with the object detection model.
- Validate that the solution is functioning according to the problem requirements.

1.2 Theoretical Framework

1.2.1 Internet of Things for Waste Management

Internet of Things (or IoT, as it is vastly known) is the expansion of software beyond the boundaries of computers and networks. It entails the integration of sensors and other components such as industrial artifacts or common domestic home appliances to further improve what technology can do for human kind. IoT has applications in various fields such as the automotive industry, health and security [1].

Said applications, also, include waste management as a benefactor from IoT systems. As it can be seen in [2], object detection models can be used to even go as far as detecting individual waste (not just garbage bags) and classifying them as glass, metal, cardboard, paper plastic or general trash. The work also references the use of radiofrequency identification (RFID) in monitoring of solid waste and grey level co-occurrence matrixes for trash separation that can be seen in [3].

1.2.2 MobileNet v1 SSD for Object Detection

Conceived, first, for embedded applications and mobile environments, MobileNet serves as an improvement in performance and efficiency for computer vision. While the architecture itself is composed of 28 layers, the basis behind the model is a depth-wise convolution with a 1x1 pointwise convolution for reducing computation and model size [4]. After that, every layer finishes with a batch normalization and ReLU applied for optimization purposes. SSD Stands for “Single Shot Detector” and it is included as the last layer in MobileNet for Object Detection models. This algorithm places an already predefined number of boxes in the whole image with both a class and a score for each class that helps detect where the objects are placed [5].

2. Architecture Design

The following image depicts the overall design of the architecture defined for the solution needed. In this section a discussion about what each element represents, the responsibilities and dynamics that come to play can be found.

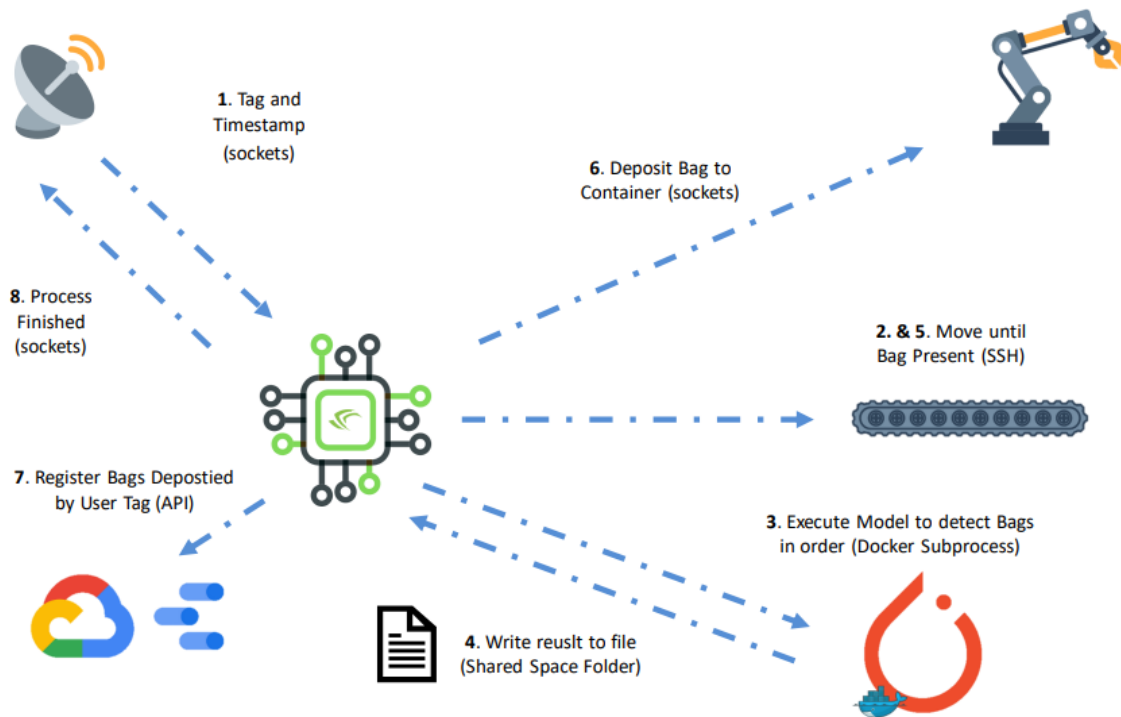


Figure 1. Architecture Design Representation.

In Figure 1, at the top left corner, an image of an antenna is shown, this is to represent the RFID module of the architecture. Next, at the bottom left corner icons of the Google Cloud Platform and Data Studio can be seen, this is to represent the analytics engine which was built with said technologies. Furthermore, at the center of the figure, there is a CPU icon that represents the Nano Jetson server used for processing and core orchestration of the entire IoT architecture. Next, at the bottom right corner there is a compound icon of Pytorch

and Docker, to represent the object detection module. At the center right side of the image there is a long band that represent where the bags go before being deposited. Finally, at the top right corner is the representation of the robotic arm, which, for this project, was a UR3.

Now, it is relevant to mention a deep dive on each component and the responsibility that it holds in the architecture:

- **RFID Antenna:** This sensor was used for the recognition of users in the system. Its job was to detect when a tag was placed between the two receivers. From there the information of the tag was sent, to a connected PC via USB, to serve as the first input of the model.
- **RFID Server:** The laptop connected to the RFID Antenna was in charge of receiving the tag information to be sent to the Nano Jetson, along with the datetime registered when the RFID tag was detected.
- **Nano Jetson:** This is where the orchestration took place. The Nano Jetson was in charged of calling for the bands to move, run a Docker subprocess with the object detection model was and get the information needed to send for the robot to deposit the bags detected where they belonged. Not to mention, also, that this server was also in charge of sending out all of the information to the Analytics Engine for further visualization of the process.
- **Object Detection Model:** The model, built with Pytorch, ran on a Docker container inside the Nano Jetson as a subprocess. Its job was, upon execution from the core program in the server, to take a picture of the current status of the bands, in order to detect the bags to deposit, and then to write a file with the exact order in which to deposit each bag, depending on the color.
- **Bands:** The bands where managed by a TXT Controller. From there a .py file was executed via SSH to move at a predefined speed. The bands also had ultrasound sensors that stopped the execution every time there was an object (bags) between them.
- **Robotic Arm:** The robotic arm received the right coordinates to deposit each bag detected depending on the color.

Finally, a more detailed step by step of the process is adequate to understand how the architecture functions as a whole:

1. An RFID tag is placed in the antenna for user recognition. The RFID Server gets the tag, concatenates a message with the current datetime and establishes connection with the Nano Jetson via Sockets to send said message.
2. The Nano Jetson receives the message and splits it to send the user and datetime information appropriately to the Analytics Engine. Then it executes the .py file in the Bands via SSH to move them a first time. This is to get some initial separation of the bags in order to take the picture and detect them.
3. Once the bags detect the first bag, and it is in place to be deposited, the bands stop and the process in the core program of the Nano Jetson continues. Now, the Object Detection model can be executed. Next, the program inside the Docker container takes the picture of the status of the bands, detects the bags present and writes a file with the order of colors that need to be deposited. The order is established in relation to the proximity with the robotic arm.
4. Now that the bags are detected, the core program in the Nano Jetson server proceeds to read the file that the Docker program wrote to coordinate bands and robot movement.
5. For each color detected the Nano Jetson commands the bands to move. The bands move until the ultrasound sensors are interrupted. Afterwards execution in the band ends and continues in the Nano Jetson.
6. Once the bag is placed into position, a command to the robot is sent via Sockets to be moved to the location of the specific container for the color.
7. Then, when steps 5 and 6 have been repeated for all of the colors detected, the Nano Jetson sends all of the information to BigQuery through an API. One row for each color detected with the information of the user and datetime registered by the RFID Server.
8. Lastly, the Nano Jetson establishes a connection, via Sockets, with the RFID Server that sent the signal in order to enable it for more tags to be received.

3. Results

This chapter intends to show the final deliverables that were made, based on the architecture design and the results of the executions of the project.

3.1 Pseudocode

All of the following programs were made in Python, a list of the links to all the real implementations can be found in Annex A. However, the following figures serve as an overview of the overall behavior, in a more technical sense than the step by step listed in the architecture design. Every external method find within the procedures explains code that is very specific to connection or sensor management, i.e., code for initializing and binding sockets, that was found irrelevant to explain the functioning of the architecture.

3.1.1 Pseudocode for Nano Jetson Server

The code below corresponds to the core orchestration made by the Nano Jetson with all the other systems in the architecture. Lines 1-3 correspond to the initial steps needed to define the table schema in BigQuery. Line 2 describes the columns that are used for the main analysis table: RFID Tag, the color of the bag, how many bags for this color were deposited on one full execution and the datetime of this execution. Line 4 is crucial to identify the Docker container that will run the object detection model. Line 5 establishes SSH connection with the bands to order them to start the motion. After that, in line 6, the server-like behavior begins (infinite loop, as an infinite number of users can use the architecture for garbage disposal).

Procedure Nano_Jestson_Final_Integration (Bands_File_Run_Until_Sensor_Interrupted, Docker_Objectc_Detection_File, Shared_File_Docker, Robot_IP, Bands_IP)

```

1. Bigquery_API.Establish_Connection_Bigquery()
2. Column_Schema = [RFID_TAG, Bag_Color, Bag_Count, Date_Registered]
3. Bigquery_API.Create_Table_If_Does_Not_Exist(Column_Schema)
4. Docker_ID ← Subprocess.Run(Find ID of Current Docker Running)
5. Bands_API.Establish_SSH_Connection_Bands(Bands_IP)
6. Repeat (Server Type Behavior: Always Running)
7.     Socket.Set_Socket_On_Listenning_Port()
8.     RFID_IP, Message ← Socket.Accept_Message_Close_Connection()
9.     RFID_TAG, DATE ← Message.Split()
10.    Base_Dictionary ← {'RFID': RFID_TAG, 'DATE_REGISTERED': DATE}
11.    Subprocess.Run(Run Docker_Objectc_Detection_File in Docker With Docker_ID)
12.    Bags_Colors_Ordered ← Read_Resulting_File_As_Array(Shared_File_Docker)
13.    Bag_Color_Count = {'Black':0, 'White':0, 'Green':0, 'Undefined':0,}
14.    Robot_API.Establish_Connection(Robot_IP)
15.    For Bag_Color in Bags_Colors_Ordered Do
16.        Bands_API.Execute_Command(Bands_File_Run_Until_Sensor_Interrupted)
17.        Robot_API.Deposit_Bag(Bag_Color)
18.        Bag_Color_Count[Bag_Color] =+ 1
19.    Endfor
20.    Robot_API.Close_Connection()
21.    For Color in Bag_Color_Count_Dict Do
22.        Bag_Color_Registry ← Base_Dictionary.copyWith(Color=Bag_Color_Count[Color])
23.        Bigquery_API.Register_New_Row(Bag_Color_Registry)
24.    Endfor
25.    Socket.Connect(RFID_IP)
26.    Socket.Send_And_Close_Connection('STATUS:OK')
27. End
28. End Nano_Jestson_Final_Integration

```

Figure 2. Pseudocode for Nano_Jetson_Final_Integration.

Lines 7-9 correspond to the initial connection with the RFID Module. First the Nano Jetson waits for the message, then it establishes it with the RFID Server that wants to communicate, annotates the IP to respond later and separates the message received to use for uploading the table info. Line 10 is intended for creating a base dictionary, this is the way that rows are sent through the BigQuery API, one copy of this dictionary will be created later for every color deposited. Line 11 starts the Object Detection program as a subprocess in the Docker container identified earlier in line 4. Once that program has run and written the order of the

bags detected in a shared folder, the Nano Jetson server reads it in line 12. The results are read as an array, which is already ordered depending on the closest colors to the robotic arm. Line 13 creates a counter for each garbage bag color possible, in order to later create the row dictionaries that will be sent to BigQuery. This serves the purpose of keeping a number in memory, rather than sending a request to the Analytics engine for every bag deposited in the next loop. Line 14 represents connecting with the UR3 Robot through Sockets. Lines 15-19 represent the actual garbage disposal behavior. For each bag color, the bands are moved to place it for the robotic arm to grab it, the robot goes to the container of that specific color and then the counter for that specific color is increased. Line 20 closes connection with the Robot. Lines 21-24 are about creating a copy of the Base Dictionary from line 10 with the color count from line 13 to insert a row in BigQuery for each color. Lines 25 and 26 are about connecting with the RFID Server that sent the request to enable it again for more tag reception.

3.1.2 Pseudocode for Docker Container with the PyTorch Object Detection Model

Procedure Docker_Object_Detection_File (*Shared_File_Docker, Camera, Detection_Model_Directory, Labels_Path*)

1. Best_Min_Loss = 1000
2. Best_Model_Path
3. **For** Resulting_File in *Detection_Model_Directory* **do**
4. Current_Loss = Read_Loss(Resulting_File)
5. **If** Current_Loss < Best_Min_Loss **then**
6. Best_Model_Path = Resulting_File
7. **EndIf**
8. **EndFor**
9. Class_Labels ← Read_As_Array(*Labels_Path*)
10. SSD_MobileNet ← NVDLI_API.Create_Mobilenetv1_SSD(Best_Model_Path, SizeOf(Class_Labels))
11. Predictor ← NVDLI_API.Create_Mobilenetv1_SSD_Predictor(SSD_Mobilenet)
12. Image ← OpenCV.VideoCapture(*Camera*).read()
13. Labels_Present_In_Camera ← Predictor.Predict(Image)
14. Labels_Present_In_Camera.Sort(Closest to Robot Arm on X axis)
15. Write_Labels(*Shared_File_Docker*)
16. **Return**
17. **End** docker_controller_API

Figure 3. Docker Container with PyTorch Model

The figure presented above holds the representation for the Docker container with the Object detection model, implemented in PyTorch. In line 1 a variable to find the best loss possible among the trained epochs of the model. Line 2 initializes a variable to find this best model in the files. Lines 3-8 iterate over the trained models and finds the best possible one depending on the loss it had. This allows for retraining, and using in the architecture automatically, the detection model to improve performance. For line 9 the program reads the labels for this model. Line 10 initializes the Convolutional Neural Network (CNN) with the model found in line 6, for this project the Mobilenetv1 architecture was used to detect the objects. Line 11 creates the predictor to read the camera input and detect objects based on the CNN of line 10. Line 12 specifies to take a picture. Line 13 detects the labels present in the bands. Then, at line 14 the labels are sorted depending on the proximity, of the X axis, in regards to the robot. Line 15 represents writing the file, with the labels ordered, in the shared spaced folder between the Docker container and the Nano Jetson itself.

3.1.3 Pseudocode for the RFID Servers

The following code is much more straightforward, since its main concern is communication, and the user recognition part is simplified by the RFID antenna that detects the tags.

Procedure RFID_module_communication(***Nano_Jetson_IP***)

1. **Repeat (Server Type Behavior: Always Running)**
2. RFID_Tag ← Received Input Signal from Antenna
3. Date_Registered ← DateTime.Now()
4. Message_To_Nano ← Concatenate(RFID_Tag, Date_Registered)
5. Socket.Connect(***Nano_Jetson_IP***)
6. Socket.Send_And_Close_Connection(Message_To_Nano)
7. Socket.Set_Socket_On_Listenning_Port()
8. Continue ← Socket.Accept_Message_Close_Connection()
9. **End**
10. **End** RFID_module_communication

Figure 4. Pseudocode for the RFID Server.

It is always in an infinite loop since an infinite number of users can deposit bags indefinitely. Line 2 begins with the reception of the RFID tag in the antenna. Line 3 registers the current date and time. Line 4 creates the message that needs to be sent to the Nano Jetson server in order to start depositing bags. Line 5 creates the connection and in line 6 the message

is officially sent. Line 7 sets the RFID module to listen until it receives a message from the Nano Jetson in line 8 to continue receiving tag signals.

3.1.4 Pseudocode for the Bands

Procedure Bands_File_Run_Until_Sensor_Interrupted(**UltraSound_Sensor_Max_Distance**, **MaxSpeed**)

```

1. UltraSound_Sensor
2. Repeat(OnTimer Continuously Check Band Status)
3.   If UltraSound_Sensor.Distance < UltraSound_Sensor_Max_Distance
4.     MoveBandsAtSpeed(0)
5.     ExitWithStatus(0)
6.   Else
7.     MoveBandsAtSpeed(MaxSpeed)
8.   EndIf
9. End
10. End Bands_File_Run_Until_Sensor_Interrupted

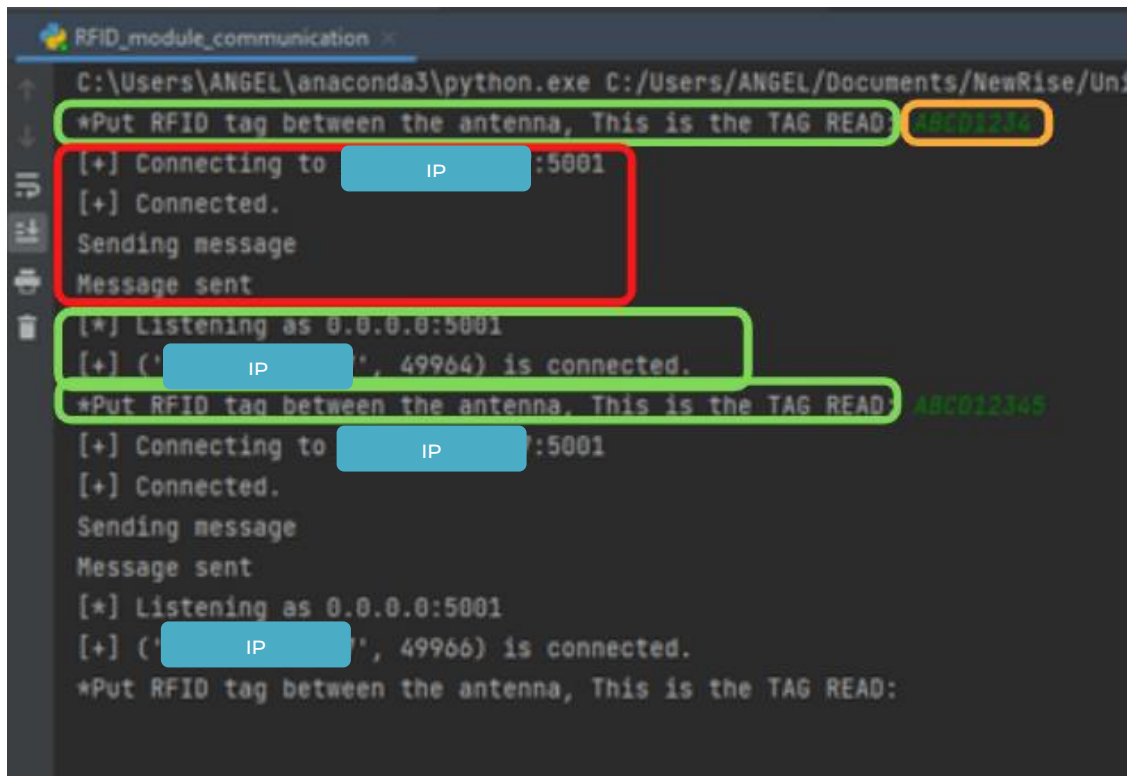
```

Figure 5. Bands Pseudocode

The figure above describes the behavior of the bands, which is simple. As the title denounces it, the bands move until the sensor is obstructed by an object which, in this case, are bags. Line 1 initializes the sensor. Line 2 declares the loop to stay attentive to changes in it. Line 3 declares a condition. If the distance in the sensor is reduced, the bands stop in line 4 and the program finishes in line 5 so that the program in the Nano Jetson can keep the execution going. Otherwise, the bands keep moving at the determined speed in line 7.

3.2 Work Progress and Evidence

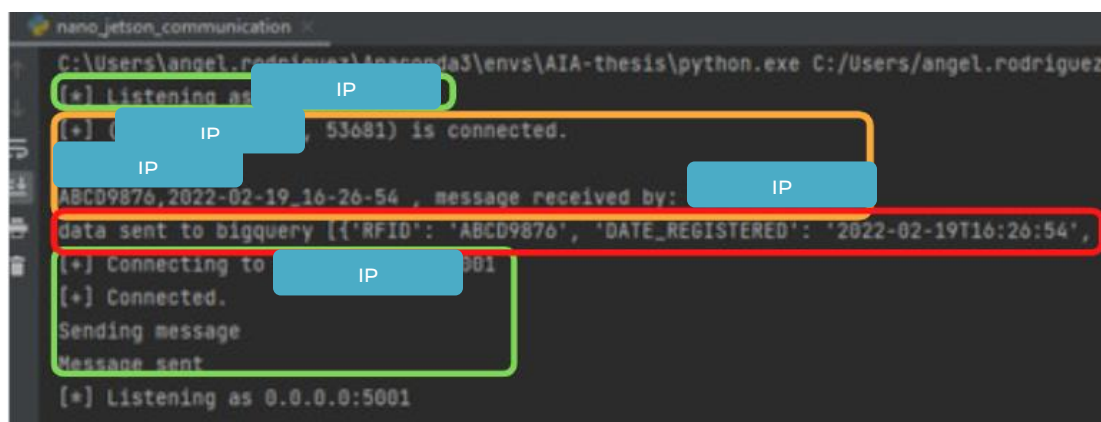
The first step for the project was to establish a communication model between the RFID server and the Nano Jetson. In order to establish an on-going, dynamic and simple protocol the decision was to use Sockets for this task. As it was defined before in the pseudocodes, the RFID first waits for the input of the Antenna. Once it gets it, it connects to the Nano Jetson, and sends a message with the tag and datetime. Conversely, the Nano Jetson always starts listening, then when the RFID server wishes to communicate, it connects and gets the message. See figures 6 and 7 for reference.



```
RFID_module_communication x
C:\Users\ANGEL\anaconda3\python.exe C:/Users/ANGEL/Documents/NewRise/Uni
*Put RFID tag between the antenna, This is the TAG READ: ABCD1234
[+] Connecting to IP:5001
[+] Connected.
Sending message
Message sent
[*] Listening as 0.0.0.0:5001
[+] (IP, 49964) is connected.
*Put RFID tag between the antenna, This is the TAG READ: ABCD12345
[+] Connecting to IP:5001
[+] Connected.
Sending message
Message sent
[*] Listening as 0.0.0.0:5001
[+] (IP, 49966) is connected.
*Put RFID tag between the antenna, This is the TAG READ:
```

Figure 6. Execution Result of RFID Communication.

The first long green rectangle corresponds to when the RFID is waiting for input. On yellow, to the right side of it, the input finally comes in. Next, on red, the connection is established with the Nano Jetson, and the message is sent. Then, on a taller green rectangle, the RFID server waits for response from the Nano Jetson. Once it gets the 'OK' message, the RFID is enabled to get more tags and the process keeps repeating itself.



```
nano_jetson_communication x
C:\Users\angel.rodri...\anaconda3\envs\AIA-thesis\python.exe C:/Users/angel.rodri...
[*] Listening as IP
[+] (IP, 53681) is connected.
IP
ABCD9876,2022-02-19_16-26-54 , message received by: IP
data sent to bigquery [{ 'RFID': 'ABCD9876', 'DATE_REGISTERED': '2022-02-19T16:26:54',
[+] Connecting to IP:5001
[+] Connected.
Sending message
Message sent
[*] Listening as 0.0.0.0:5001
```

Figure 7. Execution result of the Nano Jetson server.

Here, the Nano Jetson server starts listening. In the yellow rectangle, the RFID has communicated, and the message has been read. The red rectangle depicts the next step in the process, which was to send data to BigQuery. Lastly, the Nano Jetson communicates with the RFID server and enables the process for another user.

On figure 8, displayed below, is the visualization report created to analyze the data of the process (sent by the Nano Jetson server, through an API).

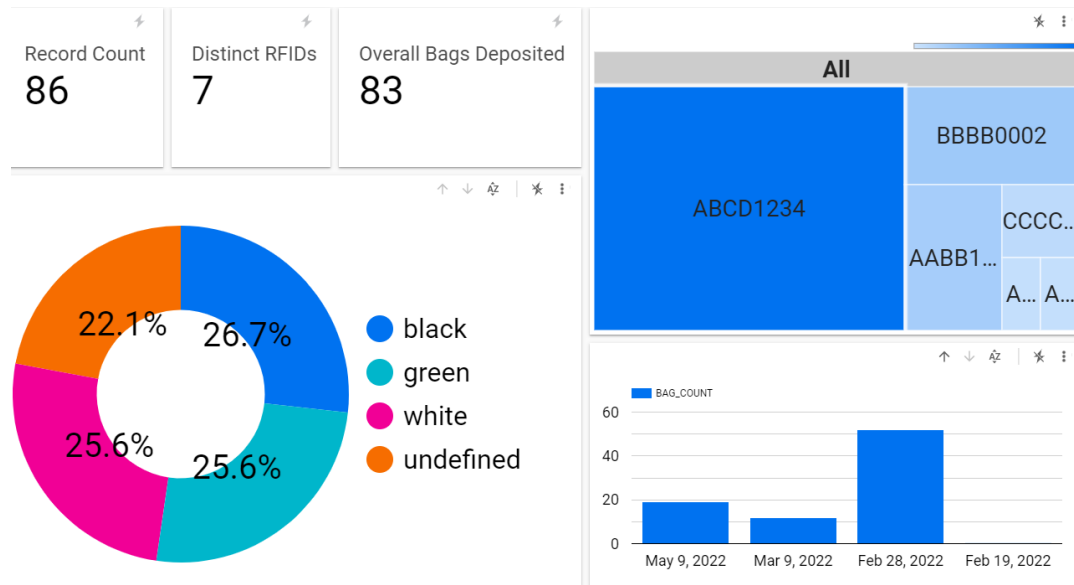


Figure 8. Report for Analysis of the Waste Management Process.

Once the behavior of the RFID Module was finished, and there was progress for both the Nano Jetson and the Analytics Engine, it was time to integrate the movement of the Robotic Arm and moving the bands to place the bags in pickup position. Figure 9 represents three moments of the deposit process. First, the bags reach the bands (Image on the left), after the RFID sends the tag to the Nano jetson server. The communication is established via SSH through the Paramiko library. It allows for the connection with the TXT controller of the bands, and it enables sending shell commands. This is how the Nano Jetson is able to execute the Python file that orders the bands to move until there is an object. Once it finishes execution, because a bag interrupts the ultrasound sensor (image on the center), the Nano Jetson commands the robot to pick up the bag, then, depending on the color, it tells it to go to a specific container to deposit the bag. If there are more bags, the process continues, until all the bags have been deposited.

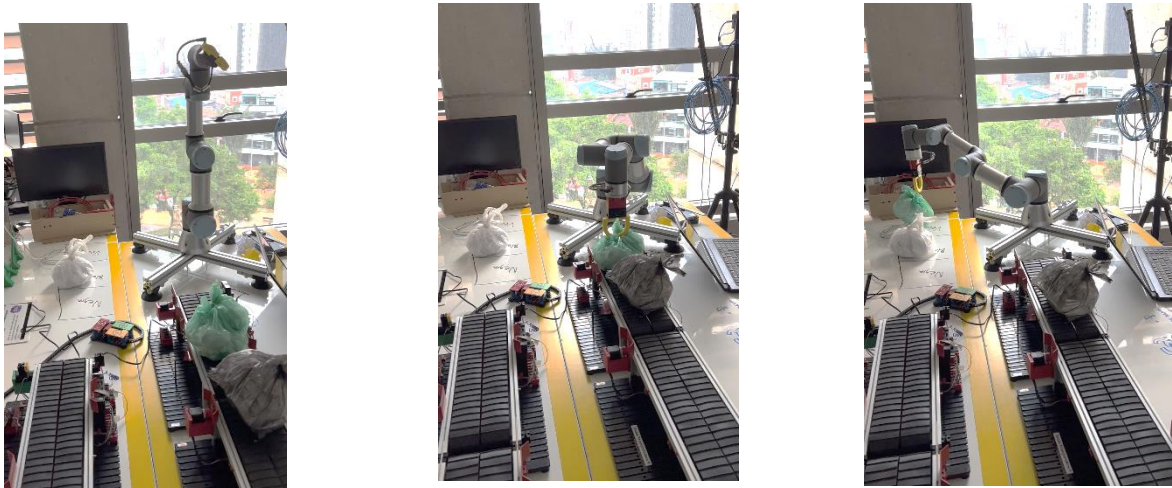


Figure 9. Three Moments of the Deposit Process. (Left) The bags get to the bands, (Center) bands place the bags in position for pickup. (Right) Robot deposits the bag.

Up until this point, a full execution can be made, and the architecture would appear to be working as expected. But the configuration of the bags was already predefined, it would only work for a specific number of bags and colors, in a particular order. So, it was necessary to implement the final component of the architecture: the object detection model. The first step for this was to train and test the model. The sample consisted of 300 pictures with all four types of bags that could be detected: black, white, green and undefined (for bags not compliant with the color-coding schema determined by the Government of Colombia). Every picture with a small variation, i.e., lighting, placement of the bags and order in which they were placed on the bands. With this configuration, only 3 epochs were needed to get the results found below in figure 10. Each box has the bounding box defined by the model and the confidence probability assigned to that detection.

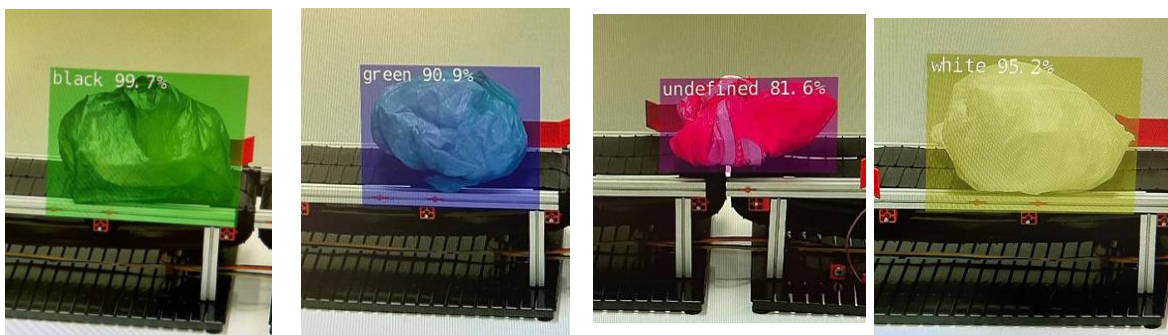


Figure 10. Results of the Object Detection Model.

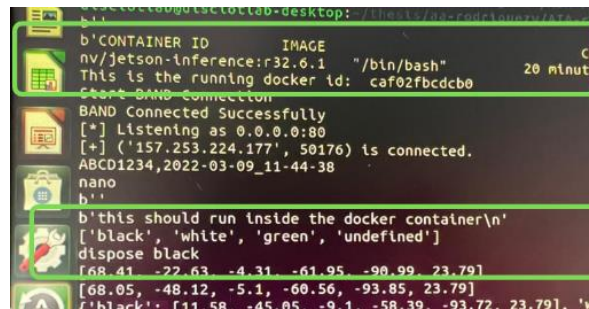
The image above corresponds to a screenshot of the detection model running on a live camera stream on the Nano Jetson server. Table 1, shown below, lists the configuration of the detection model that led to the results in figure 10.

	Metric	Measure
Train	Images Sample	300
	Manually Drawn Boxes	1200
	Box/Class	300
	Epochs Ran	3
Result	Best Loss Acquired	1.28
	Best Epoch	#3
	Avg. Inference Time	109s

Table 1. Configuration and Performance Results of the Object Detection Model.

It is worth reminding that the model was implemented using PyTorch, using the SSD Mobilenet V1 CNN Architecture. The process of shooting the photos, labeling and effectively training, and validating, the model took approximately 8 hours.

Though having the object detection model made for great progress, having a standalone model would make no impact on the architecture without communication. The next, and last step, in the project was to establish a way in which the model could be used, based on the current status of the bands, and the core program running in the Nano Jetson could read the detection to proceed with depositing the bags. For this to be achieved, first it was necessary identify the Docker container running on the Nano with the detection model the figure 11 below shows an execution where the Nano Jetson, running a subprocess, identifies the container.



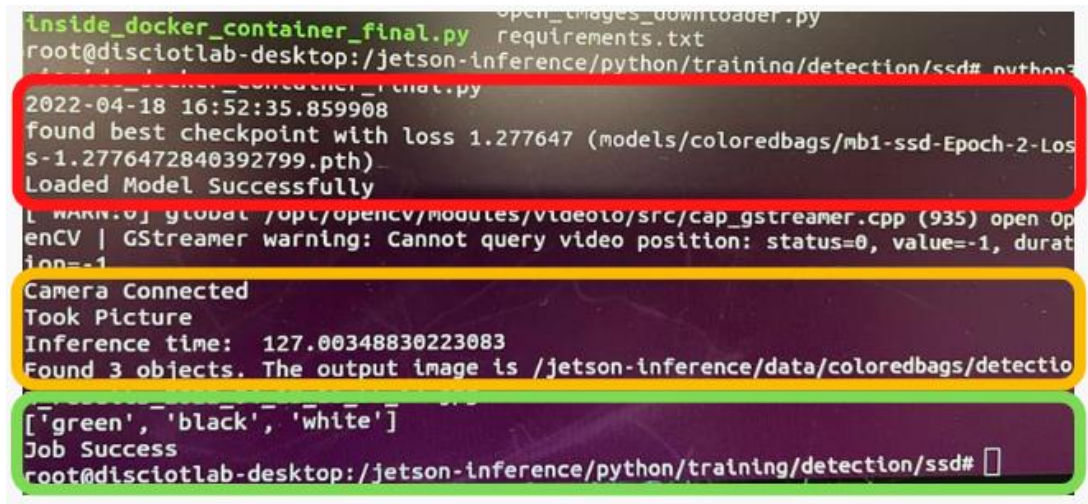
```

b'CONTAINER ID        IMAGE                                     COMMAND                  Status
nv/jetson-inference:r32.6.1   "/bin/bash"                  20 minutes ago
This is the running docker id: caf02fbcdbc0
Start BAND Connection
BAND Connected Successfully
[*] Listening as 0.0.0.0:80
[+] ('157.253.224.177', 50176) is connected.
ABCD1234,2022-03-09_11-44-38
nano
b''
b'this should run inside the docker container\n'
['black', 'white', 'green', 'undefined']
dispose black
[68.41, -22.63, -4.31, -61.95, -90.99, 23.79]
[68.05, -48.12, -5.1, -60.56, -93.85, 23.79]
['black': [11.58, -45.05, -9.1, -58.39, -93.72, 23.79], 'w

```

Figure 11. Nano Jetson server identifying the running.

Once the communication was possible, it was time to start using the object detection model in the architecture. Figure 12 shows an execution of the program inside the Docker container that loads the model (red rectangle), takes a picture, detects objects (yellow rectangle) and writes the result to a file in a shared space folder (green rectangle).



```
inside_docker_container_final.py open_images_downloader.py
root@disclotlab-desktop:/jetson-inference/python/training/detection/ssd# python3
2022-04-18 16:52:35.859908
found best checkpoint with loss 1.277647 (models/coloredbags/mb1-ssd-Epoch-2-Loss-1.2776472840392799.pth)
Loaded Model Successfully
[ WARN.0] global /opt/opencv/modules/videoio/src/cap_gstreamer.cpp (935) open OpenCV | GStreamer warning: Cannot query video position: status=0, value=-1, duration=-1
Camera Connected
Took Picture
Inference time: 127.00348830223083
Found 3 objects. The output image is /jetson-inference/data/coloredbags/detection/
['green', 'black', 'white']
Job Success
root@disclotlab-desktop:/jetson-inference/python/training/detection/ssd#
```

Figure 12. Execution inside the Docker container.

The execution above resulted from the picture shown below in figure 13.



Figure 13. Status of the Bands for Execution of Figure 14.

Finally, as an added bonus for the project, to verify the performance of the model in the architecture itself (not just on a live camera stream running independently, like in figure 10) the program running inside the docker container writes the captured image with the bounding boxes and the confidence probability for the bags detected.



Figure 14. Model Performance Evidence in the Architecture.

Figure 14 shows the captured image in one of the executions made and it is worth noting that these images, like the result file the Nano Jetson reads, were also placed in a shared spaced folder for persistence.

4. Conclusions and Future Work

4.1 Conclusions

From the results the main conclusion drawn is that it is possible to implement an IoT architecture that allows for the automation and analysis of the process for classifying and separating garbage bags. The solution presented is resilient and robust enough to serve its purpose well and allows for improvements and incremental developments to be built on top. Another conclusion that can be drawn from the project is that, building an object detection model for similar elements that only differ in color does not require a great and significant effort (in this case it took a day's work in time and a fairly small sample of images). However, in order for this project to be a solution ready to be launched to the public and implemented in real buildings there are some considerations and further implementations that need to be considered.

Currently, there is a need for the RFID antenna to be connected to another server, but this could increase the price of implementation unnecessarily, as its only job is to communicate the date of registry and the incoming tag to the Nano Jetson. Another problem that arises with users is related to the concurrency. For now, the architecture can hold one user at a time. But, virtually, an infinite number of users could deposit bags at the same time. With it, not only it becomes a problem to receive multiple users at the same time, but identifying which bags belong to each user becomes another challenge. Furthermore, there are some improvements that can be made related to connection. For instance, every IP on the architecture is pre-defined, but there could be mechanisms that can be developed for the Nano Jetson to identify the Robot and the Bands within the same network. Additionally, there could be more advanced developments regarding the interaction with the Robotic Arm and the bags. The robot could optimize the placement of the grip to grab the bags and automatically find the container that matches with the color of the bag. Finally, testing

different detection models and increasing the sample to improve performance could be of great help to reduce the inference time and make the process run faster.

4.2 Future Work

Based on the conclusions established above, the following items deposit a set of possible developments that could improve the architecture:

- Designing an RFID Server for multiple antennas: The need for a server connected to one particular antenna, through USB, can be pricey, but having multiple antennas converging to only one server per building might reduce costs and could even improve performance of the architecture.
- Establish a Publisher-Subscriber model for concurrency: For the project to be ready to make an impact in a real-world scenario the need for concurrency is imperative. For this a model of publisher and subscriber threads is proposed.
- Integrate more sensors to distinguish bags for each user: Once concurrency is enabled, there needs to be a way to associate bags running through the same vent to each concurrent user. For this project, the use of an infrared light sensor was discussed but not implemented.
- Building the Object Detection Model with different architectures and new samples: Since the main purpose for this project was to get the implementation done, rather than performance, the SSD MobileNet V1 architecture was picked, without much deliberation, it was considered because it is widely used and supported. But there can be tests to prove which CNN is better or faster. YOLOv3 was also considered.
- QA Systems: Given that the model already saves images with the detection of the bags and probabilities for the classes assigned, there could be a system built on top for users to verify the performance and quality of the model. This model could be retrained and improved to be re-introduced in the architecture with higher accuracy.

A. Anex: Real Code Implementations

- Implementation of the RFID Server: https://github.com/aa-rodriguezv/AIA-robot-project/blob/master/communication_trials/RFID_module_communication.py
- Implementation of the Nano Jetson Server: https://github.com/aa-rodriguezv/AIA-robot-project/blob/master/communication_trials/nano_jetson_final_integration.py
- Implementation of the Bands Behavior: https://github.com/aa-rodriguezv/AIA-robot-project/blob/master/miscellaneous/OneStopUltraSound/one_stop_ultrasound.py
- Implementation of the Object Detection Model: https://github.com/aa-rodriguezv/pytorch-ssd-modified-AIA-thesis/blob/master/inside_docker_container_final.py

The code listed above may use other libraries and specific APIs, but these can be found in the projects that contain them. The modularity design was intended for the code to be as representative of the behavior of each component as possible, leaving connection-specific commands in separated files.

The following repositories were used to leverage the PyTorch and BigQuery technologies in the project:

- <https://github.com/dusty-nv/pytorch-ssd>
- <https://github.com/tylertreat/BigQuery-Python>

Bibliography

- [1] ORACLE Corp. (N.D.) What is IoT? Retrieved from: <https://www.oracle.com/co/internet-of-things/what-is-iot/>
- [2] Mitra, A. (2020). Detection of Waste Materials Using Deep Learning and Image Processing. Retrieved from: <https://scholarworks.calstate.edu/downloads/gx41mn74q>
- [3] Arebey, M. Hannan, M. Begum, R. Basri, H. (2012). Solid waste bin level detection using gray level co-occurrence matrix feature extraction approach. Journal of environmental management, 104, 9–18.
- [4] Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto Mand Adam H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. Retrieved from: <http://arxiv.org/abs/1704.04861>
- [5] E Suharto et al. (2020). J. Phys.: Conf. Ser. 1524 012105
- [6] Pineda, J. (2017). El problema Ambiental de la Basura. Disponible en: <https://encolombia.com/medio-ambiente/interes-a/problema-ambiental-basura/>
- [7] Vera, K. T. (2021). Año nuevo con nuevo código de colores para separar residuos. Pequisa Javeriana. Retrieved from: <https://www.javeriana.edu.co/pesquisa/ano-nuevo-con-nuevo-codigode-colores-para-separar-residuos-en-colombia/>
- [8] VIDA. (2019). Se usarán tres colores para reciclar en todo el país. El Tiempo. Retrieved from: <https://www.eltiempo.com/vida/medio-ambiente/blanco-negro-y-verde-nuevo-codigo-decolores-para-reciclar-447228>
- [9] S.A. (2021). Manejo de Basuras, Una problemática en Colombia. Medio Ambiente. Retrieved from: <https://www.kienyke.com/medio-ambiente/manejo-de-las-basuras-una-problematica-encolombia>
- [10] S.A. (2021). Problemática por basuras: voces de expertos, testimonios y cifras. RTVC Noticias. Retrieved from: <https://www.rtvnoticias.com/basuras-colombia-toneladas-manejo-reciclaje>

- [11] S.A. (2019). La mayoría de los colombianos todavía prefieren vivir en casas que en apartamentos. La República. Retrieved from: <https://www.larepublica.co/economia/la-mayoriade-los-colombianos-todavia-prefieren-vivir-en-casas-que-en-apartamentos-2903583>
- [12] Friends of The Earth. (2018). 7 Benefits of Recycling. Disponible en: <https://friendsoftheearth.uk/sustainable-living/7-benefits-recycling>
- [13] S.A. (2021). Tasa de desempleo en Colombia fue de 10,8 % en noviembre del 2021. Portafolio. <https://www.portafolio.co/economia/empleo/tasa-de-desempleo-en-colombia-en-noviembredel-2021-560102>