RAFIK HARIRI UNIVERSITY

Smart Stand

Done by

MANWEL SHDEED

EYAD J. SHAYYA

HADI DERGHAM

ABDULHAKIM HUJAYRI

Submitted to

DR. HASSAN HARIRI

This Senior Project is submitted in Partial Fulfillment of the Requirements of the BE Degree of Mechatronics and Mechanical Engineering Major of the College of Engineering at Rafik Hariri University

MECHREF-LEBANON

JANUARY 2022

Manwel Shdeed (2018-0124)

Eyad J. SHAYYA (2018-0069)

Hadi Dergham (2018-0137)

Abdulhakim Hujayri (2017-0247)

ACKNOWLEDGEMENTS

I'd want to offer our heartfelt thanks to all of the people who have supported us throughout the years, especially our parents. I'd also want to thank our project supervisor, Dr. Hassan Hariri, for his patience, excitement, insightful remarks, important recommendations, helpful information, practical guidance, and never-ending ideas, all of which have greatly aided us during our project.

ABSTRACT

This book explains how to run a supermarket smart stand from start to finish. By collecting data on customers and all items on the shelf, this supermarket smart stand aids in the surveillance of their behavior and all products on the shelf. Furthermore, this book depicts the entire process of constructing the Smart supermarket stand (research, design, and simulation) with its surveillance tools, including outline, mechanical and electrical parts, figuring, and codes, as well as a complete depiction of the smart supermarket stand's equipment arrangement.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

A. CHAPTER 1:

INTRODUCTION

1.1 Background Information

Supermarket becomes a primary place in our lives and most of the days do not go without shopping. In traditional stores, owners must worry about a lot of things such as arranging shifts, labor costs, stock quantity. But today, a lot of technologies like sensors, Internet of things, and many others are emerged to support a smart supermarket stand, where a brand manager of a product have the chance to improve their profits by increasing sales, and at the same time, shoppers and merchandisers get a lot of facilitations during their presence in the supermarket. There is a wide variety of smart stands like “AWM”, “Amazon”, “Wise shelf” … and every brand of smart shelves has its features and technologies. For example, in “Amazon” smart shelves, engineers use load cells on the shelfs with web/mobile interface to extract the data about the quantity of SKUs and send this data to merchandisers to get notifications about the count or quantity of products and the stock on the shelf. So Smart supermarket stand is a technological breakthrough that enhances the overall shopping experience and improves a store’s operational efficiency.

1.2 Motivation

In a normal supermarket stand there is nothing that attracts the shopper, knows what he exactly wants or searching for and the way he interacts with the design of this stand or the products this stand holding, in addition to the way he looks to the products, prices, and offers on this stand. Therefore, we need to build a smart stand that combines all what was mentioned above in addition to some other features.

1.3 Literature Review

Intelligent shelf, often known as The Smart Shelf, is a new technology that will transform the quality of service that merchants can provide. Smart Shelf solutions will provide shoppers with tailored experiences while also offering real-time in-store analytics that will boost sales and efficiency across the chain**.** Intelligent shelving will likely be at the forefront of the value proposition of computing within the shopping experience when the physical and digital worlds converge in retail. Retailers who seize the opportunities given by digital retail technologies will be able to ride a tsunami of innovation that will carry them into the next several decades. By dramatically improving the customer experience and giving higher-quality insights, they will gain a major competitive advantage. Retailers should consider the Smart Shelf as a platform for developing and delivering new products and services, such as at-shelf advertising, dynamic pricing, virtual sales support, and more.

This report introduces the design, methodology used and the experimental verification of the smart stand. In addition to that, the components, codes, and constraints are also introduced. The smart stand is expected to be beneficial in delivering a better customer experience and enhanced corporate efficiency.

Currently there are many companies that uses smart stands, each has its different technologies and features. However, some of these companies faces some sensors errors like accuracy, durability...

The technologies used by the companies and its features:

i. SHOPPERCEPTION[1]:
1.1.Features:

- Capture user movements.
- Gives heat map of most picked shelves
- Gives number of SKUs picked
- Gives which SKU picked

1.2.Technology Used:

- RGB-D camera.

- Human Segmentation (A.I).
- Depth image to know from which stand product is taken.
- Processor for image/depth acquisition and ML model.

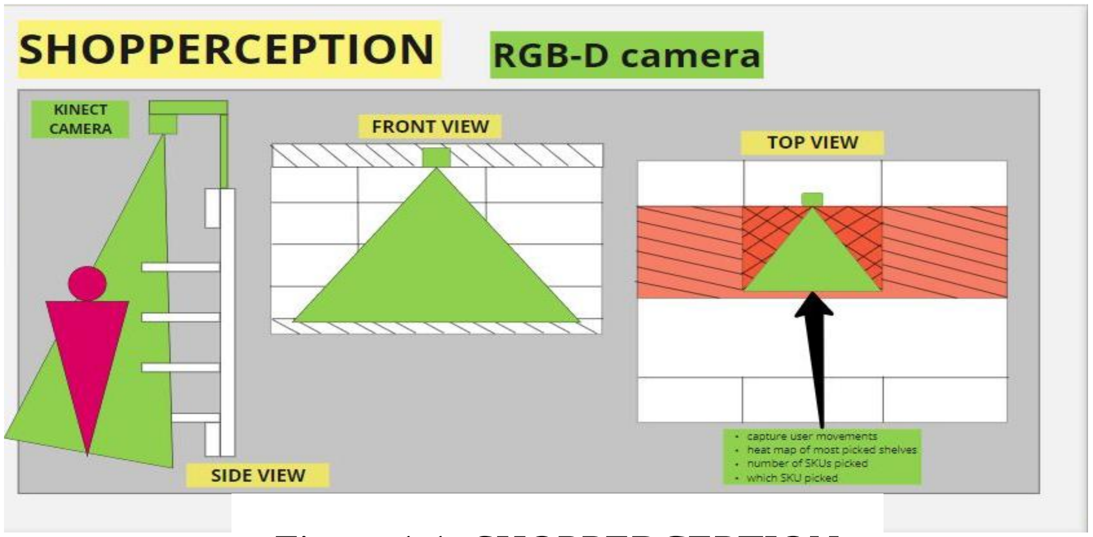


Figure 1.1: SHOPPERCEPTION

ii. AWM[2]:

2.1. Features:

- Motion triggering
- Demographic data collection
- SKU and planogram monitoring
- Promoting displaying (enticing)
- Out of stock notifications
- Planogram displaying (for merchandisers)

2.2. Technology Used:

- Wide angle RGB camera
- Object recognition for SKUs
- Human detection for demographics and motion triggers
- Shelf edge and header displays (LCD or OLED displays)

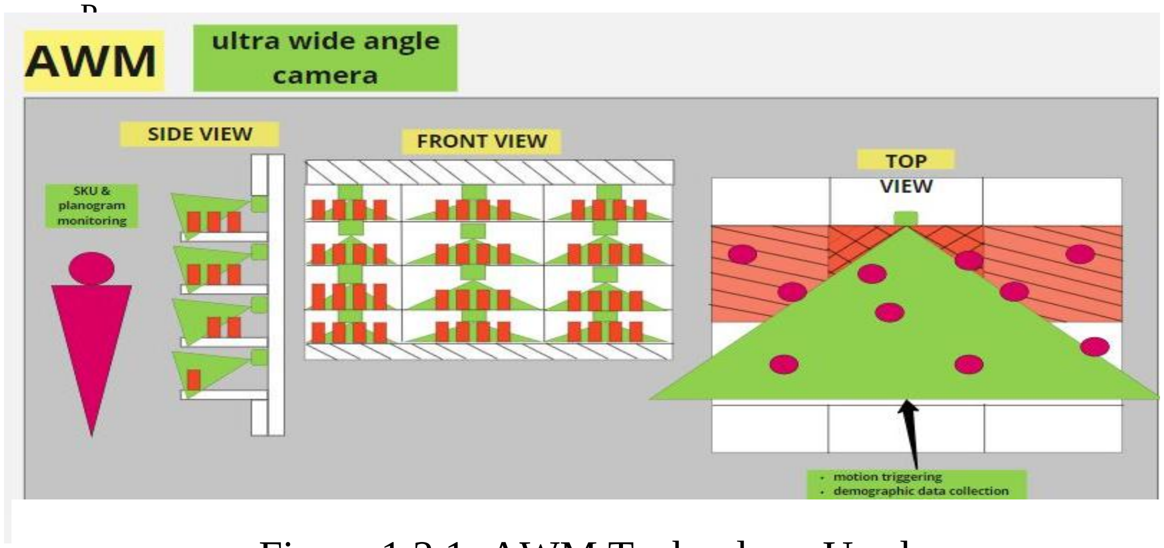


Figure 1.2.1: AWM Technology Used.

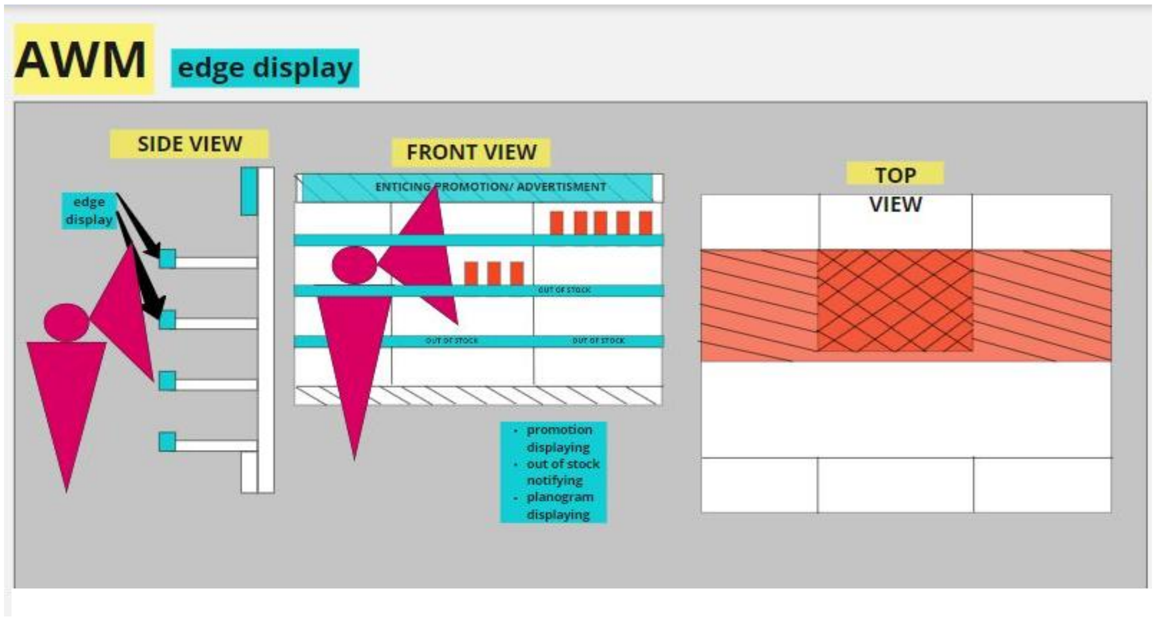


Figure 1.2.2: AWM Technology Used

iii. THINGSQUARE[3]:

3.1. Features:

- Detects on shelf SKU count
- Gives restock notifications
- Counts SKUs removed over time

3.2. Technology used:

- Distance sensor
- Processor
- I

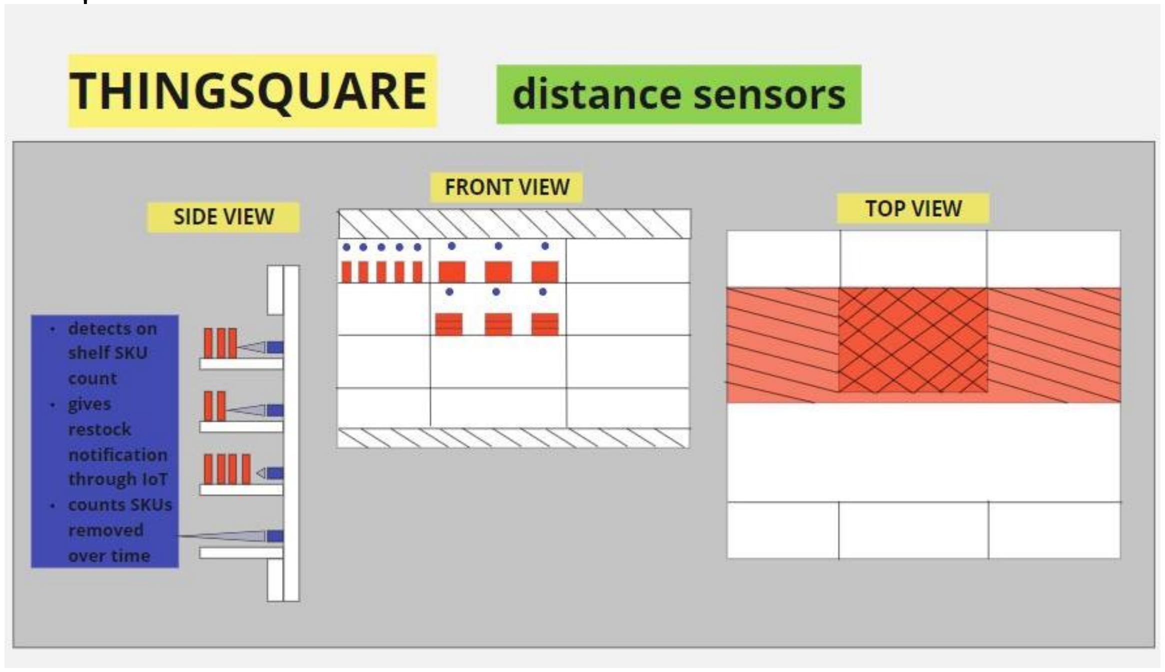

THINGSQUARE
distance sensors
SIDE VIEW
FRONT VIEW
TOP VIEW
detects on shelf SKU count
gives restock notification through IoT
counts SKUs removed over time

Figure 1.3: THINGSQUARE Technology used

iv. EYEWARE[4]:

4.1. Features:

- Advertising and marketing improvement by knowing what the users are interacting and enticing
- Real time customer in store tracking
- Know how the user scans the stand
- Multi-person eye tracking

4.2. Technology used:

- RGB-D camera
- Eye tracking A.I
- processor

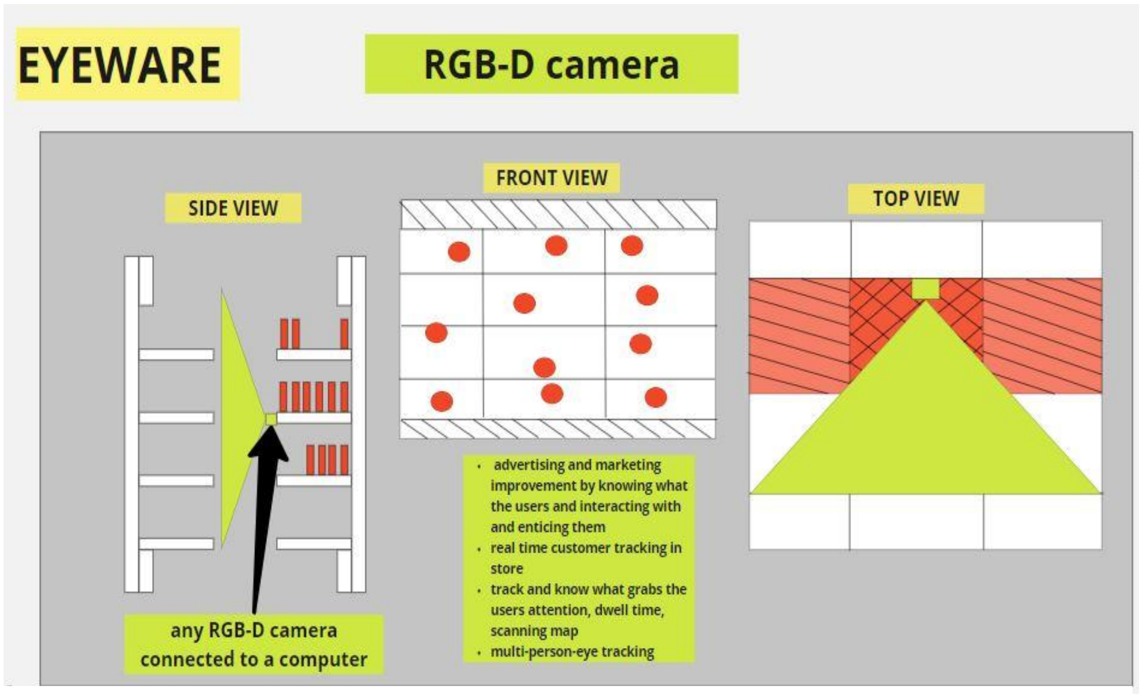

EYEWARE
RGB-D camera
SIDE VIEW
FRONT VIEW
TOP VIEW
any RGB-D camera connected to a computer
advertising and marketing improvement by knowing what the users and interacting with and enticing them
real time customer tracking in store
track and know what grabs the users attention, dwell time, scanning map
multi-person-eye tracking

Figure 1.4: EYEWARE Technology Used

B. CHAPTER 2:

Project defining and problem decomposition

This chapter includes the processes of obtaining the project aim, and that is based on the needs of stakeholders: brand managers, merchandisers, and shoppers.

2.1 Interview questions:

In any product development process, customers' needs are very important to determine and find solutions for the products' weaknesses or problems' and improve the performance of this product. So, the features of the smart stand came from the aspirations and pains of people dealing with a supermarket stand in their daily lives. The supermarket stand is part of the journey of 3 categories of people: brand managers, merchandisers, and shoppers. The survey of questions asked to each category is in the list below:

i. Questions for brand manager:

When planning your presence at the point of sale, what are the key metrics that you look at?

What stages of planning do you go through? What are you trying to achieve during each of these stages?

In your opinion, what are the decisions that you make at the shelf level that can improve your brand's noticeability at the point of purchase?

What are the key variables that you look at when you are planning your planogram?

What different types of SKU you use?

What mechanics do you use to detect and control your near expiry products?

What data available to you today to use while making decisions to your planogram?

What data about your shopper would you ideally like to collect? How does every piece of data collected help you in your decision-making process?

If I tell you that we have an incredibly smart stand that can make your job easier and would magically help you improve your decision making. How do you imagine that stand works? What features would you dream of getting from such a system?

ii. Questions for merchandisers:

Would you please describe to me your typical working day and what you go through from check-in to check-out?

What are the most important tasks you do in order of priority?

What is/are the most challenging task(s) you perform? What is the challenge there? What do you think are ways to solve these challenges?

What suggestions do you have to facilitate your work in the supermarket?

iii. Questions for Shoppers:

How often do you go do grocery? (Daily, weekly, monthly, on demand…)

Do you prefer the supermarket or the grocery store?

How long does it take you to complete a visit to the supermarket?

Do you usually prefer to buy from a specific store? If yes, what is special about this store that makes you like it?

Do you usually stick to your grocery list, or do you like to get surprised with new product suggestions or attractive deals/Specials?

When buying products what are the key factors that drive your decision making?

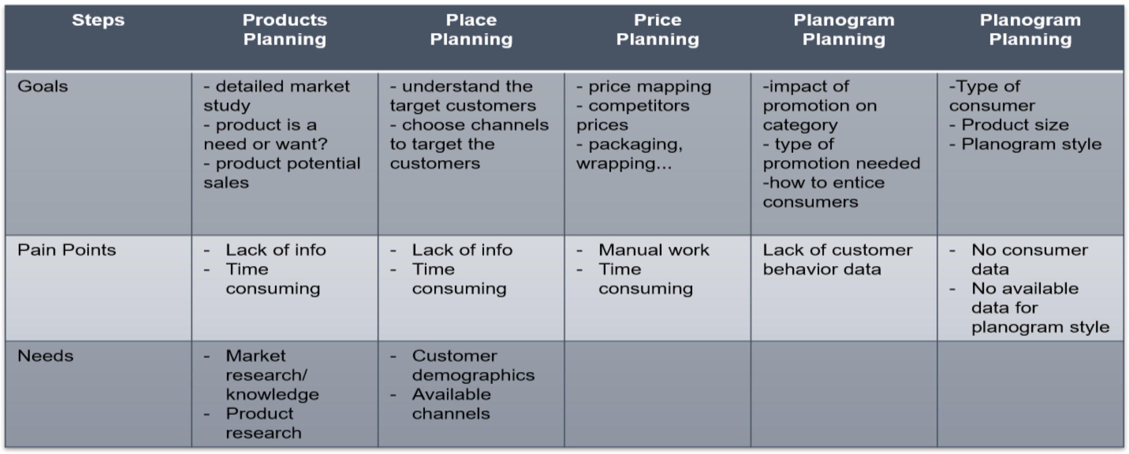
How much time do you usually stand in front of a certain category before making your decision? What are the factors that make you buy a certain product instead of its competition?

2.2 Interviews and brainstorming session:

After interviewing the brand managers of PEPSICO and Ghandour companies, in addition to a group of merchandisers and shoppers, we get their feedback then we traced the journey of both brand manager and a merchandiser:

2.2.1 Brand manager journey:

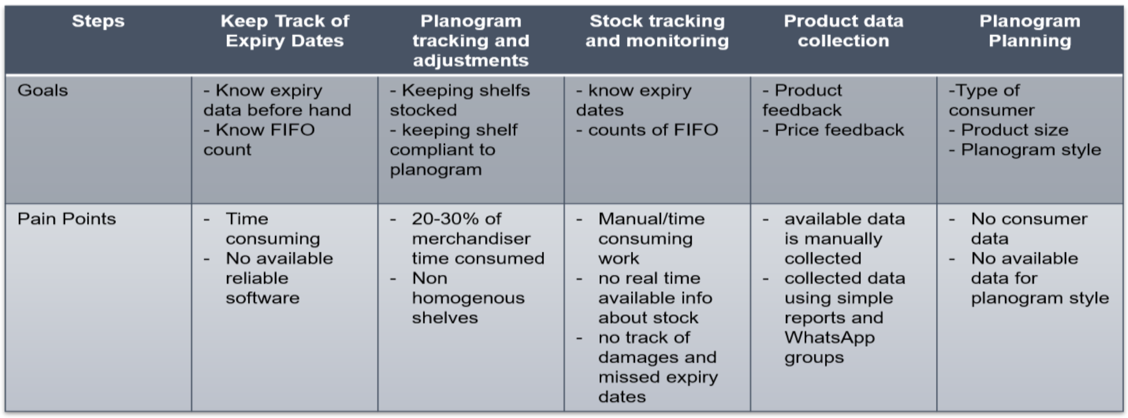
PLANNING PHASE:



Steps	Products Planning	Place Planning	Price Planning	Planogram Planning	Planogram Planning
Goals	- detailed market study - product is a need or want? - product potential sales	- understand the target customers - choose channels to target the customers	- price mapping - competitors prices - packaging, wrapping…	-impact of promotion on category - type of promotion needed -how to entice consumers	-Type of consumer - Product size - Planogram style
Pain Points	- Lack of info - Time consuming	- Lack of info - Time consuming	- Manual work - Time consuming	Lack of customer behavior data	- No consumer data - No available data for planogram style
Needs	- Market research/ knowledge - Product research	- Customer demographics - Available channels			

Figure 2.1: Planning Phase

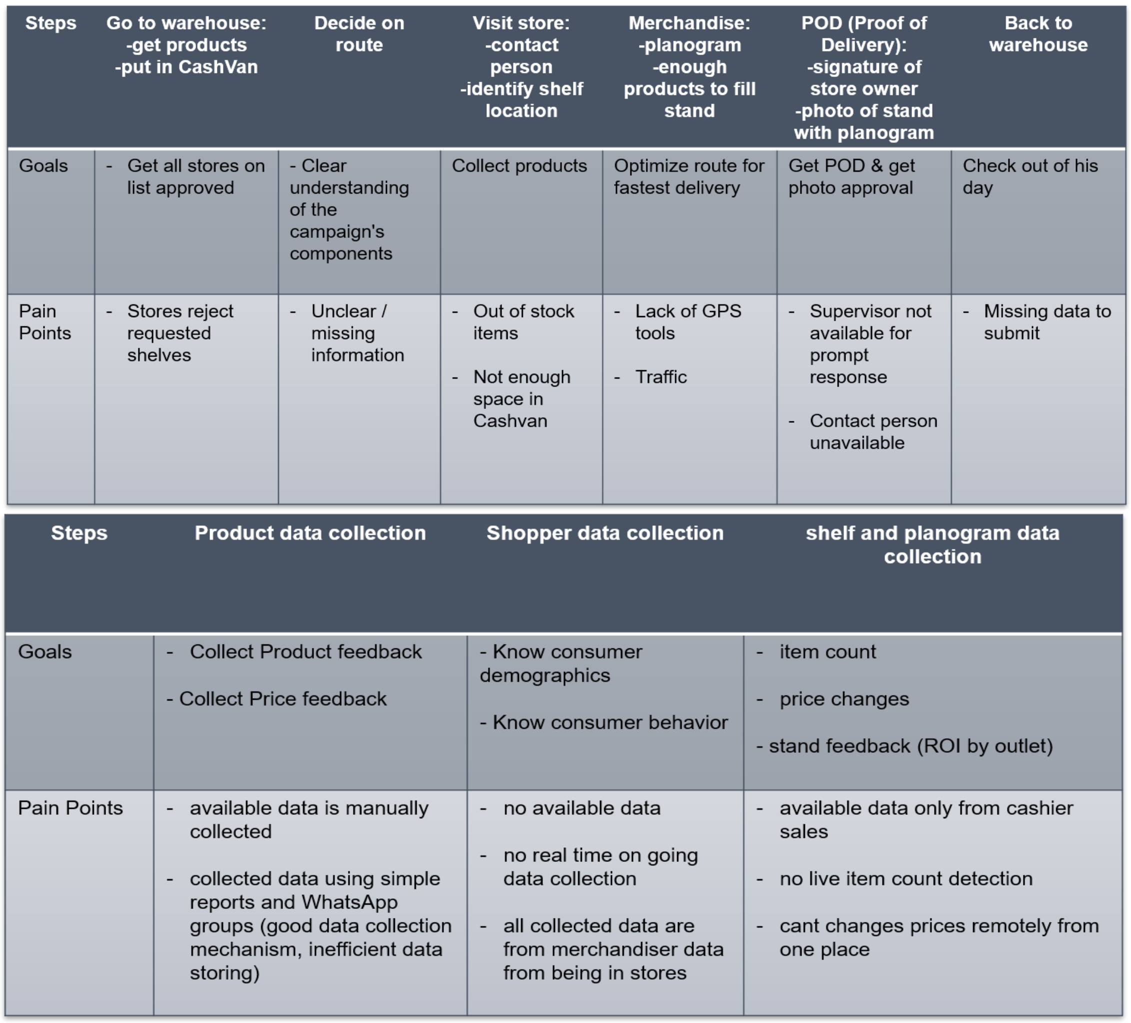
EXECUTION PHASE:



Steps	Keep Track of Expiry Dates	Planogram tracking and adjustments	Stock tracking and monitoring	Product data collection	Planogram Planning
Goals	- Know expiry data before hand - Know FIFO count	- Keeping shelfs stocked - keeping shelf compliant to planogram	- know expiry dates - counts of FIFO	- Product feedback - Price feedback	-Type of consumer - Product size - Planogram style
Pain Points	- Time consuming - No available reliable software	- 20-30% of merchandiser time consumed - Non homogenous shelves	- Manual/time consuming work - no real time available info about stock - no track of damages and missed expiry dates	- available data is manually collected - collected data using simple reports and WhatsApp groups	- No consumer data - No available data for planogram style

Figure 2.2: Execution Phase

DATA COLLECTION PHASE:



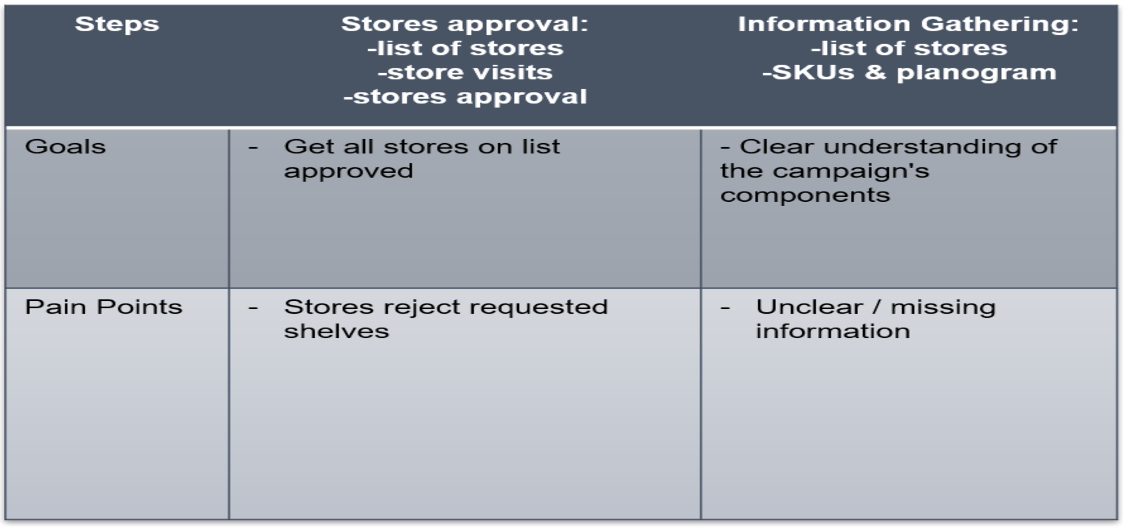
Steps	Go to warehouse: -get products -put in CashVan	Decide on route	Visit store: -contact person -identify shelf location	Merchandise: -planogram -enough products to fill stand	POD (Proof of Delivery): -signature of store owner -photo of stand with planogram	Back to warehouse
Goals	- Get all stores on list approved	- Clear understanding of the campaign's components	Collect products	Optimize route for fastest delivery	Get POD & get photo approval	Check out of his day
Pain Points	- Stores reject requested shelves	- Unclear / missing information	- Out of stock items - Not enough space in Cashvan	- Lack of GPS tools - Traffic	- Supervisor not available for prompt response - Contact person unavailable	- Missing data to submit

Steps	Product data collection	Shopper data collection	shelf and planogram data collection
Goals	- Collect Product feedback - Collect Price feedback	- Know consumer demographics - Know consumer behavior	- item count - price changes - stand feedback (ROI by outlet)
Pain Points	- available data is manually collected - collected data using simple reports and WhatsApp groups (good data collection mechanism, inefficient data storing)	- no available data - no real time on going data collection - all collected data are from merchandiser data from being in stores	- available data only from cashier sales - no live item count detection - cant changes prices remotely from one place

Figure 2.3: Data Collection Phase

2.2.2 Merchandiser journey:

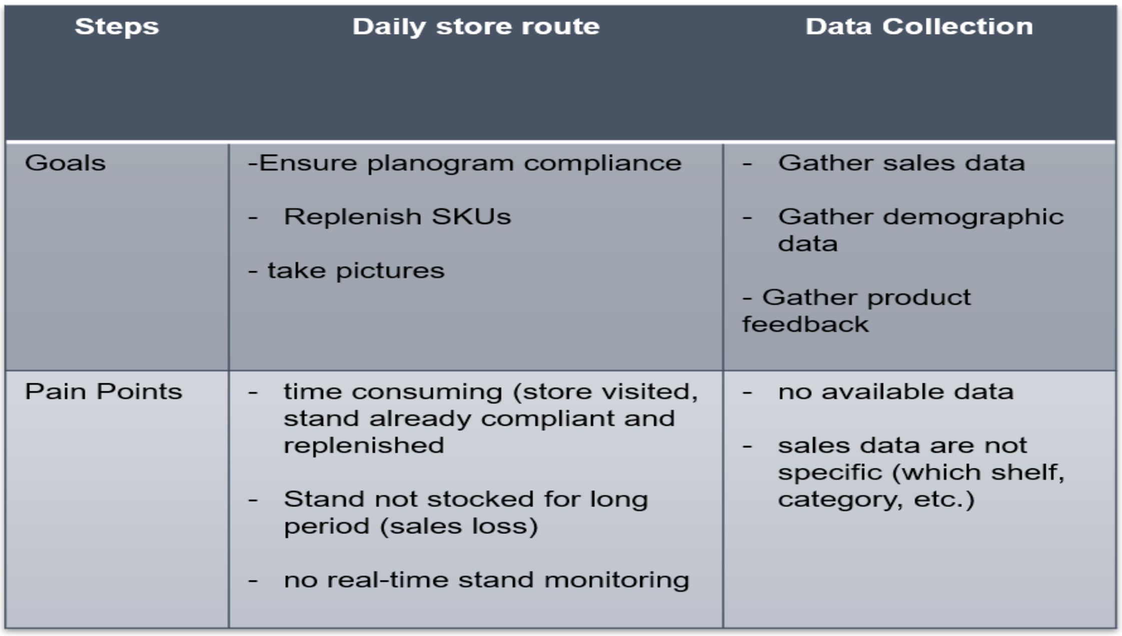
PREPARATION PHASE:



Steps	Stores approval: -list of stores -store visits -stores approval	Information Gathering: -list of stores -SKUs & planogram
Goals	- Get all stores on list approved	- Clear understanding of the campaign's components
Pain Points	- Stores reject requested shelves	- Unclear / missing information

Figure 2.4: Preparation Phase

DAILY ROUTINE:



Steps	Daily store route	Data Collection
Goals	-Ensure planogram compliance - Replenish SKUs - take pictures	- Gather sales data - Gather demographic data - Gather product feedback
Pain Points	- time consuming (store visited, stand already compliant and replenished - Stand not stocked for long period (sales loss) - no real-time stand monitoring	- no available data - sales data are not specific (which shelf, category, etc.)

Figure 2.5: Daily Routine

Extracting the problems:

During the brainstorming session, we translated every pain and every goal into problems, then after filtering these problems, we obtained this list of customer needs:

Product data collection:

- what SKUs are on the stand.

- how many SKUs are on the stand.

- which SKUs picked from the stand.

-heat map of the shelves.

Shoppers's data collection:

-Demographics of the shoppers.

-shelf scanning flow of the shoppers.

-how many shoppers passed by the stand.

-dwell time in front of the shelves.

-dwell time in the aisle.

Visual features:

-change visual to entice shoppers.

-change visual to represent picked products.

2.3 Generating the project statement

At the end of the brainstorming session, we formulated our problem statement as following:

Create an in-store on shelf Data collection system to be used by the Brands to collect shopper behavior Analytics based on predefined planogram. The unit needs to provide dynamic price changes and targeted communication based on individual shoppers profiling on behavior.

C. CHAPTER 3:

Design of Experiments

In this chapter, we will introduce our experimental designs for every aspect in our project. These designs were made before starting the implementation, after finishing testing all the experiments, we will have a final list of components which will be specified and defined in the following chapters.

These are the technologies generates as solutions for every problem/pain:

Problems	Proposed Solution
Count Of Placed Products	(RGB-D Camera / Weight Sensors / Distance Sensors) + Web-interface
Out Of Stock	(RGB-D Camera / Weight Sensors / Distance Sensors) + Web-Interface
List Of SKUs	RGB Camera / Distance Sensors
Consumer Demographics	AI + RGB Camera + Microcontroller
Change Prices Remotely	Display (LCD Screens) + web interface
Enticing Consumers using collected data	Display (LCD Screens) + web interface
Heat Map	RGB Camera + Software processing
Know dwell time in front of shelf	(RGB camera / Weight Sensor) + Microcontroller
Change Visuals to entice shoppers	Display (LCD Screens) + Acoustics (sounds)
Know What SKUs are on the stand	(Distance Sensors/RGB) + Microcontroller
Change visuals to represent picked product	Distance Sensors/RGB/Weight Sensors

Types of sensors/components	
RGB-D Camera	Kinect Camera / Real sense depth
Load Cells/Weight Sensors	
RGB Camera	
Distance Sensors	Ultrasonic / Laser LIDAR / digital IR
Displays	LCD Color / LED matrix / LCD Monochrome
Microcontroller	Arduino / Raspberry pi

3.1 Product:

In this section, we need to study and to solve several problems related to the product. First, the take-off count which helps us know the number of products picked up by the customer. Second, the out of stock, which alert the merchandizers in case the stand ran of products, and the count of products. We generated and designed several solutions for these problems as follows:

i. Distance sensor and Pusher:

The distance sensor will be placed behind the products, and whenever a product is picked, the pusher will push the products, which will increase the distance between the distance sensor and the last product. Below you can find a link that shows how does a pusher works: https://youtu.be/9vN3uU7Ukag (Figure 3.10,3.11,3.12)

Every time a product is picked, the ultrasonic sensor will read a higher distance.

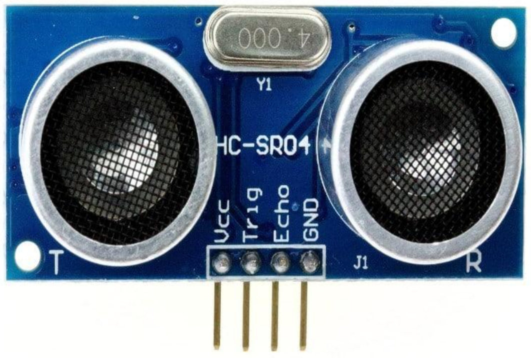


Figure 3.11: Ultrasonic distance sensor

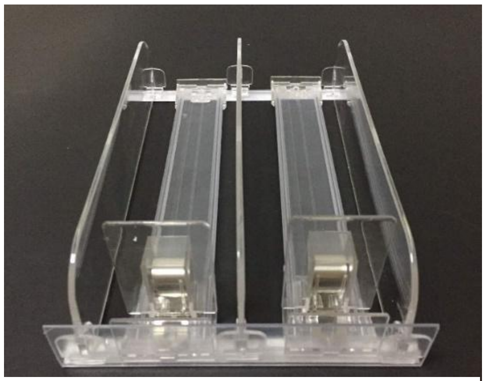


Figure 3.10: Pusher

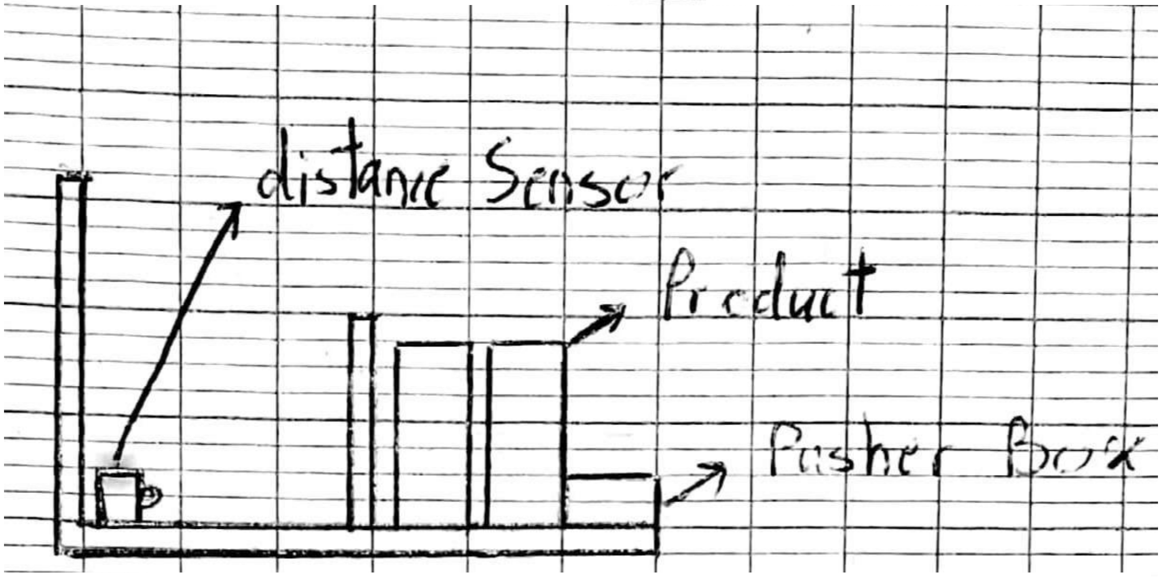


Figure 3.12: Side View

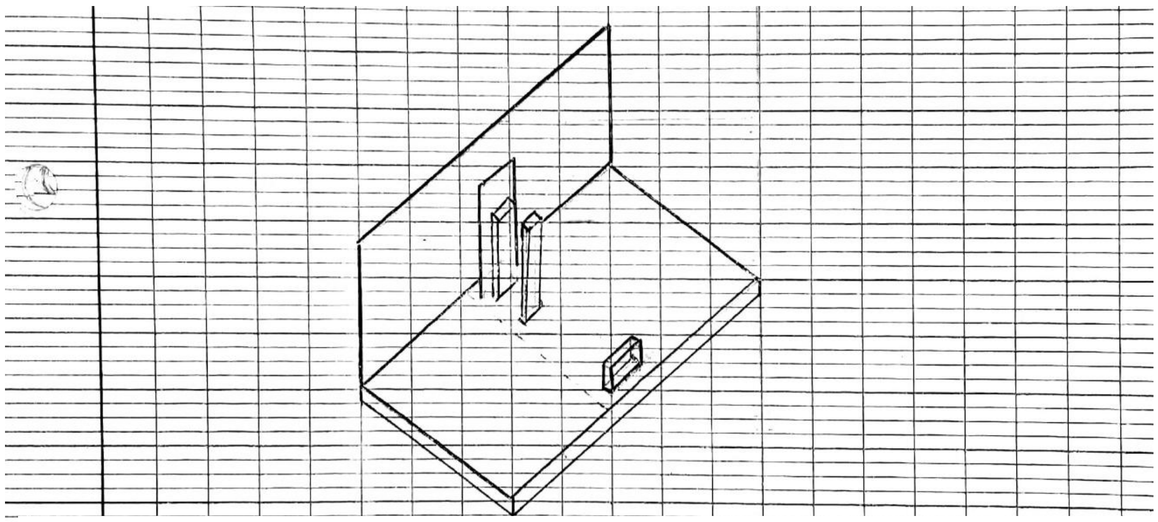


Figure 3.13: Isometric view

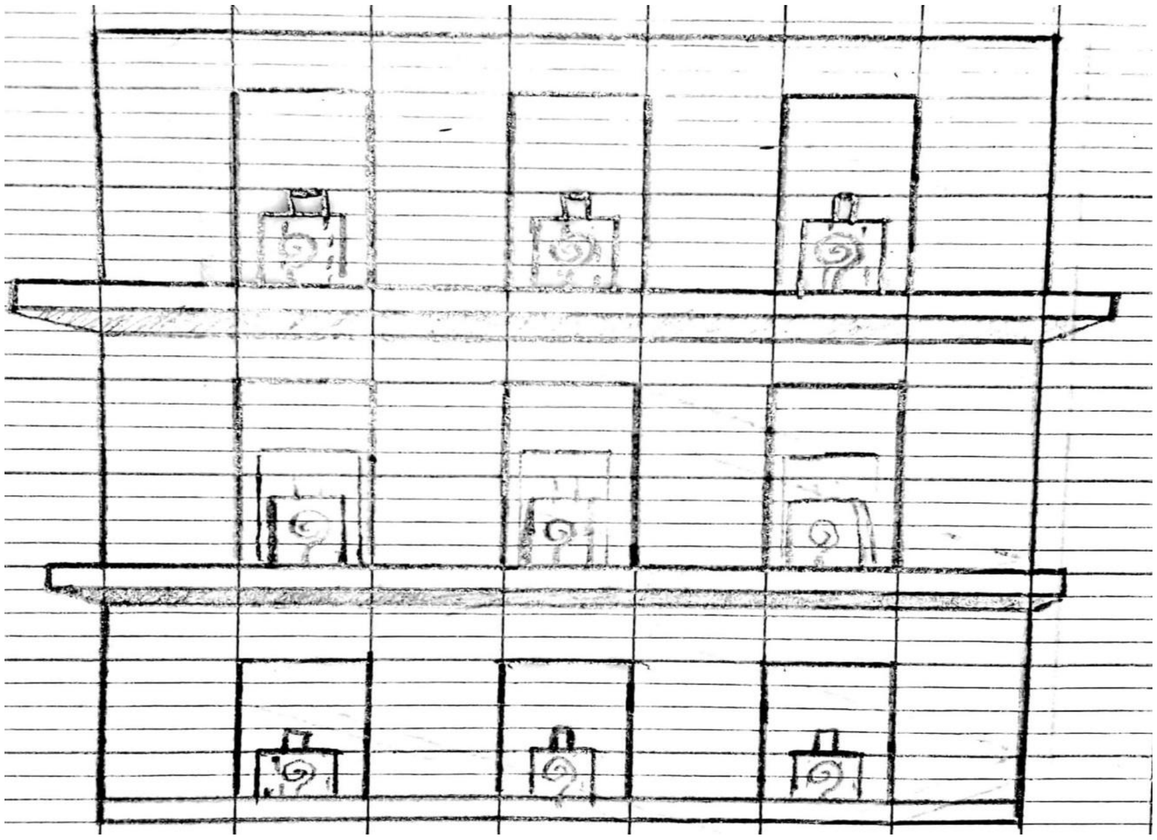


Figure 3.14: Front View

ii. Weight Sensor / Pressure Pad:

Before and after the taking off the products, the weight of the stand will decrease. This weight sensor will be recognized by the weight sensor/Pressure Pad. (Figure 3.153)

In This application, we have two options for the weight sensor. The first option is the half bridge weight sensor (Figure 3.151) and the full bridge weight sensor (Figure 3.152).

Figure 3.151: Half Bridge Weight Sensor

Figure 3.152: Full Bridge Weight Sensor

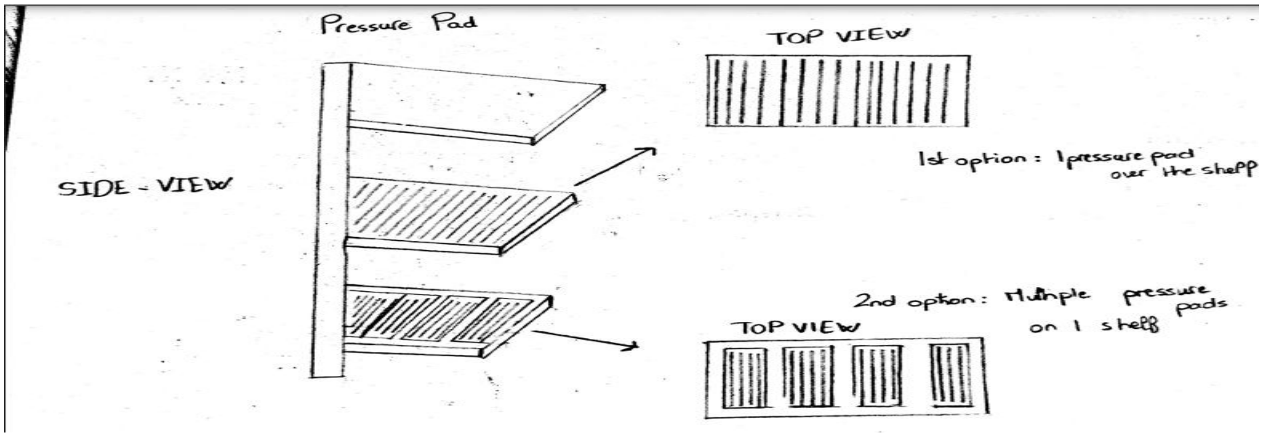

Pressure Pad
TOP VIEW
SIDE - VIEW
1st option : 1 pressure pad over the shelf
2nd option : Multiple pressure pads on 1 shelf
TOP VIEW

Figure 3.153: Side and Top View

iii. LDR (Light Dependent Resistor):

The LDRs are light sensitive devices most often used to indicate the presence or absence of light, or to measure the light intensity. Under each product, a LDR will be inserted through a hole. When the product is in its place, the LDR sees darkness, and when the product is picked up by the customer, the LDR sees light. (Figure 3.16)

In other words, each time the LDR sees light, it means that there is no product on the hole, because it is picked by the customer. The

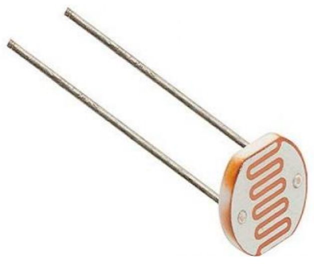


Figure 3.161: LDR

number of LDR seeing darkness is the number of products picked up by the customers. Thus, collecting this type of information help us to know the count of products.

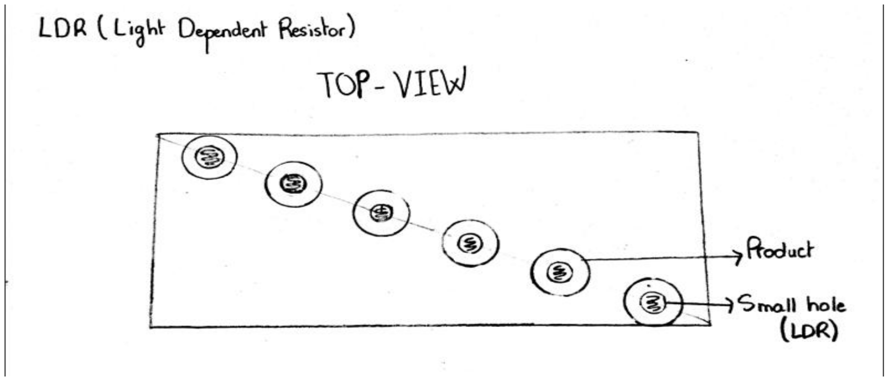


Figure 3.162: Top View

iv. 2D Scanner:

The 2D scanner will be implemented above the targeted shelf, which means that it will be under the upper shelf as shown in the following figures. The 2D scanner should give a XY coordinates, specifying the position and coordinates of every product on the stand. (Figure 3.17 & 3.18).

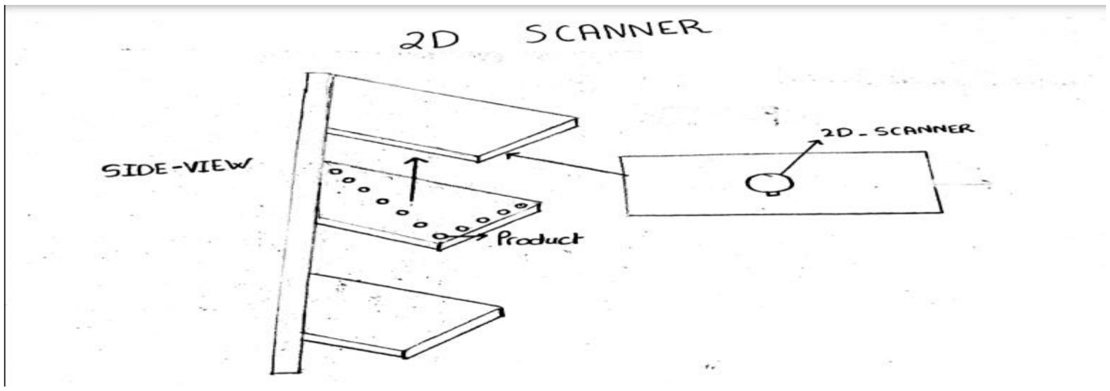


Figure 3.17: The view from the 2D scanner

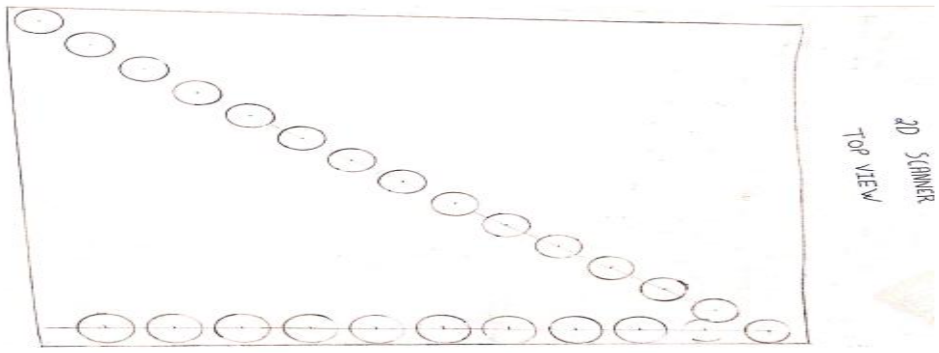


Figure 3.18: Side View

3.2 Communication:

In this section, we need to solve several problems related to the communication between the customer and the stands. To achieve our business goals, we need to ensure enticing customers based on the collected data, and to change the prices remotely, so it would be easier and better economically.

i. Enticing customers based on collected data:

We can add screens on the shelf, header, and sides to present some advertisement videos about the products, and a full detailed informative page for each product present on the corresponding stand. In addition, we can add microphone, speaker, and touch screen to go to another level in enticing customers.

ii. Change Prices Remotely:

Changing the paper labels daily by the merchandisers cost a lot, therefore we thought of a long term, eco-friendly and easy solution for this problem. We can use screens, digital price labels and QR codes to solve this problem as shown in the figures below (Figures 3.21 & 3.22).

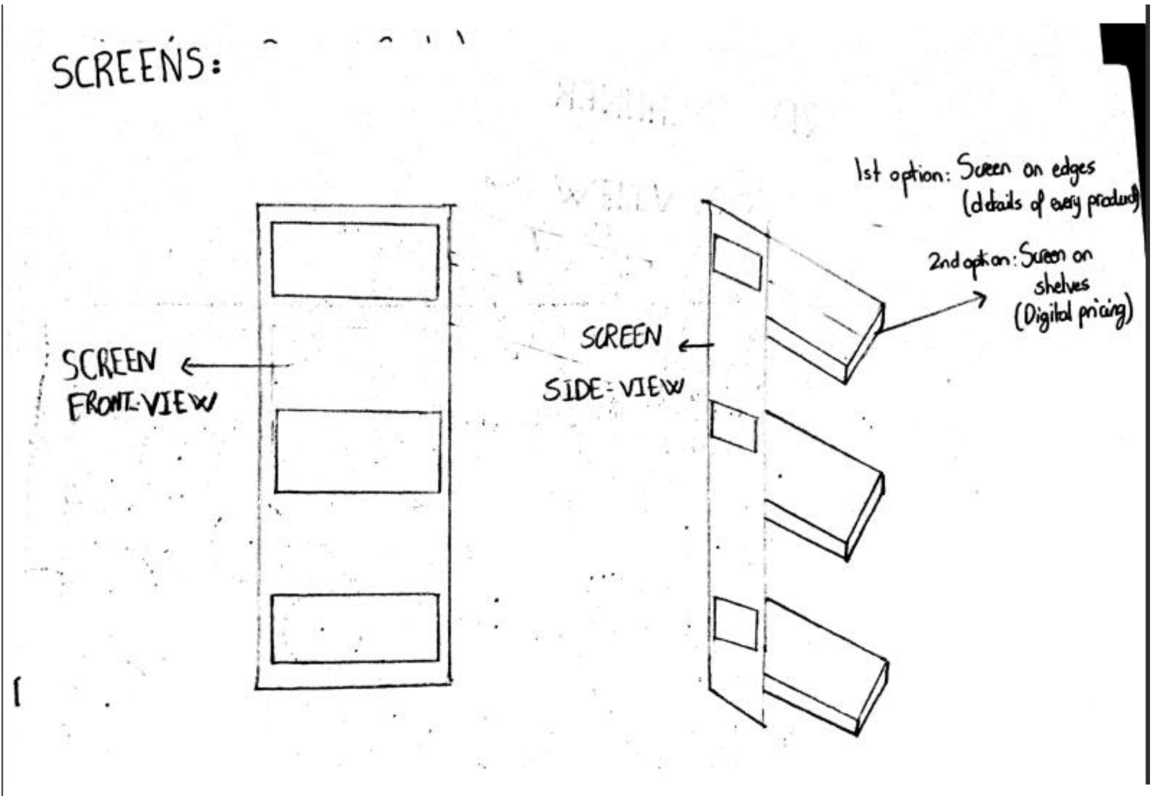

SCREENS:
SCREEN FRONT-VIEW
SCREEN SIDE-VIEW
1st option: Screen on edges (details of every product)
2nd option: Screen on shelves (Digital pricing)

Figure 3.21: Screens Side and Front view

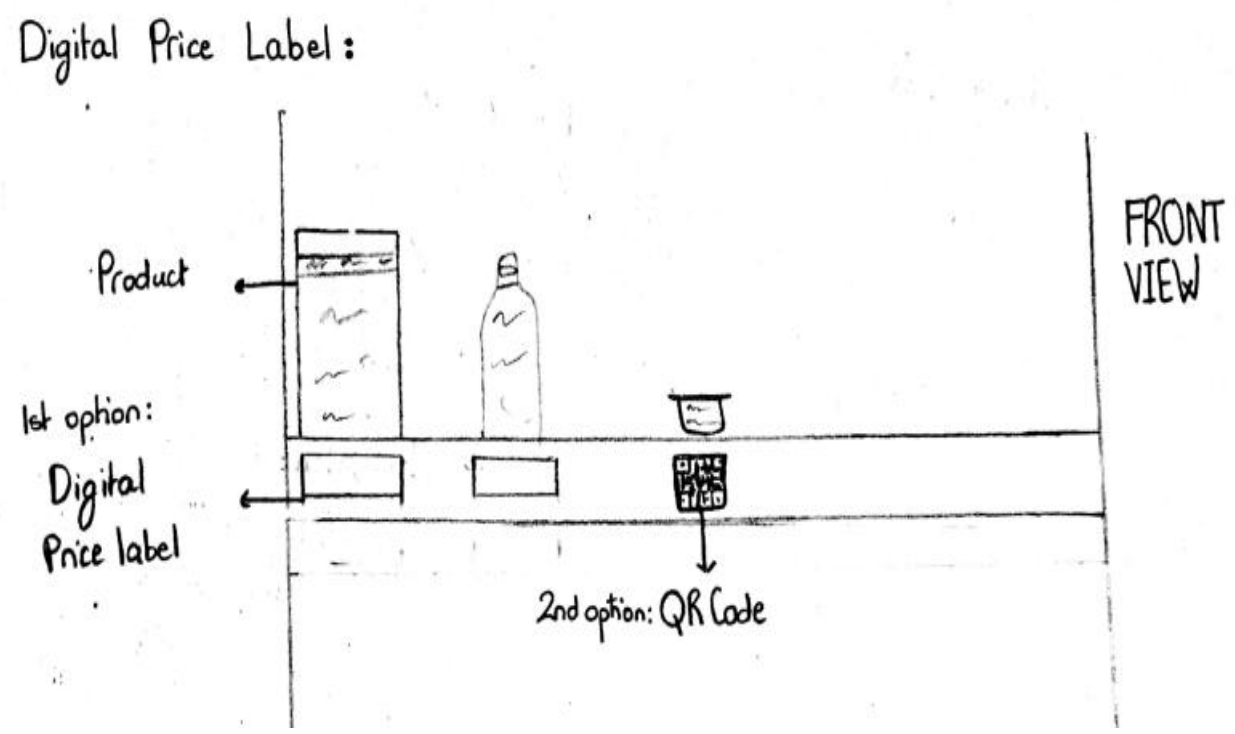


Figure 3.22: Digital Price Label Front view

3.3 Planogram:

The planogram schematic drawing or plan for displaying merchandise so as to maximize sales. In this section, our main goal is to find the best approach regarding the stock planning and visual merchandizing to increase sales, improve inventory control and prevent of out of stock. Moreover, and to reduce the cost, the planogram aims to improve the overall inventory management.

Hence, we will be focusing on two main points:

i. **Eye Tracking:**

In which we will be using a RGB camera implanted on the shelf that focuses on the costumers' tracking their eye dynamics to find the most approached positions. In the figure below (figure 3.31), a block is found in front of the shelf representing a tall human being (210cm) in which the RGB camera is able to scan this full body and focus on the facial expressions. Hence, the RGB camera using camera vision will be able to detect the movement of the person's eyes hence tracking the most approached brands. As a result, we would be able to have a better planogram resulting in increasing the brands' sales. Sketch of the eye tracking technique is shown the figure below (figure 3.31)

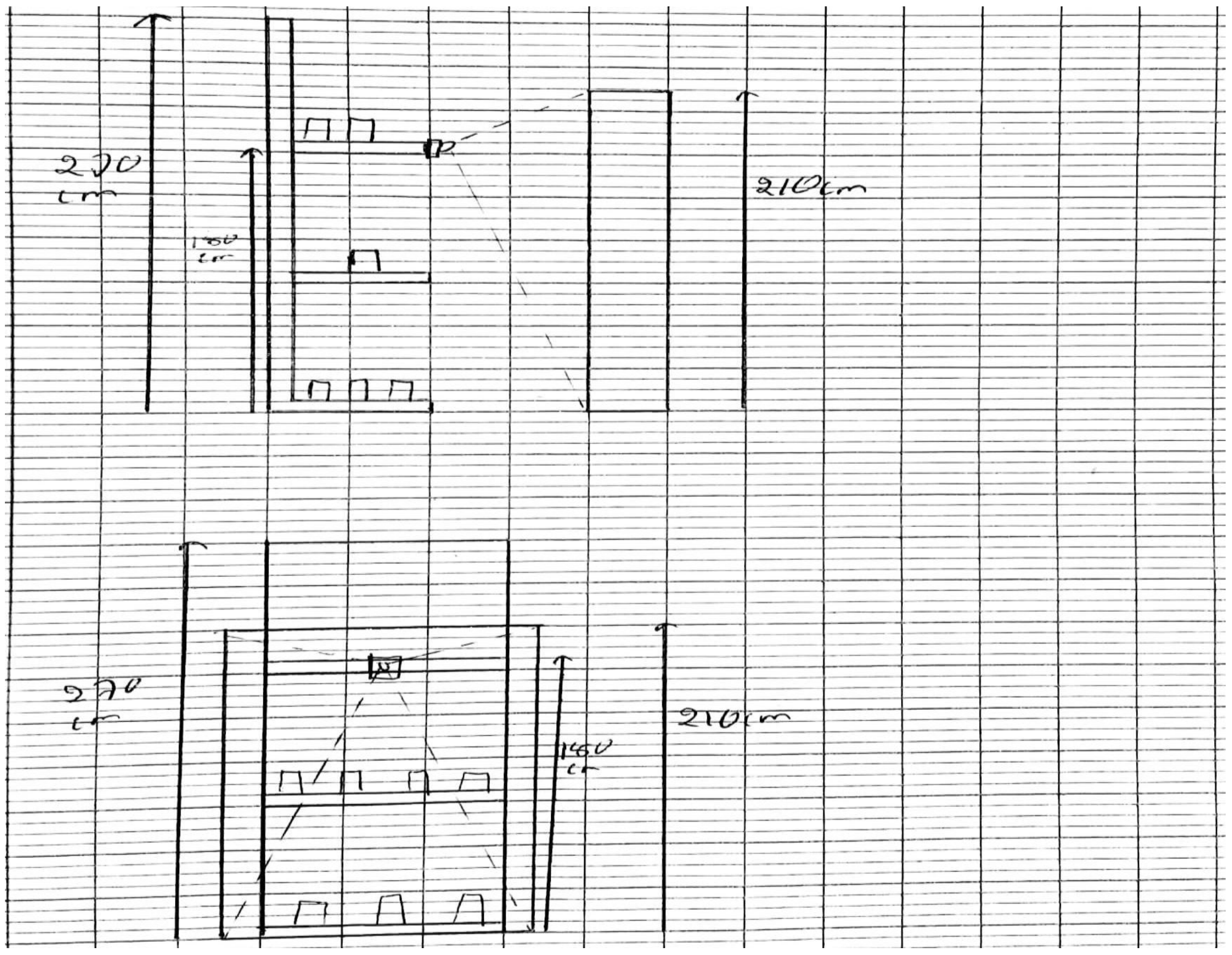


Figure 3.3: Eye Tracking Side and Front View

ii. Proof of Planogram Compliance:

To ensure the proper positioning and quantity found of brands. Furthermore, supervising the location of distributed brands with proper merchandizing hence going accordingly with the planogram.

3.4 Costumer:

In the following section, we will be focusing on analyzing everything about the costumers' information among the isle in which our smart shelf is located. Hence, we will be implementing engineering techniques using sensors, cameras, microcontrollers, and software's to study the behavior of all costumers' passing by the smart shelf and the whole isle and analyze the way those parameters are reflected on sales. We will be focusing on four main points:

1. **Demographics:**

We will be using a RGB camera with computer vision and algorithms to detect the costumers' facial expressions to study their emotions while standing in front of the smart shelf (Happy, Sad, Angry...), in which the camera found is capable of integrating the area of the person standing in front of the shelf. The sketch is as shown in the following figure (figure 3.41):

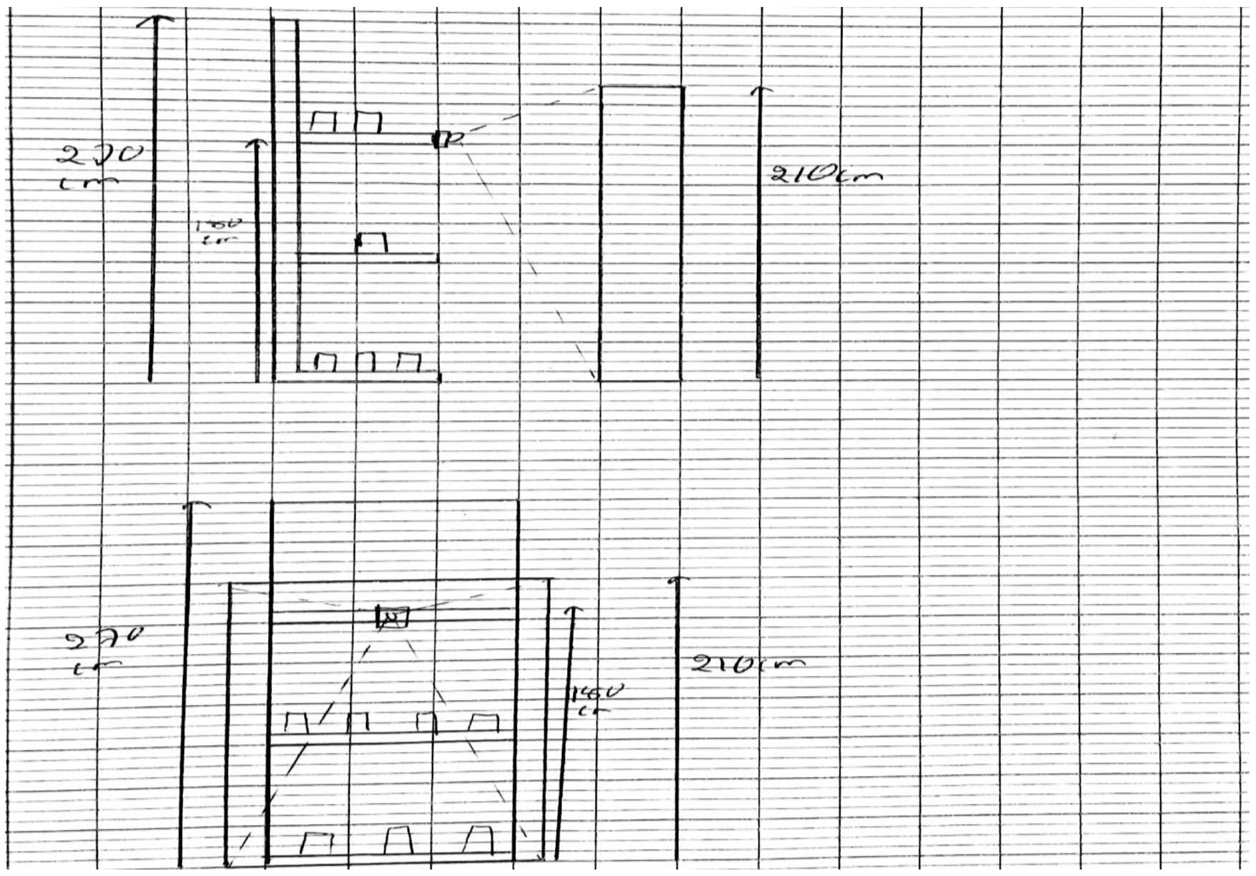


Figure 3.41: Demographics study inside and front view.

2. Dwell Time:

Having a big number of costumers standing in front of our smart shelf will logically lead to increase in sales. Hence, we will be using various techniques to study the dwell time among the shelf's range such as ultrasonic sensor, pressure sensor, proximity sensor, 2d-3d lidars and RGB camera. Distance sensors will be implemented on the shelf in which they will be studying the distance difference over a defined range of time which will be representing the presence of costumers. Moreover, pressure sensors will be implemented on the ground around the limit distance range of the shelf which will be calculating the different weights over a defined time range. The following figure shows the sketch approach of the implemented sensors on and in front of the smart shelf (figure 3.42).

3. Foot Fall:

Like the dwell time, the foot fall is the study of costumers' presence but among the whole isle where our shelf is located. Ultrasonic sensors and 2d LIDARS are to be used studying the distance differences in order to analyze the amount of traffic (costumers) passing by the isle where our smart stand is located. Hence, finding the foot fall aims to have the best location of our smart shelf increasing brands' sales. The following figure shows the sketch approach of the implemented sensors on the smart shelf (figure 3.42).

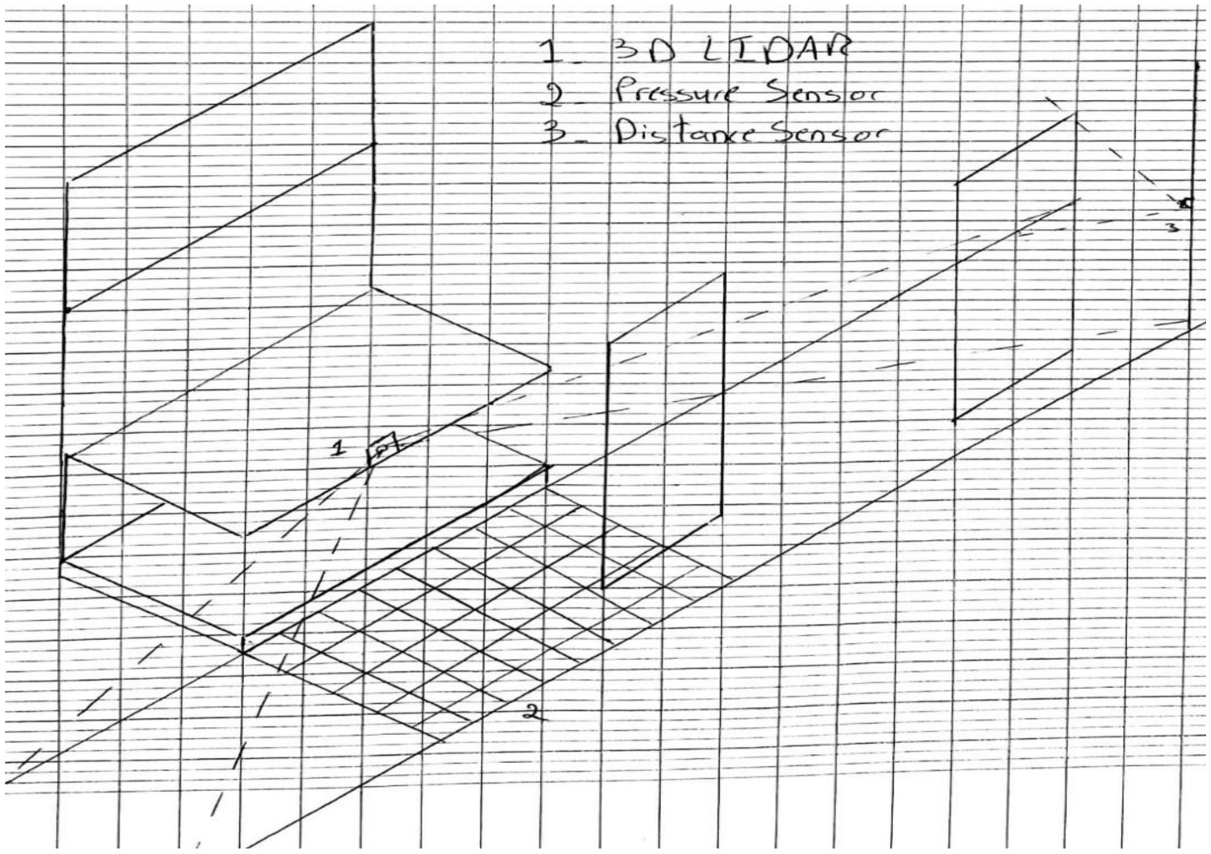


Figure 3.42: Dwell time and Foot Fall Isometric View.

4. Heat Map:

A 2d LIDAR implemented on the smart shelf front side will be used to generate a heat map using engineering software's (ex: MATLAB). This map will be used to study the most approached positions on the smart shelf to find the most needed brands. Moreover, this helps improving the planogram and merchandizing supervision hence we can increase sales and have a better management for the whole goods found on the shelf. The following figure shows a sketch for the sensor implementation and heat map generation (figure 3.43).

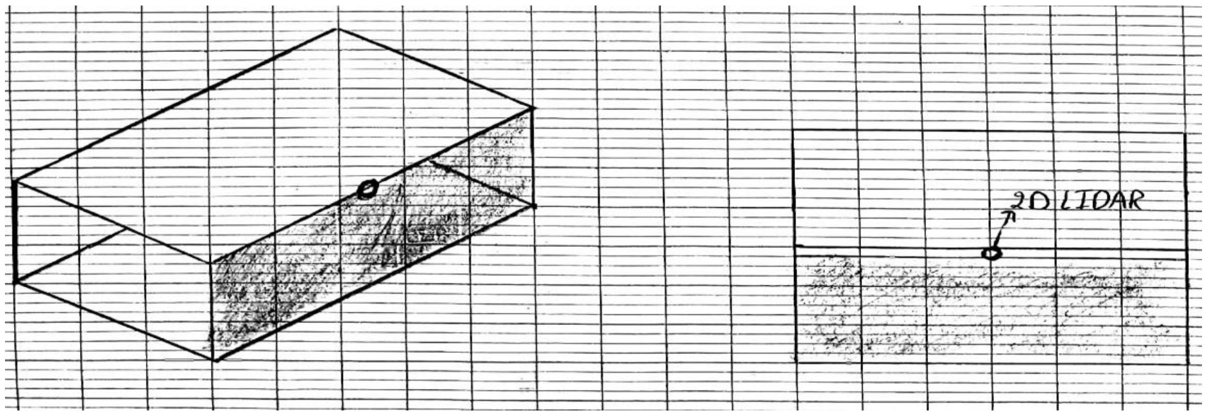


Figure 3.43: Heat map generation using 2d LIDAR Isometric and Front Views.

D. CHAPTER 4:

DESIGN OF EXPERIMENTS RESULTS AND DISCUSSION

4.1 Product Count Using Sharp IR

[1]Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

ABSTRACT

The objective of this experiment is to find the quantity of products found on the retail pusher. We should design and implement a circuit using an Arduino Uno, Sharp IR sensor (0A41SK0F Model) and a product pusher. As a conclusion, screen counter decrements as products taken off the pusher and vice versa.

Keywords: Arduino Uno, Sharp IR Sensor, Display Screen.

1. INTRODUCTION

The main objective behind this experiment is to find the quantity of products found on the retail pusher. First, we fix the Sharp IR sensor to set an initial value of distance to the back end of the retail pusher. As the customer takes products off the pusher, the measured distance increases hence indicating the decrease in products' quantity. On the other hand, and in case the customer put any taken product back on the retail pusher, the measured distance decreases indicating the increase in products' quantity. Flowchart is shown in figure 4.11.

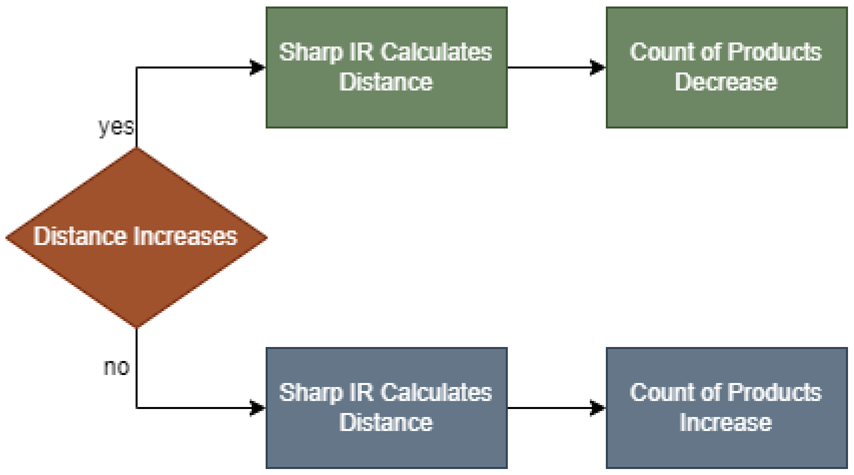

Sharp IR Calculates Distance
Count of Products Decrease
yes
Distance Increases
no
Sharp IR Calculates Distance
Count of Products Increase

FIGURE *4.11*: SHARP IR SENSOR BLOCK DIAGRAM

\

2. MATERIALS AND METHODS

1.1 Components

- Sharp 0A41SK0F IR Sensor:

Sharp A41SK0F IR Sensor is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IR-LED (infrared emitting diode) and signal processing circuit [5]

FIGURE *4.12*: SHARP A41SK04F IR SENSOR

- Product Pusher:

By using this product pusher (figure 4.13), the packaged goods that you store on your shelves will be automatically pushed to the front each time a customer retrieves one. When your packaged goods are constantly pushed to the front, customers can easily identify the different products and conveniently access them from the front. [6]

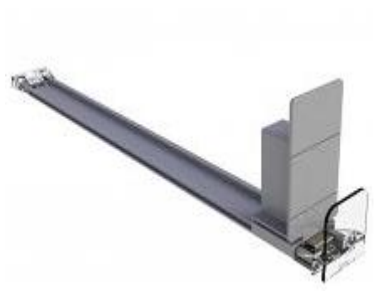


FIGURE 4.13: Product Pusher

1.2 Methodology

- Distance Calculation:

The distance read by the Sharp IR sensor is calculated using the input voltage of the sensor into the Arduino. Moreover, multiplying this voltage by (5/1024) in order to map the input voltage between 0 and 5V into integer values between 0 and 1023 (10 bit). Finally, raising the calculated voltage to the exponent of -1 and multiplying it by 13 in order to get the final distance as worked out from the sensor's datasheet.

$$Voltage\ (V) = Input \times ({}^{5}/_{1024})$$

Equation 1: Voltage Equation in (V)

$$\boldsymbol{Distance\ (cm) = 13 \times Voltage^{-1}}$$

Equation 2: Distance Equation in (cm)

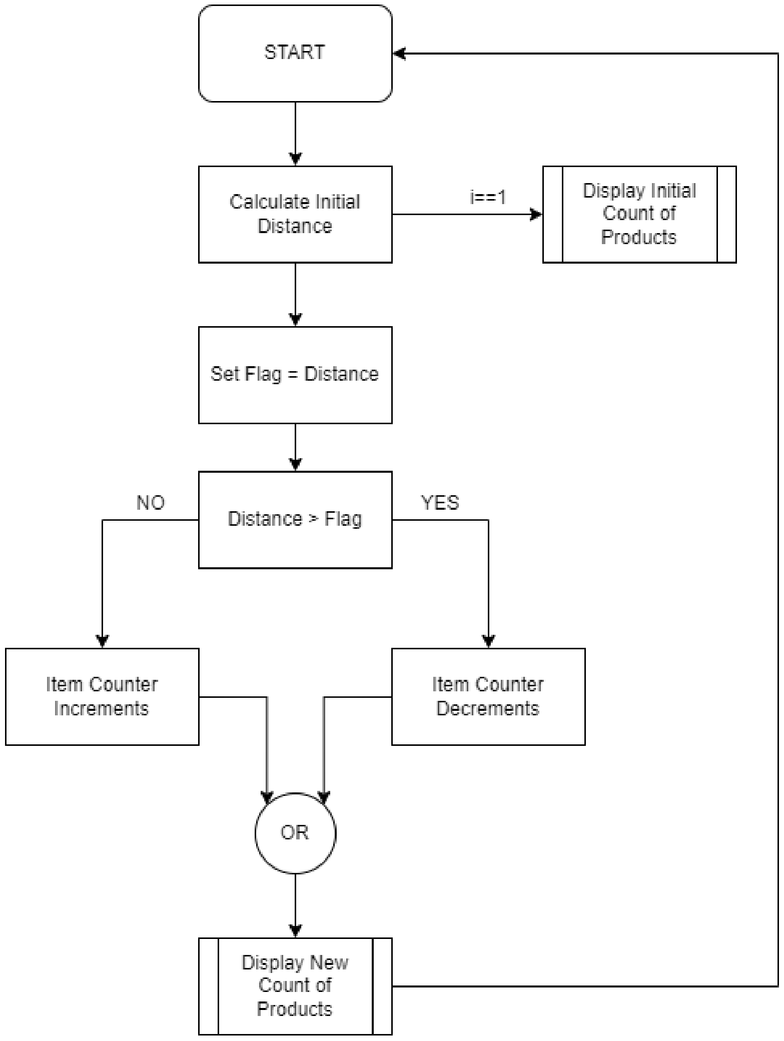


FIGURE 4.14: SHARP IR SENSOR FLOW CHART

3. RESULTS AND DISCUSSION

We first connect the SHARP IR Sensor to the Arduino Uno, in which we connected the sensor's VCC to 5V, V_{ref} to ground and the signal wire to pin A0. Moreover, we connected the LCD screen to the Arduino in order to display the measured distance.

Second, we fixed the IR Sensor to the retail pusher hence as the custom purchase any of the products found, the distance measured increases hence indicating the decrement in the count of products. On the other hand, in case the customer adds a product on the pusher, the distance measured by the sensor decreases hence indicating the increment by the count of products.

Third, if the customer adds a product that wasn’t initially found on the pusher, in which the product added has different dimensions from the ones already found. The new distance measured won’t be compatible with the previous one hence the LCD screen won’t increment the count of product and displays that a new type of products was added to the retail pusher.

Problems faced:

As the customer adds a product with the same or a duplicate dimension as the initial ones found on the pusher, the count increment by 1 or the number of dimension duplication. Hence, the problems faced are:

- Items with the same dimensions cannot be differentiated.
- Items with duplicate dimensions cannot be differentiated.

Proposed Solutions:

In order to better implementation of the experiment and achieve better results, several solutions could be proposed:

- Integrating the IR Sharp sensor with another sensor (Ex: Weight Sensor)
- Using mm accuracy sensor.
- Stop the counter from adding duplicates.
- Changing the planogram.

The circuit:

Below is the implemented circuit using Arduino UNO, Sharp IR Sensor, Product Pusher and jumper wires (figure4.15).

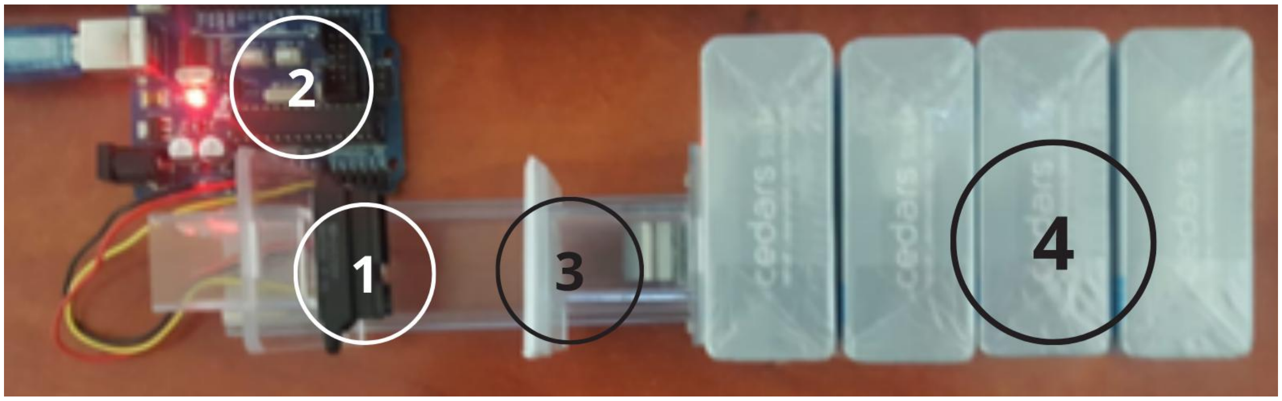


FIGURE 4.15: The Implemented Circuit

COMPONENT’S NUMBER	**COMPONENT’S NAME**

1	SHARP IR SENSOR (0A41SK0F MODEL)
2	ARDUINO UNO
3	RETAIL PUSHER
4	PRODUCTS

TABLE 1: CIRCUIT COMPONENT'S NAME AS SHOWN IN FIGURE 9

4. Prototypes

Pusher CAD Model #1:

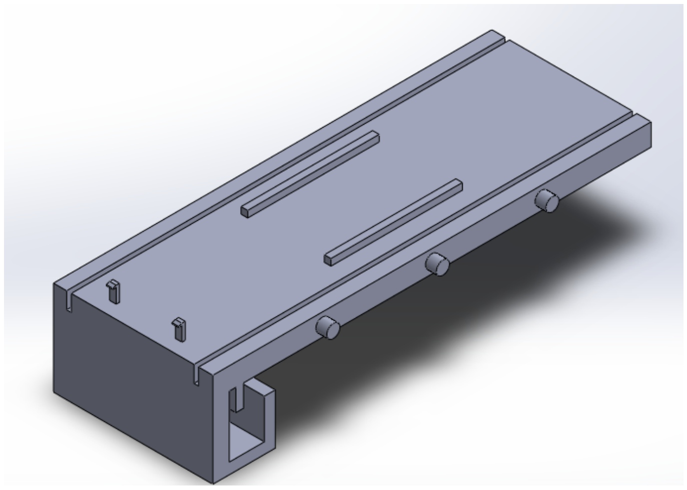


FIGURE 4.16: First Part Implemented on Shelf

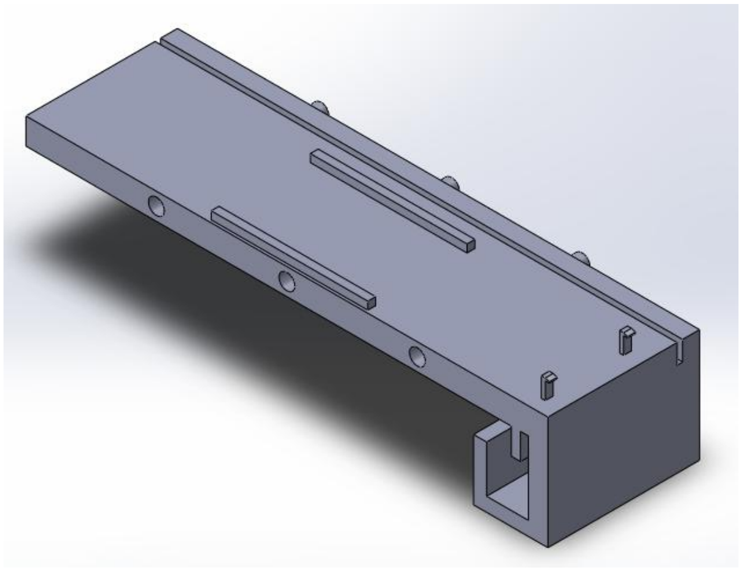


FIGURE 4.17: Completion of First Base Part

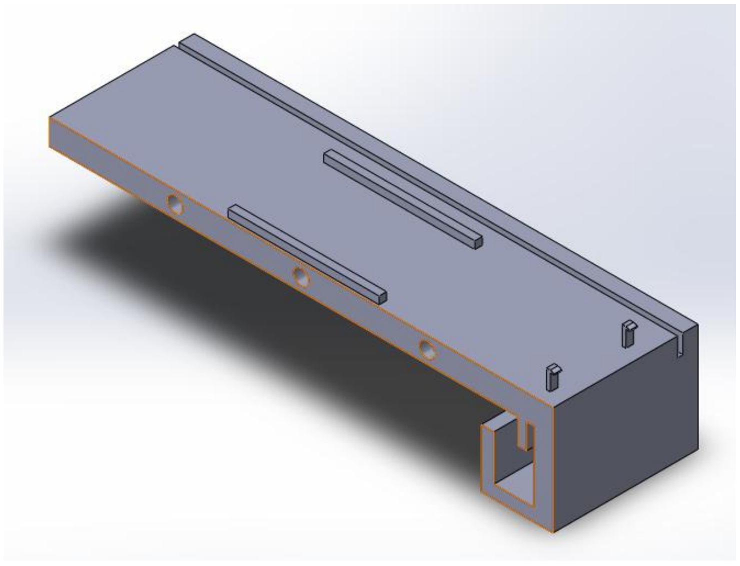


FIGURE 4.18: Last Base Part to be implemented on Shelf

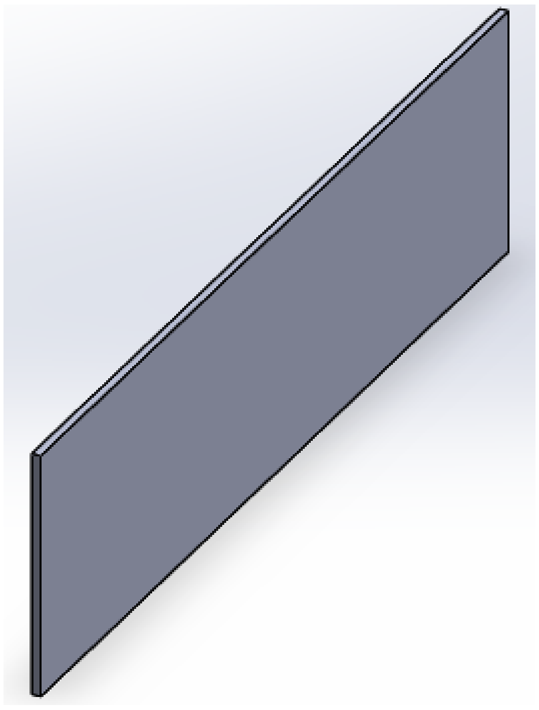


FIGURE 4.19: Rails Implemented on Base Parts

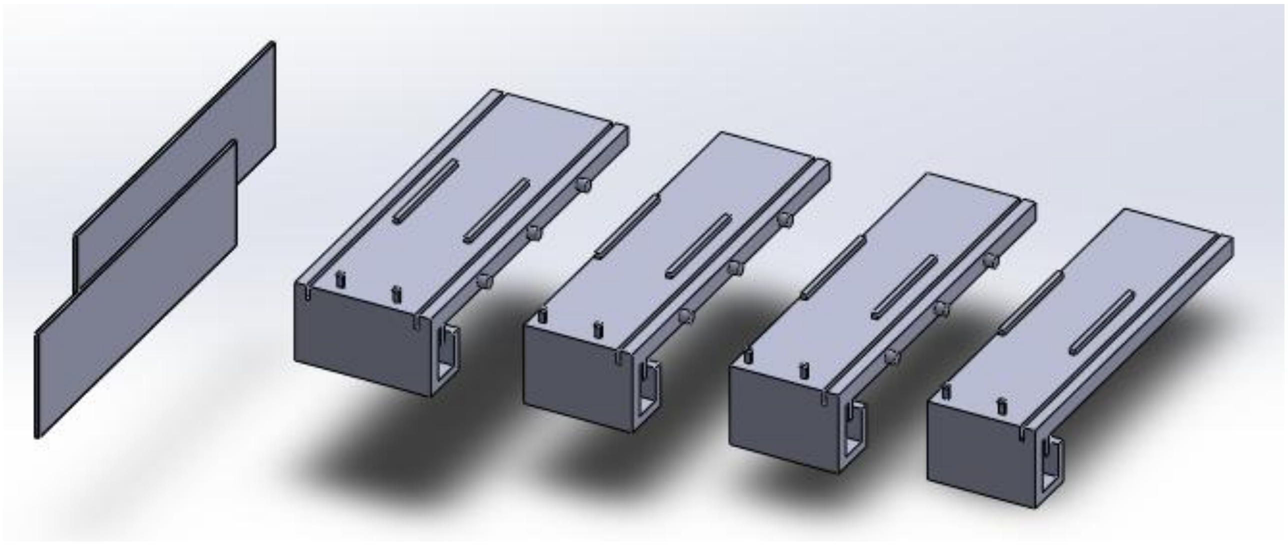


FIGURE 4.2: Assembly of The CAD Model

Although this CAD model had very good dimension precision to be first implemented on the stand and second to hold the retail pusher, some limitations are to be taken into consideration regarding this CAD prototype in which the models have big dimensions and need a lot of time to be 3D printed.

Pusher CAD Model #2

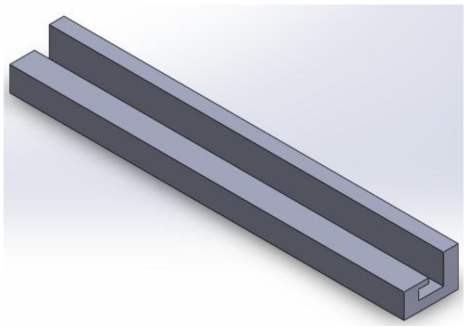


FIGURE 4.21: Base

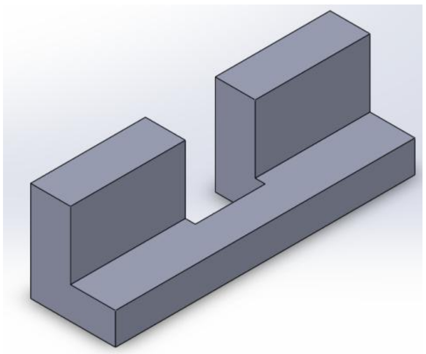


FIGURE 4.22: Rail Base

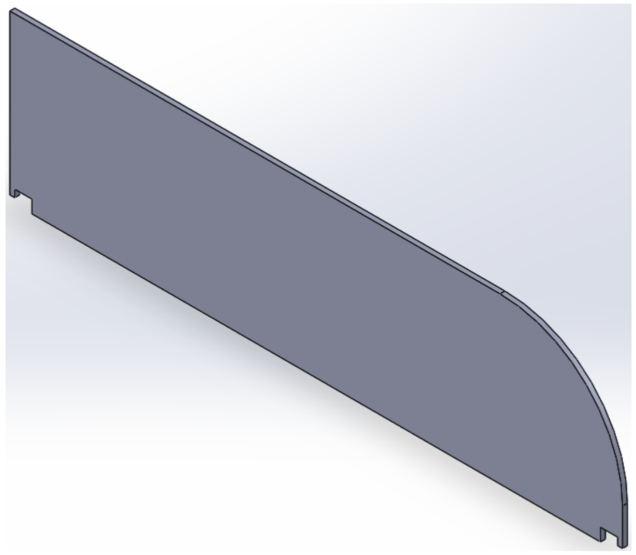


FIGURE 4.23: Rails

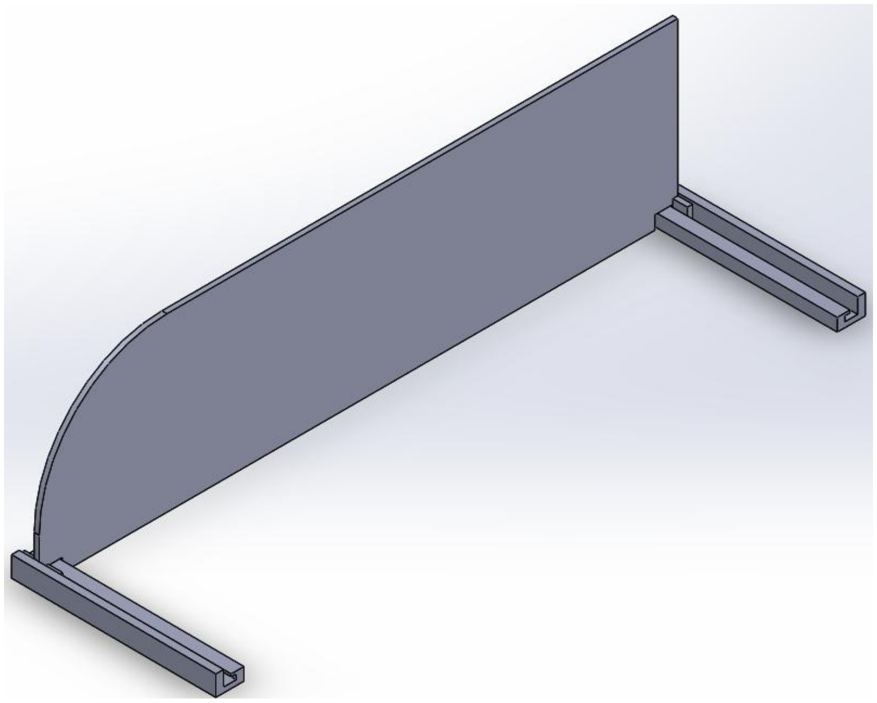


FIGURE 4.24: 2^{ND} Prototype Assembly of the CAD Model

5. IMPLEMENTATION And RESULTS

We implemented three Sharp IR sensors with retail pushers, and as the code runs we randomly added and took cigarette packs on and off the pusher in order to observe the variation in the count of products' number found on a certain pusher and detected by the sensor. Moreover, the code calculates the total number of products found with every random selection done by the customer, hence giving a precise study about the count of products.

The following figures (425 and 426) show the results of the experiment and count of products on every retail pusher:

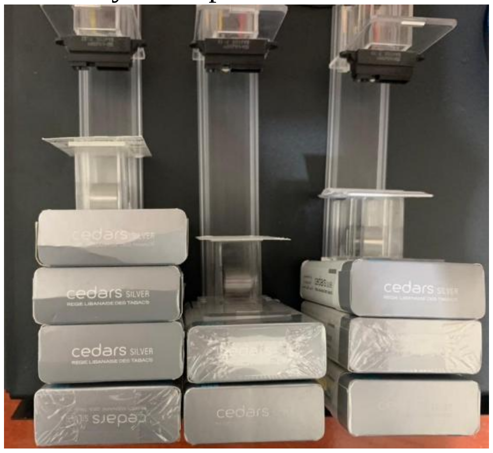
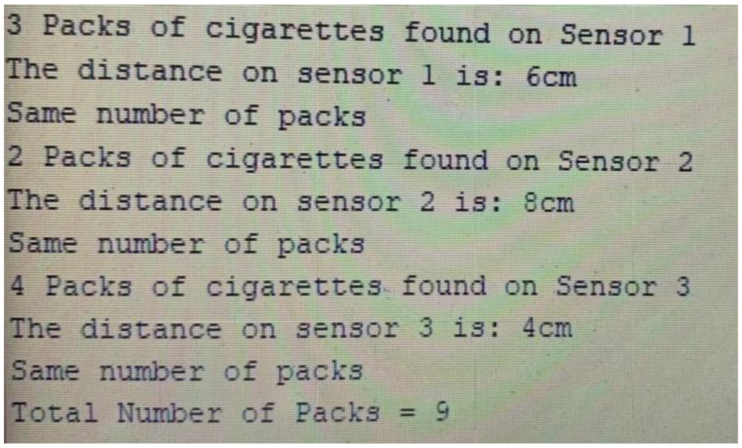


FIGURE 4.251: Products on Shelf



```
3 Packs of cigarettes found on Sensor 1
The distance on sensor 1 is: 6cm
Same number of packs
2 Packs of cigarettes found on Sensor 2
The distance on sensor 2 is: 8cm
Same number of packs
4 Packs of cigarettes found on Sensor 3
The distance on sensor 3 is: 4cm
Same number of packs
Total Number of Packs = 9
```

FIGURE 4.252: Calculated Number of Products on Every Sensor

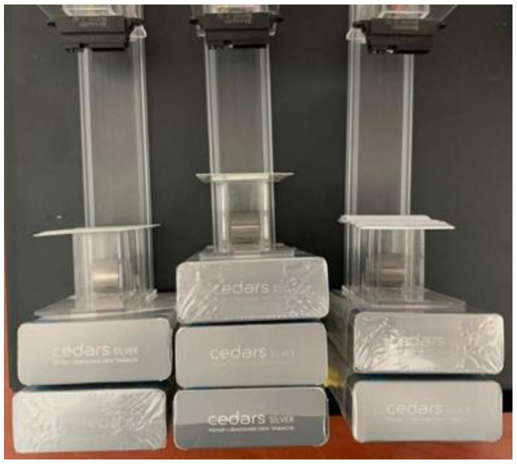
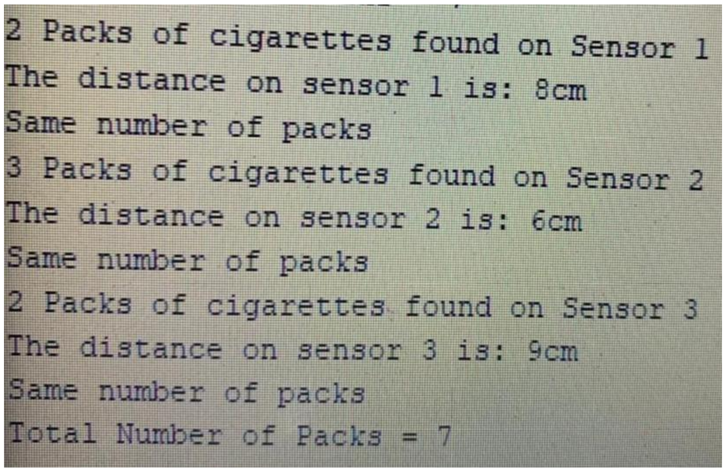


FIGURE 4.261: Products on Shelf



```
2 Packs of cigarettes found on Sensor 1
The distance on sensor 1 is: 8cm
Same number of packs
3 Packs of cigarettes found on Sensor 2
The distance on sensor 2 is: 6cm
Same number of packs
2 Packs of cigarettes found on Sensor 3
The distance on sensor 3 is: 9cm
Same number of packs
Total Number of Packs = 7
```

FIGURE 4.262: Calculated number of Products on Every Sensor

6. CONCLUSION

As stated before, the main objective behind the above experiment is to find the quantity of products placed on the retail pusher as a function of distance read by the distance sensor. As customers take-off products, distance sensor reads a distance greater than the initial one hence the counter displayed on the screen decrements showing the new number of products found on the pusher. On the other hand, the counter increments and the screen also display the new number of products found.

7. Appendix 1:

```
#define sensor1 A0
#define sensor2 A4
#define sensor3 A2 // Sharp IR GP2Y0A41SK0F (4-30cm, analog)
  bool i = true;
  double dp=2.2,dc=3.5, pack_dim1, pack_dim2, pack_dim3;
  int d1=16, count1, count2, count3, cig1, cig2, cig3, total;
void setup() {
  Serial.begin(9600); // start the serial port
}
void loop() {
  float sum1 = 0, sum2 = 0, sum3 = 0;
  for (int i = 0; i < 64; i++){
    sum1 += analogRead(sensor1);
    sum2 += analogRead(sensor2);
    sum3 += analogRead(sensor3);
  }
  float volts1 = (sum1 / 64) * 0.0048828125;
  float volts2 = (sum2 / 64) * 0.0048828125;        // value from sensor * (5/1024) - if running 3.3.volts
then change 5 to 3.3
  float volts3 = (sum3 / 64) * 0.0048828125;
  int distance1 = 13 * pow(volts1, -1);
  int distance2 = 13 * pow(volts2, -1);  // worked out from graph 65 = theretical distance / (1/Volts)S -
luckylarry.co.uk
  int distance3 = 13 * pow(volts3, -1);  // worked out from graph 65 = theretical distance / (1/Volts)S -
luckylarry.co.uk
  // distance = distance * 10;
 // Serial.println(distance);
 // delay(500);
 // }


  delay(1000); // slow down serial port
 if (i== true){
  cig1 = distance1;
  pack_dim1 = round(abs(cig1-distance1));
 count1 = round((d1-(distance1+dc))/dp);


  cig2 = distance2;
  pack_dim2 = round(abs(cig2-distance2));
 count2 = round((d1-(distance2+dc))/dp);
 cig3 = distance3;
 pack_dim3 = round(abs(cig3-distance3));
count3 = round((d1-(distance3+dc))/dp);
 i=!i;
}
 if(i== false){
 pack_dim1 = round(abs(cig1-distance1));
 pack_dim2 = round(abs(cig2-distance2));
 pack_dim3 = round(abs(cig3-distance3));
 }
   Serial.print(count1);
 Serial.println(" Packs of cigarettes found on Sensor 1");
 delay(1000);



 Serial.print("The distance on sensor 1 is: ");
 if (distance1 <= 30){
   Serial.print(distance1);   // print the distance
   Serial.println("cm");
 }


 if (distance1 > cig1 && pack_dim1>1.1 ){
count1 = round((d1-(distance1+dc))/dp);
 //Serial.println("1 pack is taken");
 }
 else if (distance1 < cig1 && pack_dim1>1.1 ){
count1 = round((d1-(distance1+dc))/dp);
 //Serial.println("1 pack is added");
 }
 else
 Serial.println("Same number of packs");


   Serial.print(count2);
 Serial.println(" Packs of cigarettes found on Sensor 2");
 delay(1000);
 Serial.print("The distance on sensor 2 is: ");
 if (distance2 <= 30){
   Serial.print(distance2);   // print the distance
   Serial.println("cm");
 }

 if (distance2 > cig2 && pack_dim2>1.1 ){
count2 = round((d1-(distance2+dc))/dp);
```

```
        //Serial.println("1 pack is taken");
        }
        else if (distance2 < cig2 && pack_dim2>1.1 ){
       count2 = round((d1-(distance2+dc))/dp);
        //Serial.println("1 pack is added");
        }
        else
        Serial.println("Same number of packs");


          Serial.print(count3);
        Serial.println(" Packs of cigarettes found on Sensor 3");
        delay(1000);

        Serial.print("The distance on sensor 3 is: ");
        if (distance3 <= 30){
          Serial.print(distance3);   // print the distance
          Serial.println("cm");
        }
        if (distance3 > cig3 && pack_dim3>1.1 ){
       count3 = round((d1-(distance3+dc))/dp);
        //Serial.println("1 pack is taken");
        }
        else if (distance3 < cig3 && pack_dim3>1.1 ){
       count3 = round((d1-(distance3+dc))/dp);
        //Serial.println("1 pack is added");
        }
        else
        Serial.println("Same number of packs");
        delay(1000);
        cig1=distance1;
        cig2=distance2;
        cig3=distance3;
        Serial.print("Total Number of Packs = ");
        total=count1+count2+count3;
        Serial.println(total);
}
```

4.2 Computer Vision and Emotion Detection

<u>What is computer vision?</u>

Computer vision is a branch of artificial intelligence (AI) that allows computers and systems to extract useful information from digital photos, videos, and other visual inputs, as well as to conduct actions or make recommendations based on that data. If artificial intelligence allows computers to think, computer vision allows them to see, watch, and comprehend. Human vision is like computer vision, with the exception that people have a head start. Human vision benefits from lifetimes of context to teach it how to distinguish objects apart, how far away they are, whether they are moving, and whether something is incorrect with an image.

Computer vision teaches computers to execute similar tasks, but using cameras, data, and algorithms rather than retinas, optic nerves, and a visual cortex, it must do it in a fraction of the time. Because a system trained to check items or monitor a production asset can assess hundreds of products or processes per minute, detecting faults or issues that are invisible to humans, it can swiftly outperform humans.

Computer vision is employed in a variety of industries, including energy and utilities, manufacturing, and automotive, and the industry is still growing. By 2022, it is estimated to reach USD 48.6 billion.

How does computer vision work?

A lot of data is required for computer vision. It repeats data analysis until it detects distinctions and, eventually, recognizes images. To teach a computer to recognize automotive tires, for example, it must be fed many tire photos and tire-related materials for it to understand the differences and recognize a tire, particularly one with no faults.

Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

Machine learning is a technique that allows a computer to train itself about the context of visual input using algorithmic models. If enough data is supplied into the model, the computer will "look" at the data and learn to distinguish between images. Instead of someone training the machine to recognize an image, algorithms allow it to learn on its own.

A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way like humans.

A CNN, like a human recognizing a picture from a distance, detects hard edges and simple forms first, then fills in the details as it runs iterations of its predictions. To comprehend single images, a CNN is employed. In video applications, a recurrent neural network (RNN) is used in a similar way to help computers grasp how visuals in a sequence of frames are related to each other.

Computer Vision Examples:

Many firms lack the financial means to establish computer vision laboratories and develop deep learning models and neural networks. They may also be unable to handle large amounts of visual input due to a lack of computer capability. IBM, for example, is assisting by providing computer vision software development services. These services provide cloud-based pre-built learning models while also reducing the demand on computing resources.

Here are a few examples of established computer vision tasks:

- **Image classification** sees an image and can classify it (a dog, an apple, a person's face). More precisely, it can accurately predict that a given image belongs to a certain class. For example, a social media company might want to use it to automatically identify, and segregate objectionable images uploaded by users.
- **Object detection** can use image classification to identify a certain class of image and then detect and tabulate their appearance in an image or video. Examples include detecting damages on an assembly line or identifying machinery that requires maintenance.
- **Object tracking** follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, other cars and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.(7)
- **Content-based image retrieval** uses computer vision to browse, search and retrieve images from large data stores, based on the content of the images rather than metadata tags associated with them. This task can incorporate automatic image annotation that replaces manual image tagging. These tasks can be used for digital asset management systems and can increase the accuracy of search and retrieval.

In the following sections, we will present the applications that we worked on that are related to computer vision with a detailed tutorial.

1. **Face Recognition:**

First, this link contains a 3-hour tutorial on computer vision and python, the software used is PyCharm. At the end of the tutorial, you'll understand the basics of computer vision with python, and build 3 important CV projects. I managed to build the following face recognition code and it worked perfectly!

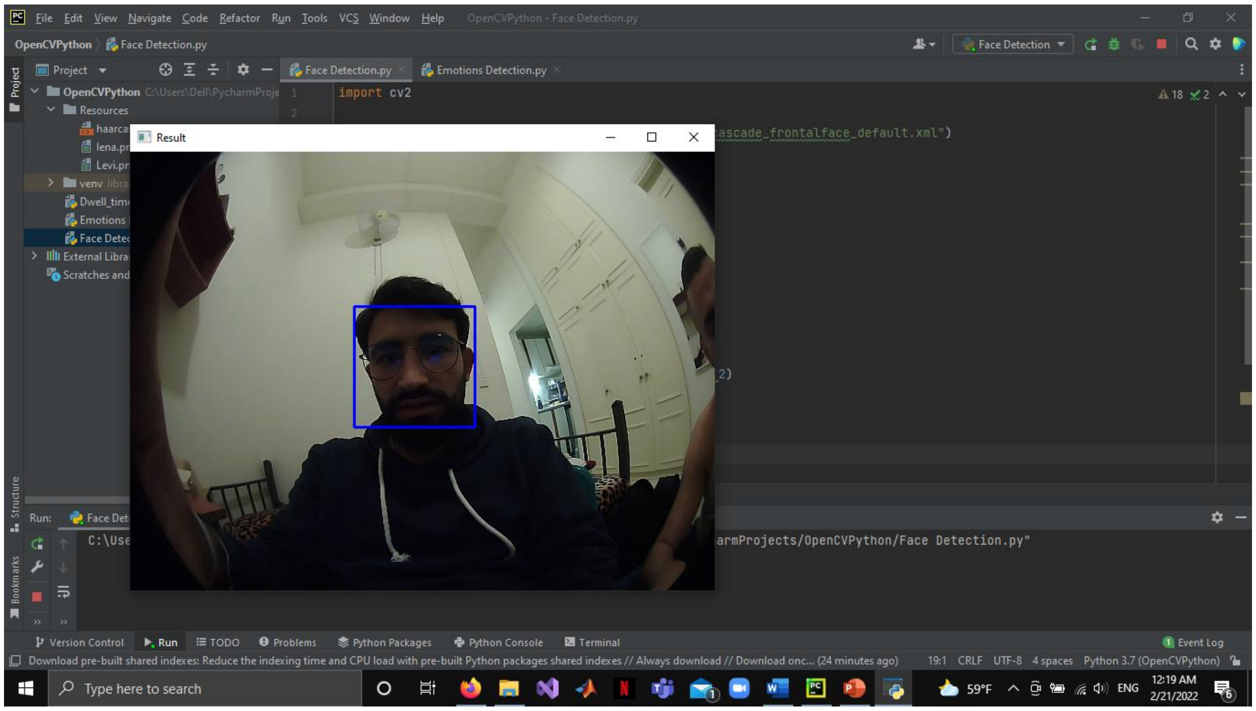
https://youtu.be/WQeoO7MI0Bs

https://youtu.be/oXlwWbU8l2o

```
import cv2
```

```
faceCascade = cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
cap.set(10,100)
while True:
    success, img = cap.read()
    #imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(img,1.1,4)

    for (x,y,w,h) in faces:

        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.imshow("Result", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Result:



```
import cv2
ascade_frontalface_default.xml")
2)
C:\Use
armProjects/OpenCVPython/Face Detection.py"
```

Figure 5.1: Face Recognition

2. Emotion Detection:

In this section, a very known library is used, and it's called DeepFace. I will share couple of links that helped me to get the best emotion detection code.

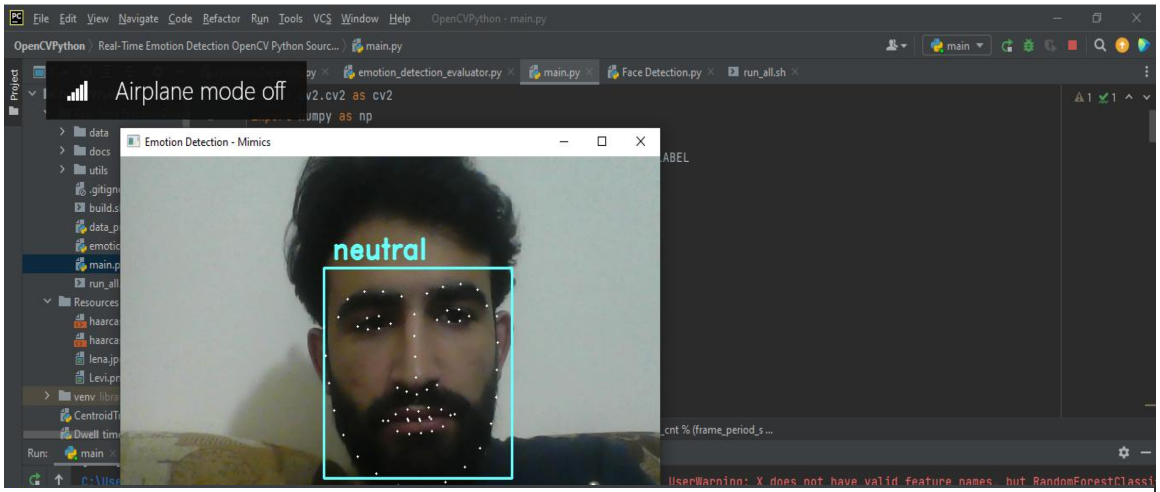
https://youtu.be/rEu5yQVx-s0
https://youtu.be/G1Uhs6NVi-M
https://youtu.be/avv9GQ3b6Qg

After trying a bunch of codes, I found that this is the best emotion detection code:

```
import cv2.cv2 as cv2
import numpy as np

from utils.image_classifier import ImageClassifier, NO_FACE_LABEL

# Color RGB Codes & Font
WHITE_COLOR = (255, 255, 255)
GREEN_COLOR = (0, 255, 0)
BLUE_COLOR = (255, 255, 104)
FONT = cv2.QT_FONT_NORMAL

# Frame Width & Heightq
FRAME_WIDTH = 640
```

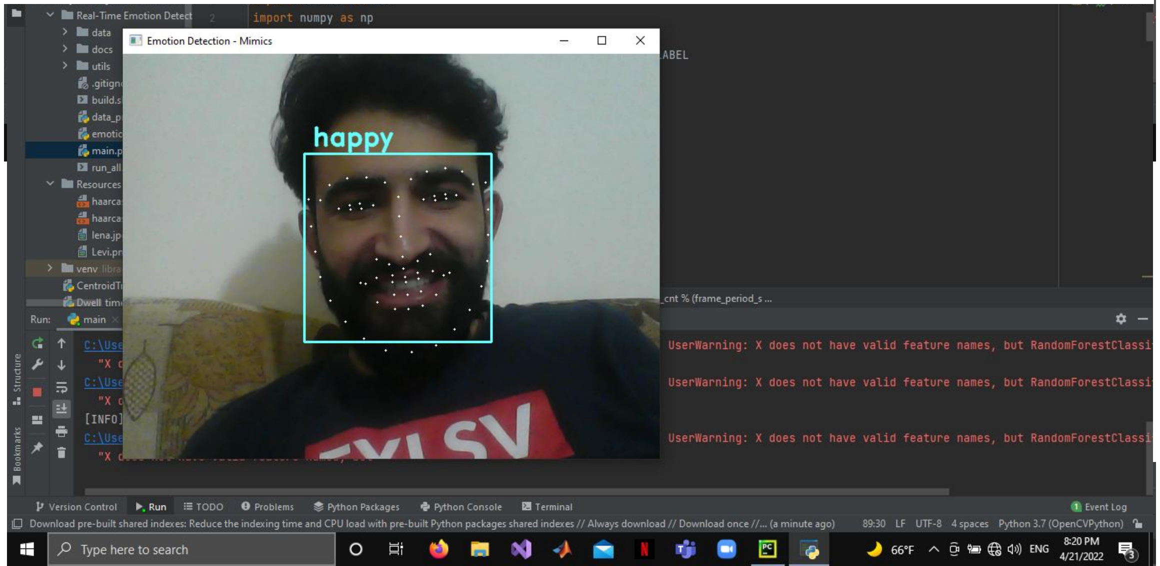
```
FRAME_HEIGHT = 490


class BoundingBox:
    def __init__(self, x, y, w, h):
        self.x = x
        self.y = y
        self.w = w
        self.h = h

    @property
    def origin(self) -> tuple:
        return self.x, self.y

    @property
    def top_right(self) -> int:
        return self.x + self.w

    @property
    def bottom_left(self) -> int:
        return self.y + self.h


def draw_face_rectangle(bb: BoundingBox, img, color=BLUE_COLOR):
    cv2.rectangle(img, bb.origin, (bb.top_right, bb.bottom_left), color, 2)


def draw_landmark_points(points: np.ndarray, img, color=WHITE_COLOR):
    if points is None:
        return None
    for (x, y) in points:
        cv2.circle(img, (x, y), 1, color, -1)


def write_label(x: int, y: int, label: str, img, color=BLUE_COLOR):
    if label == NO_FACE_LABEL:
        cv2.putText(img, label.upper(), (int(FRAME_WIDTH / 2), int(FRAME_HEIGHT / 2)),
FONT, 1, color, 2, cv2.LINE_AA)
    cv2.putText(img, label, (x + 10, y - 10), FONT, 1, color, 2, cv2.LINE_AA)


class RealTimeEmotionDetector:
    CLAHE = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

    vidCapture = None
    def __init__(self, classifier_model: ImageClassifier):
        self.__init_video_capture(camera_idx=0, frame_w=FRAME_WIDTH,
frame_h=FRAME_HEIGHT)
        self.classifier = classifier_model

    def __init_video_capture(self, camera_idx: int, frame_w: int, frame_h: int):
        self.vidCapture = cv2.VideoCapture(camera_idx)
        self.vidCapture.set(cv2.CAP_PROP_FRAME_WIDTH, frame_w)
        self.vidCapture.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_h)

    def read_frame(self) -> np.ndarray:
        rect, frame = self.vidCapture.read()
        return frame

    def transform_img(self, img: np.ndarray) -> np.ndarray:
        # load the input image, resize it, and convert it to gray-scale
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # convert to gray-scale
        resized_img = self.CLAHE.apply(gray_img)  # resize
        return resized_img

    def execute(self, wait_key_delay=33, quit_key='q', frame_period_s=0.75):
        frame_cnt = 0
        predicted_labels = ''
        old_txt = None
        rectangles = [(0, 0, 0, 0)]
        landmark_points_list = [[(0, 0)]]
        while cv2.waitKey(delay=wait_key_delay) != ord(quit_key):
            frame_cnt += 1

            frame = self.read_frame()
            if frame_cnt % (frame_period_s * 100) == 0:
                frame_cnt = 0
                predicted_labels = self.classifier.classify(img=self.transform_img(img=frame))
                rectangles = self.classifier.extract_face_rectangle(img=frame)
                landmark_points_list = self.classifier.extract_landmark_points(img=frame)
            for lbl, rectangle, lm_points in zip(predicted_labels, rectangles, landmark_points_list):
                draw_face_rectangle(BoundingBox(*rectangle), frame)
                draw_landmark_points(points=lm_points, img=frame)
                write_label(rectangle[0], rectangle[1], label=lbl, img=frame)

                if old_txt != predicted_labels:
                    print('[INFO] Predicted Labels:', predicted_labels)
                    old_txt = predicted_labels

            cv2.imshow('Emotion Detection - Mimics', frame)
        cv2.destroyAllWindows()
        self.vidCapture.release()


def run_real_time_emotion_detector(
        classifier_algorithm: str,
        predictor_path: str,
        dataset_csv: str,
        dataset_images_dir: str = None):
    from utils.data_land_marker import LandMarker
    from utils.image_classifier import ImageClassifier
    from os.path import isfile

    land_marker = LandMarker(landmark_predictor_path=predictor_path)

    if not isfile(dataset_csv):  # If data-set not built before.
        print('[INFO]', f'Dataset file: "{dataset_csv}" could not found.')
        from data_preparer import run_data_preparer
        run_data_preparer(land_marker, dataset_images_dir, dataset_csv)
    else:
        print('[INFO]', f'Dataset file: "{dataset_csv}" found.')

    classifier = ImageClassifier(csv_path=dataset_csv, algorithm=classifier_algorithm,
land_marker=land_marker)
    print('[INFO] Opening camera, press "q" to exit..')
    RealTimeEmotionDetector(classifier_model=classifier).execute()


if __name__ == "__main__":
    """The value of the parameters can change depending on the case."""
    run_real_time_emotion_detector(
        classifier_algorithm='RandomForest',  # Alternatively 'SVM'.
        predictor_path='utils/shape_predictor_68_face_landmarks.dat',
        dataset_csv='data/csv/dataset.csv',
        dataset_images_dir='data/raw'
    )
    print('Successfully terminated.')
```

Result:



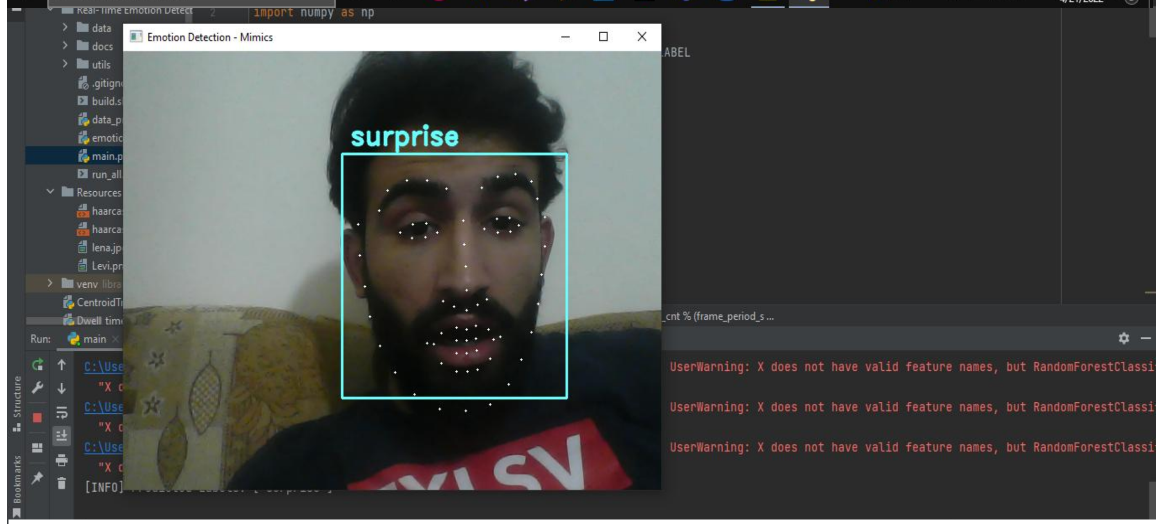
```
import cv2.cv2 as cv2
import numpy as np
UserWarning: X does not have valid feature names, but RandomForestClassi
```

Figure 5.2: Neutral Face



```
import numpy as np
UserWarning: X does not have valid feature names, but RandomForestClassi
UserWarning: X does not have valid feature names, but RandomForestClassi
UserWarning: X does not have valid feature names, but RandomForestClassi
```

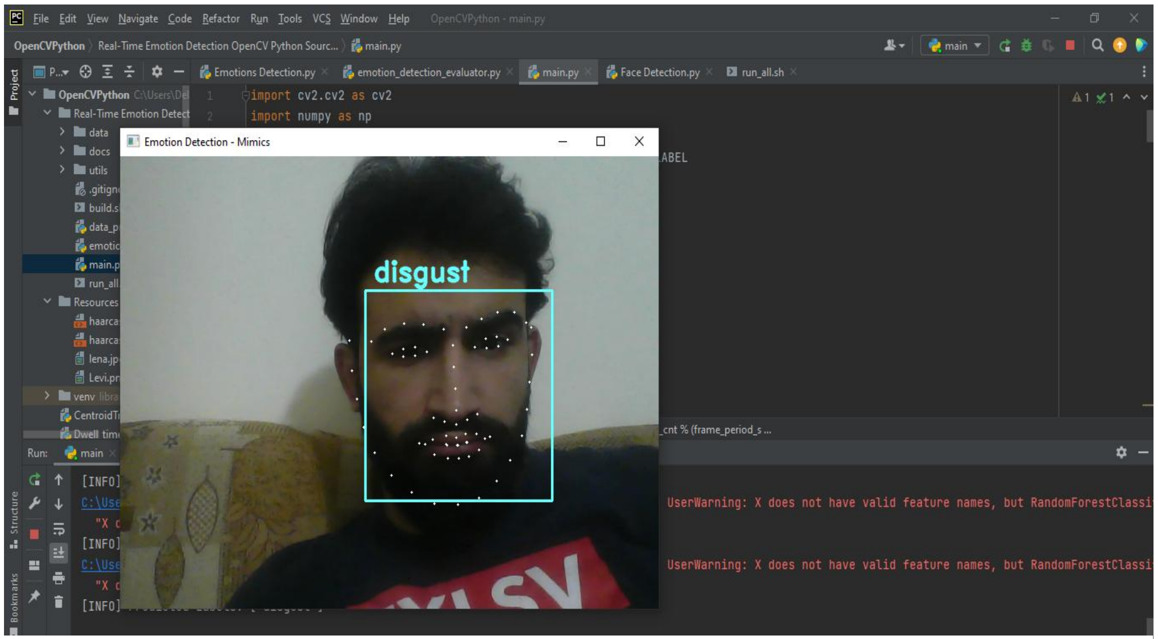
```
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once //... (a minute ago)    89:30  LF  UTF-8  4 spaces  Python 3.7 (OpenCVPython)
```



```
import numpy as np
UserWarning: X does not have valid feature names, but RandomForestClassi
UserWarning: X does not have valid feature names, but RandomForestClassi
UserWarning: X does not have valid feature names, but RandomForestClassi
```

Figure 5.3: Surprise Face

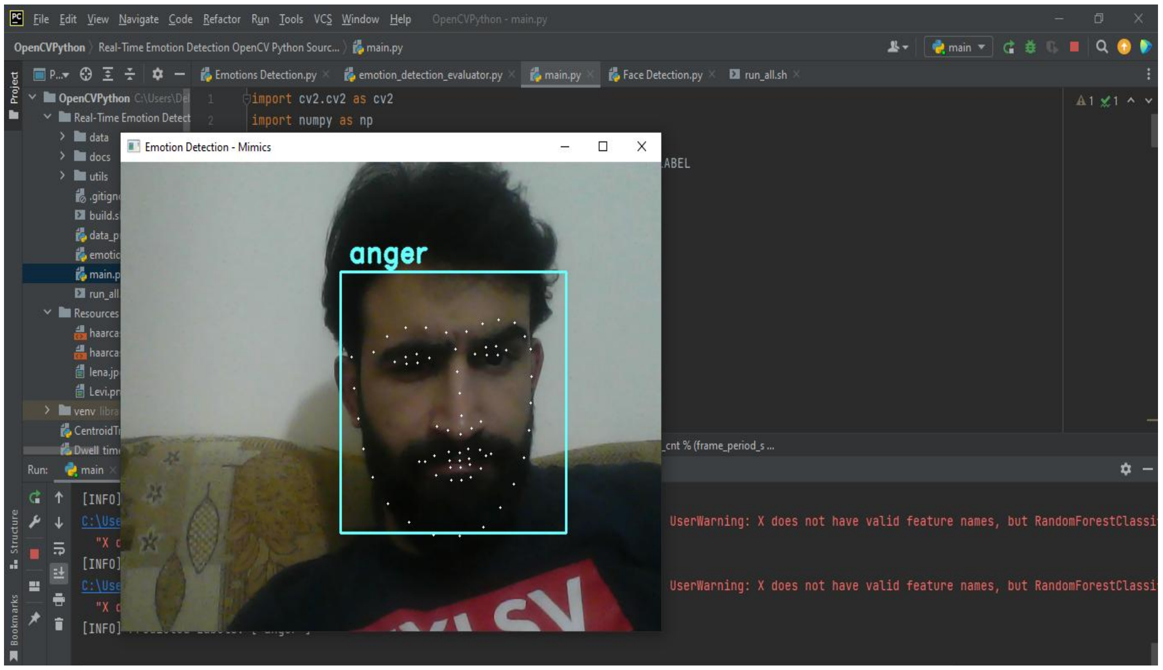
Rafik Hariri University
جامعة رفيق الحريري


File Edit View Navigate Code Refactor Run Tools VCS Window Help OpenCVPython - main.py
OpenCVPython › Real-Time Emotion Detection OpenCV Python Sourc... › main.py
Emotions Detection.py
emotion_detection_evaluator.py
main.py
Face Detection.py
run_all.sh
Emotion Detection - Mimics
disgust

```
import cv2.cv2 as cv2
import numpy as np
```

```
[INFO]
C:\Use
  "X c                UserWarning: X does not have valid feature names, but RandomForestClassi
[INFO]
C:\Use
  "X c                UserWarning: X does not have valid feature names, but RandomForestClassi
[INFO]
```

Figure 5.4: Disgust Face


File Edit View Navigate Code Refactor Run Tools VCS Window Help OpenCVPython - main.py
OpenCVPython › Real-Time Emotion Detection OpenCV Python Sourc... › main.py
Emotions Detection.py
emotion_detection_evaluator.py
main.py
Face Detection.py
run_all.sh
Emotion Detection - Mimics
anger

```
import cv2.cv2 as cv2
import numpy as np
```

```
[INFO]
C:\Use
  "X c                UserWarning: X does not have valid feature names, but RandomForestClassi
[INFO]
C:\Use
  "X c                UserWarning: X does not have valid feature names, but RandomForestClassi
[INFO]
```

Figure 5.5: Anger Face

The limitations of using this code and what needs to improve:

- The camera takes like 2 seconds to display the emotion detected, which is a long duration and needs to be decreased.
- The camera used is not a high-quality camera, and sometimes it detects objects as faces, and sometimes doesn't detect faces directly.

3. Object Detection:

- Materials Used:

- **NVIDEA Jetson-Nano:** NVIDIA® Jetson Nano™ Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts.It's simpler than ever to get started! Just insert a microSD card with the system image, boot the developer kit, and begin using the same NVIDIA JetPack SDK used across the entire NVIDIA Jetson™ family of products. JetPack is compatible with NVIDIA's world-leading AI platform for training and deploying AI software, reducing complexity and effort for developers.

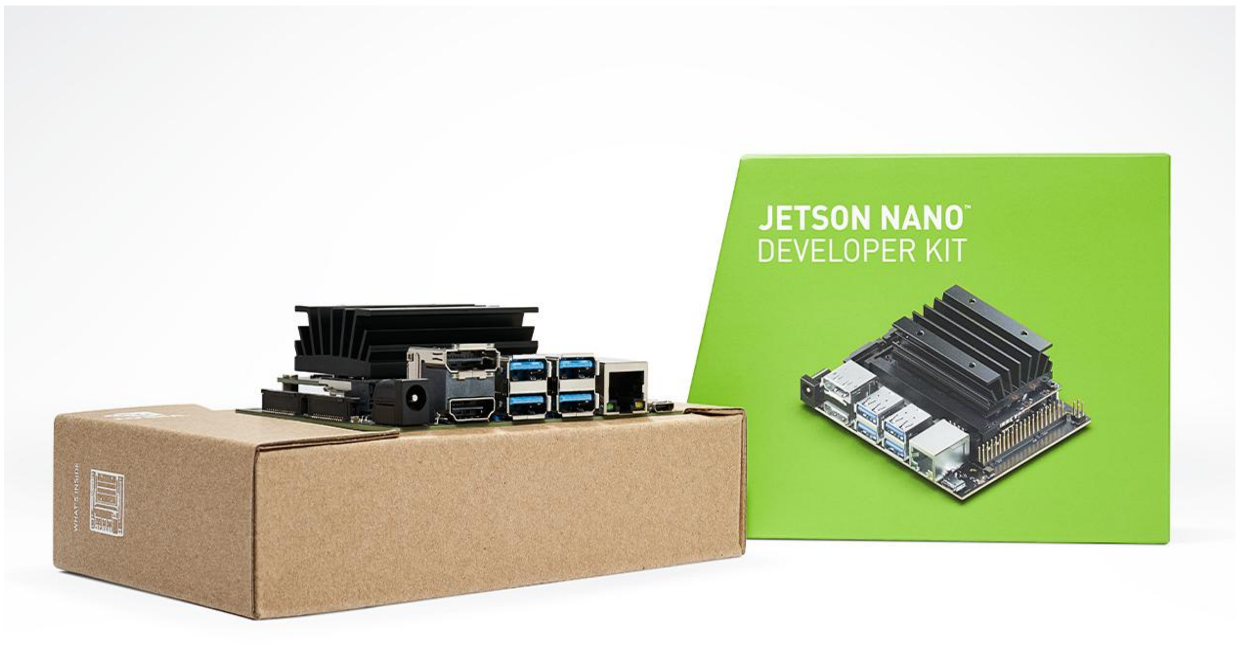


Figure 5.6: Jetson-nano

- **Fisheye Camera**

  - <u>Methodology:</u>

First, to install the jetson-nano and download the packages that are necessary to apply the object detection. The link below is an official reference (NVIDEA), following these steps will apply the object detection perfectly: https://github.com/dusty-nv/jetson-inference.

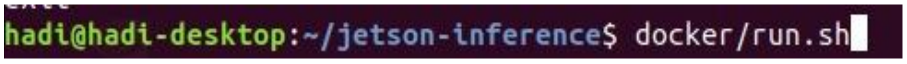
The project will be done in the docker, not in the terminal that we usually use in ubuntu. To setup the docker environment and get it ready, follow this link: https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-docker.md.

After launching the container and installing the proper libraries and extensions, follow the following link to collect your dataset, train your model, and finally test it: https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md.

Now, the detailed steps (screenshots) are as following:

We trained and built a model of 3 products (3 objects): Pipore (Matte), Taj (Salt) and Pajarito (Matte) using Jetson Nano and fish-eye camera. And the process is the following:

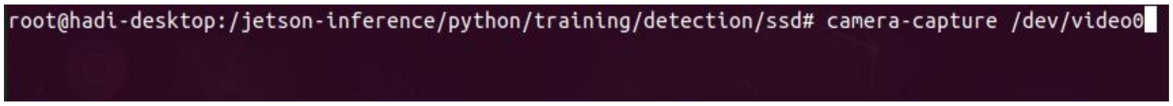
After installing the jetson-inference library, now we need to run the docker container to collect the dataset and train our model (figure 5.7).



```
hadi@hadi-desktop:~/jetson-inference$ docker/run.sh
```

Figure 5.7

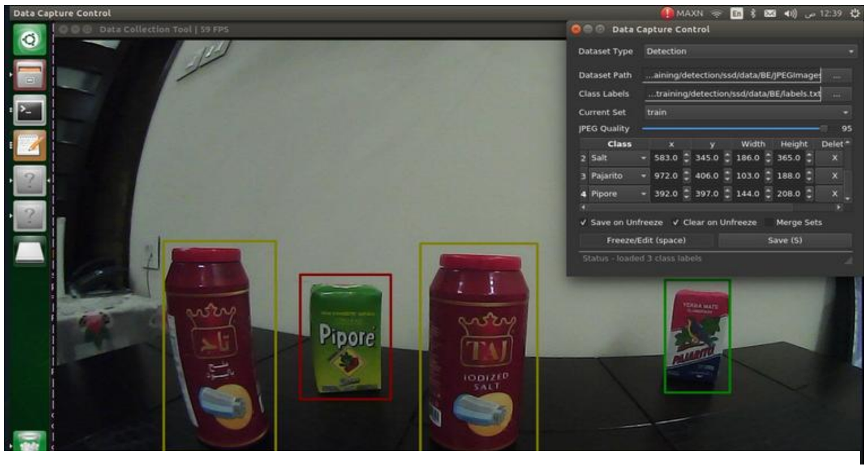
Now we open the camera to start collecting our dataset (figure 5.8).



```
root@hadi-desktop:/jetson-inference/python/training/detection/ssd# camera-capture /dev/video0
```

Figure 5.8

By drawing square with the corresponding label on each product, we took over 300 photos with different positions and angles to have a good and functional dataset (figure 5.9).



Data Capture Control

Dataset Type: Detection
Dataset Path: ...aining/detection/ssd/data/BE/JPEGImages
Class Labels: ...training/detection/ssd/data/BE/labels.txt
Current Set: train
JPEG Quality: 95

	Class	x	y	Width	Height	Delet
2	Salt	583.0	345.0	186.0	365.0	X
3	Pajarito	972.0	406.0	103.0	188.0	X
4	Pipore	392.0	397.0	144.0	208.0	X

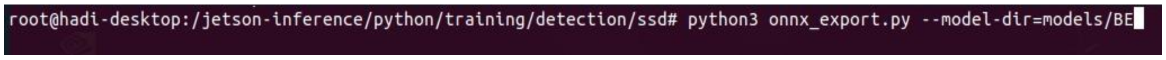
✓ Save on Unfreeze ✓ Clear on Unfreeze Merge Sets

Freeze/Edit (space) Save (S)

Status - loaded 3 class labels

Figure 5.9

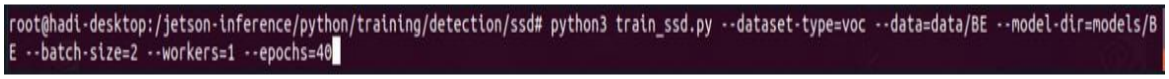
Now we need to export our model into an onnx file type, to prepare it to the training phase (figure 5.91).



```
root@hadi-desktop:/jetson-inference/python/training/detection/ssd# python3 onnx_export.py --model-dir=models/BE
```

Figure 5.91

To ensure a very high accuracy for our model, we chose 40 as epochs value. The model took like 3 hours to finish the training phase, and we got 75% of accuracy which is awesome!



```
root@hadi-desktop:/jetson-inference/python/training/detection/ssd# python3 train_ssd.py --dataset-type=voc --data=data/BE --model-dir=models/B
E --batch-size=2 --workers=1 --epochs=40
```

Figure 5.92

Finally, we ran the detectnet method to test our model and apply the object detection for our 3 products (figure 5.93).

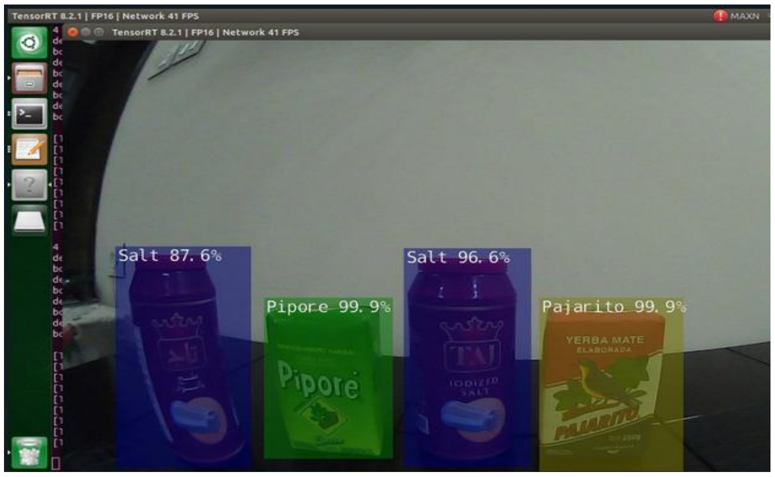


Figure 5.93

4.3 Product Count Using Half-Bridge Weight Sensor

ABSTRACT

The objective of this experiment is to find the quantity of products found on the stand. We should design and implement a circuit using an Arduino Uno and Half-bridge weight sensor. As a conclusion, screen counter decrements as products taken off the pusher and vice versa.

Keywords: Arduino Uno, Half-bridge weight sensor, Display Screen.

1. INTRODUCTION

The main objective behind this experiment is to find the quantity of a specific product found on the stand. First, we fix the Half-bridge sensor to set an initial value of the weight. As the customer takes products off the stand, the measured weight decreases hence indicating the decrease in products' quantity. On the other hand, and in case the customer put any taken product back on the stand, the measured weight increases indicating the increase in products' quantity. Flowchart is shown in figure 6.1.

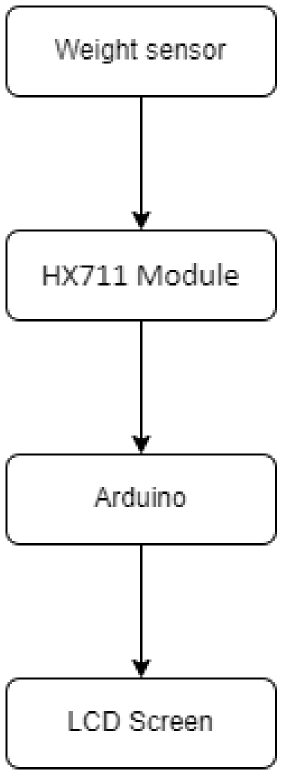


FIGURE 6.1: *Block Diagram*

2. MATERIALS AND METHODS

1.1 Components

- Half-bridge weight sensor:

  This weight sensor is suitable for electronic balance and other high accuracy electronic weighing devices. When measuring, the correct force is applied to the outer side of the strain E-shaped beam portion of the sensor and the outside edges to form a shear force in the opposite direction. [7]

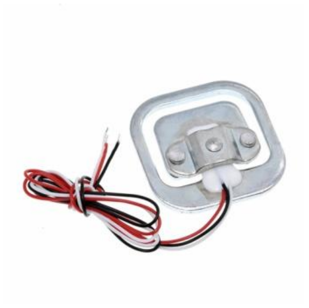


FIGURE 6.3: Half-bridge sensor

- Arduino UNO:

  Arduino Uno as shown in fig.6.3 is a microcontroller board that has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. [8]

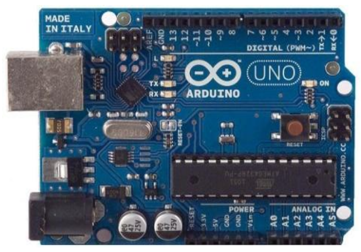


FIGURE 6.3: ARDUINO UNO

- LCD Screen:

  An LCD (Liquid Crystal Display) screen as shown in fig.6.4 is an electronic display module and has a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix displays is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.[9]

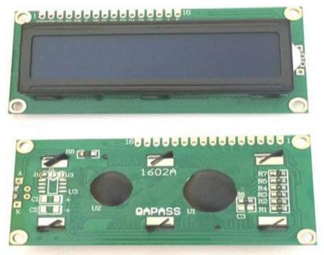


FIGURE 6.4: LCD SCREEN

- HX711 Module:

  HX711 as shown in fig.6.5 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.[10]

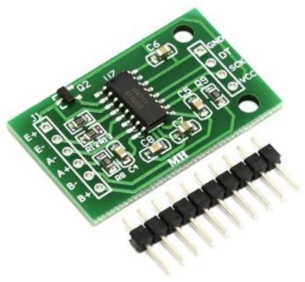


FIGURE 6.5: HX711 module

1.2 Methodology

When measuring, the correct force is applied to the outer side of the strain E-shaped beam portion of the sensor and the outside edges to form a shear force in the opposite direction.

The measured weight read by the half-bridge weight sensor is calculated using the input voltage of the sensor into the Arduino. Moreover, map the input voltage between 0 and 5V into integer values between 0 and 1023 (10 bit).

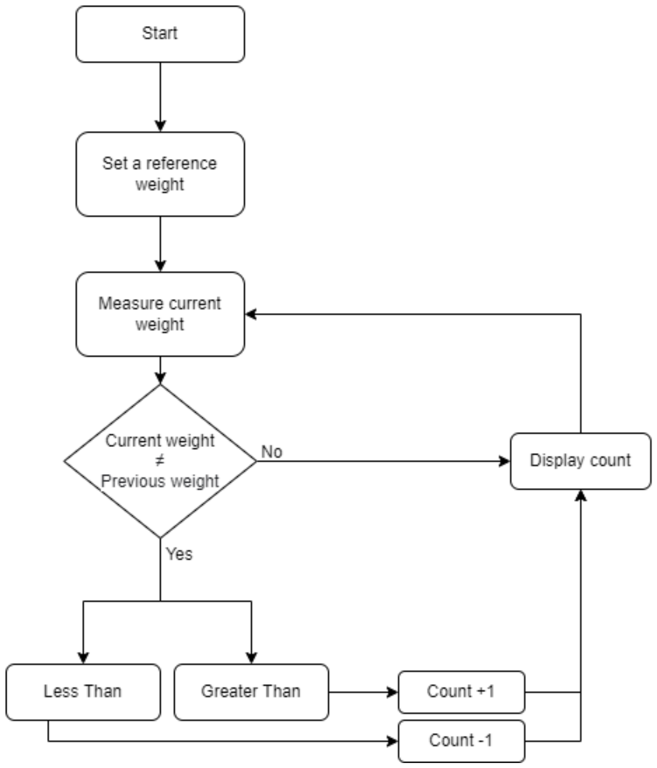


FIGURE 6.6: HALF-BRIDGE WEIGHT SENSOR FLOW CHART

3. RESULTS AND DISCUSSION

We first connect the half-bridge weight Sensor to the Arduino Uno by the HX711 module, in which we connected the HX711 module VCC to 5V, GND pin to ground, DT to pin 4 and the SCK to pin 5. Moreover, we connected the LCD screen to the Arduino in order to display the measured distance.

Second, we fixed the weight sensor to a plate of wood hence as the custom purchase any of the products found, the weight measured decreases hence indicating the decrement in the count of products. On the other hand, in case the customer adds a product on the Sensor, the weight measured by the sensor increases hence indicating the increment by the count of products.

Third, if the customer adds a product that wasn't initially found on the wood plate, in which the product added, has different weight from the ones already found. The new weight measured won't be compatible with the previous one hence the LCD screen won't increment the count of product and displays that a new type of products was added to the retail pusher.

Problems faced:

As the customer adds a product with the same or a duplicate weight as the initial ones found on the wood plate (weight sensor), the count increment by 1 or the weight duplicate. Moreover, the sensor is not accurate enough for products with light weight.

Hence, the problems faced are:

- Items with the same weight cannot be differentiated.
- Items with duplicate weight cannot be differentiated.
- Items with Light weight cannot be differentiated or read accurately.

Proposed Solutions:

In order to better implementation of the experiment and achieve better results, several solutions could be proposed:

- Integrating the weight sensor with another sensor (Ex: camera)
- Using more accurate sensor.
- Stop the counter from adding duplicates.
- Changing the planogram.

The circuit:

Below is the implemented circuit using Arduino UNO, Half-bridge weight sensor, wood plate, HX711 Module, LCD screen and jumper wires (figure6.7&6.8).

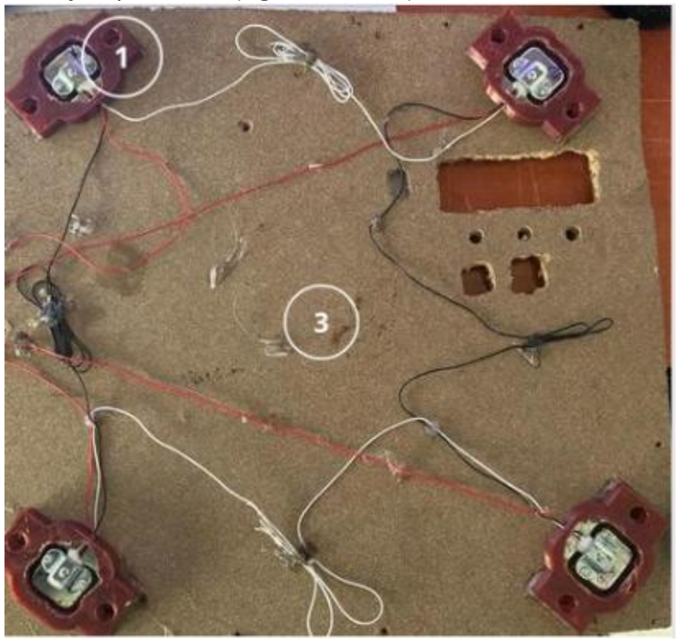


FIGURE 6.7: The Implemented Circuit

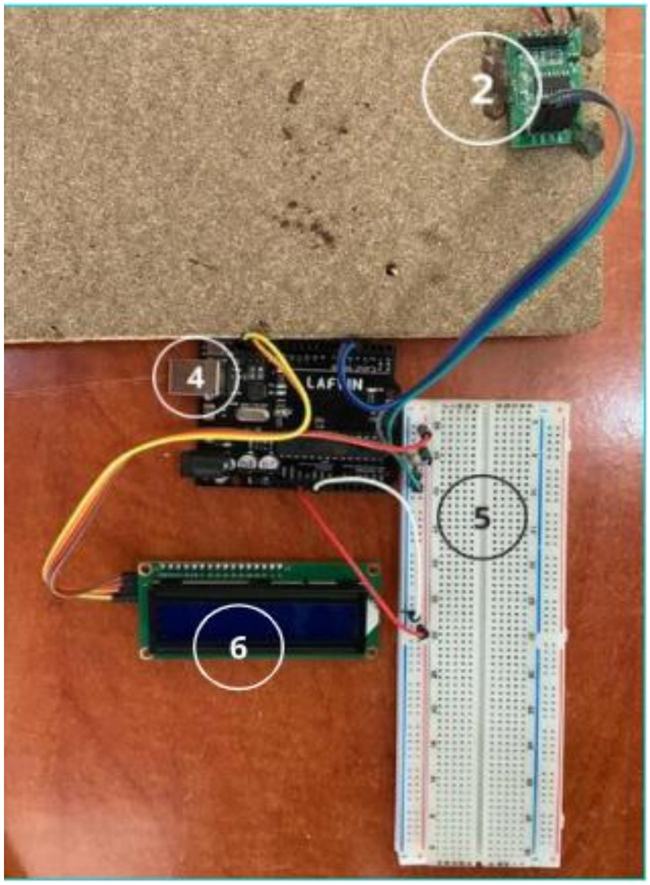


FIGURE 6.8: The Implemented Circuit

COMPONENT'S NUMBER	COMPONENT'S NAME
1	Half-bridge Weight sensor
2	HX711 Module
3	Wood Plate
4	ARDUINO UNO
5	Bread board
6	LCD Screen

TABLE 2: CIRCUIT COMPONENT'S NAME AS SHOWN IN FIGURE 7 and 8

4. IMPLEMENTATION And RESULTS

We implemented two weight sensors on 2 PVC plates, and as the code runs we randomly added and took products (Water bottles and mate packs) on and off the weight sensor in order to observe the variation in the count of products' number found on a certain weight sensor or PVC plate and detected by the sensor. Moreover, the code calculates the total number of products found with every random selection done by the customer, hence giving a precise study about the count of products.

The following figures (6.9, 6.91, 6.92, 6.93, and 6.94) show the results of the experiment and count of products on every retail pusher:

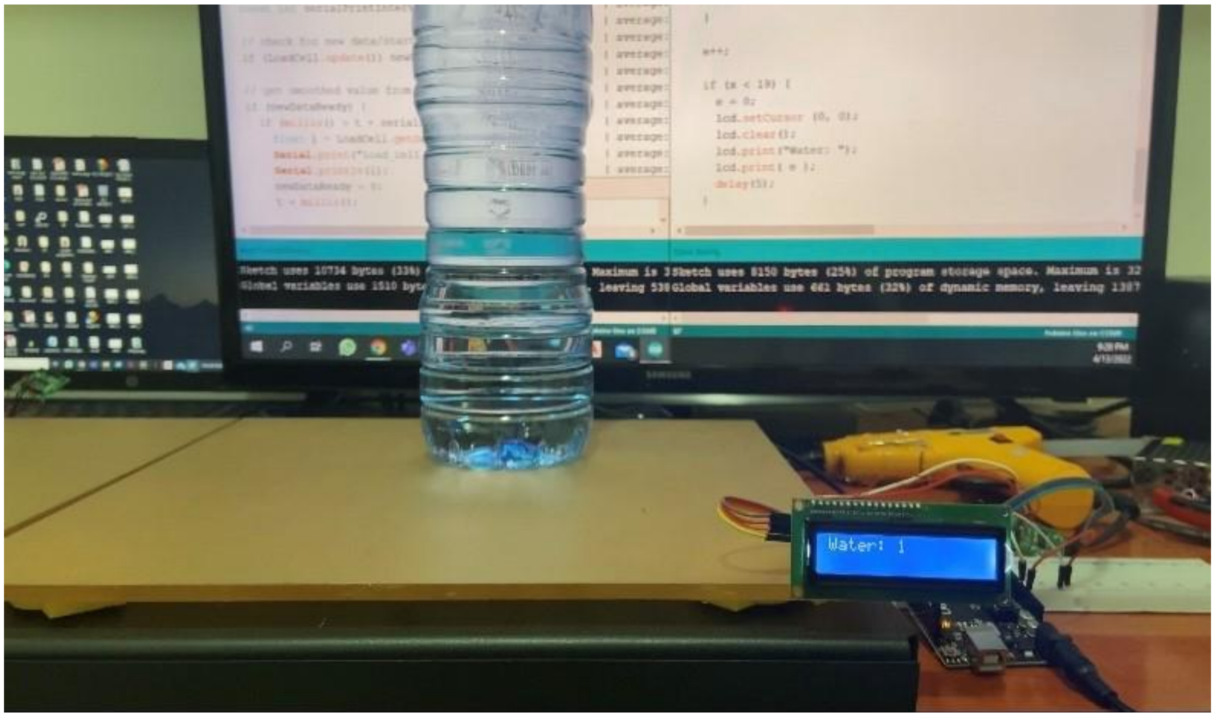


Figure 6.9: Calculated number of Products on Every Sensor

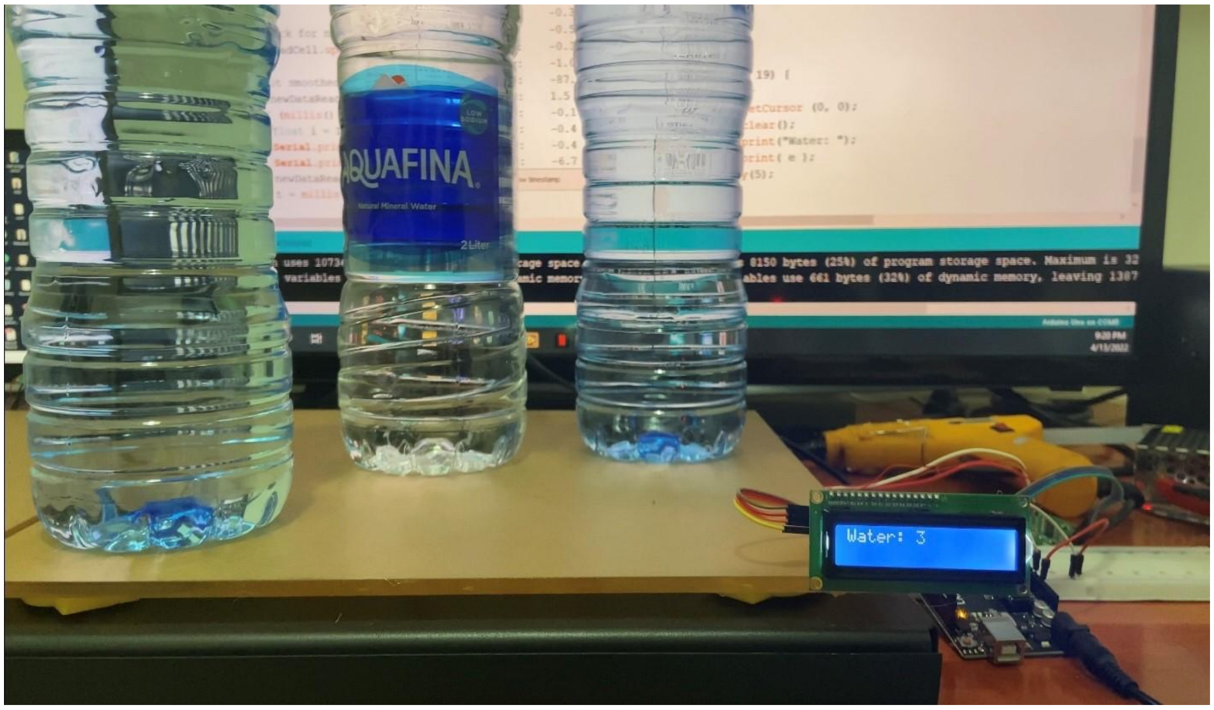


Figure 6.91: Calculated number of Products on Every Sensor

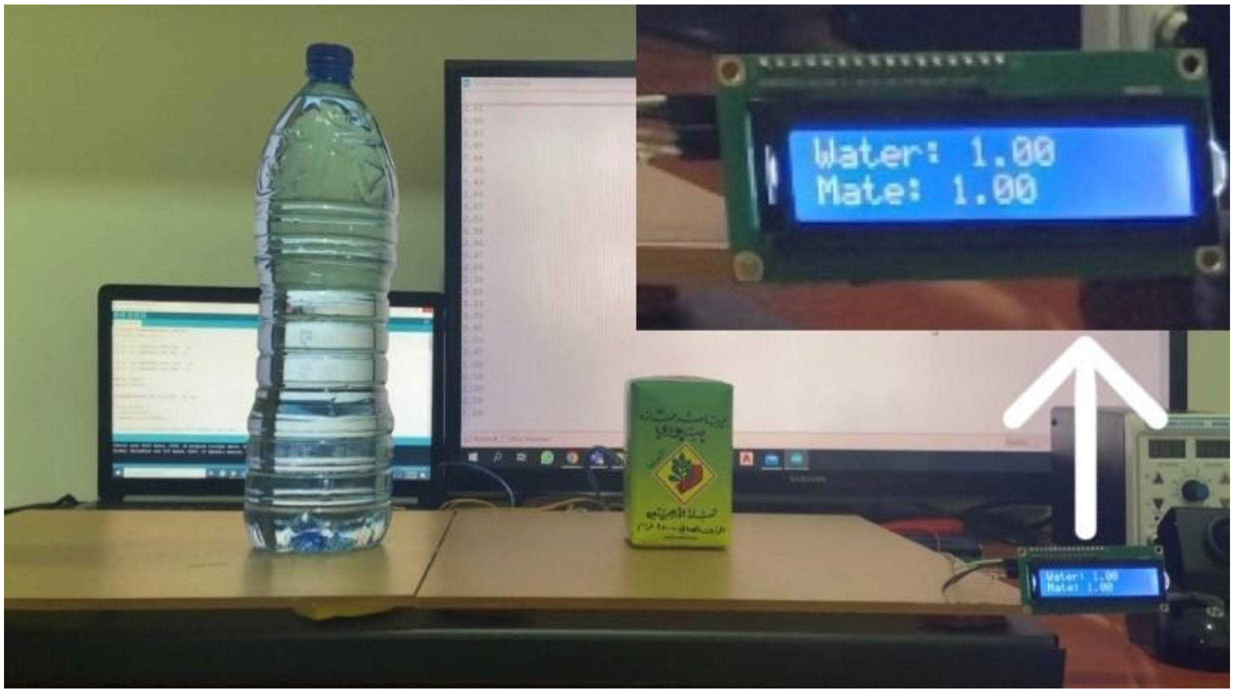


Figure 6.92: Calculated number of Products on Every Sensor

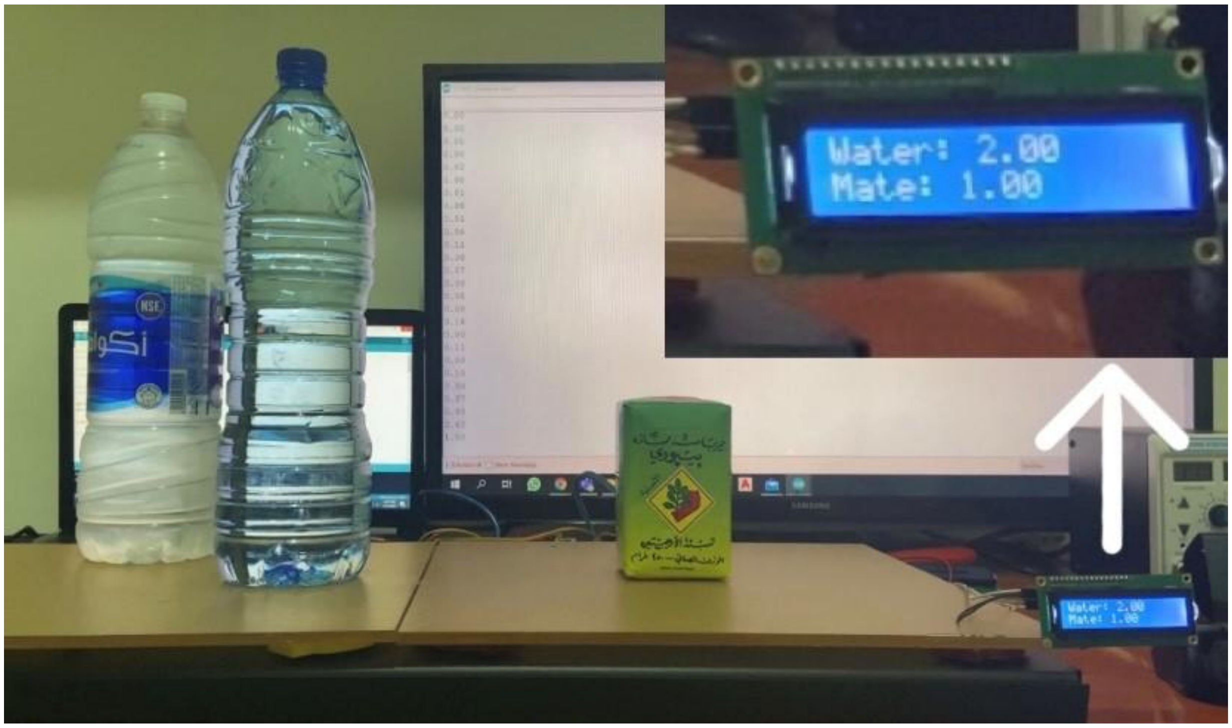


Figure 6.93: Calculated number of Products on Every Sensor

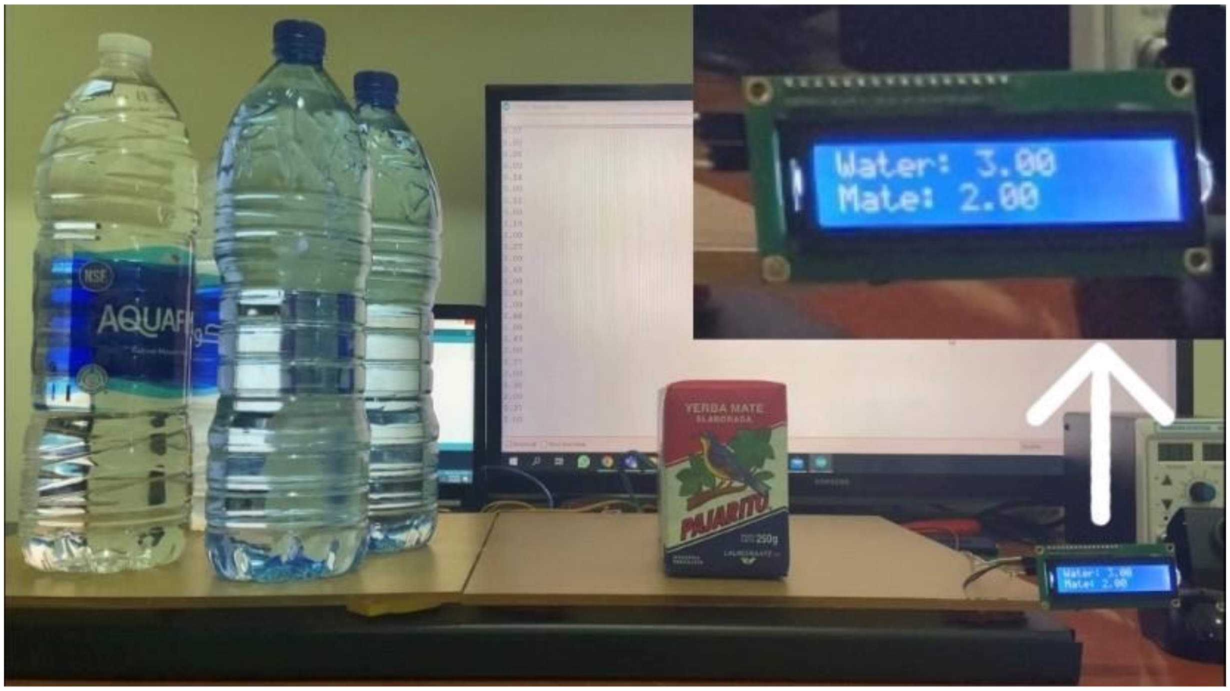


Figure 6.94: Calculated number of Products on Every Sensor

5. CONCLUSION

As stated before, the main objective behind the above experiment is to find the quantity of products placed on the Stand (wood plate) as a function of weight read by the Half-bridge weight sensor. As customers take-off products, weight sensor reads a weight less than the initial one hence the counter displayed on the screen decrements showing the new number of products found on the pusher. On the other hand, the counter increments and the screen also displays the new number of products found.

6. APPENDIX

Code for 1 weight sensor

```
#include <LiquidCrystal_I2C.h>
#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.begin();
  Serial.begin(57600); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  unsigned long stabilizingtime = 2000;
  boolean _tare = true;
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
    LoadCell.setCalFactor(1.0);
    Serial.println("Startup is complete");
  }
  while (!LoadCell.update());
  calibrate();
}

void loop() {

  int e, j, k;
  int l;
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0;

  if (LoadCell.update()) newDataReady = true;

   if (newDataReady) {
    if (millis() > t + serialPrintInterval) {
      float i = LoadCell.getData();
      Serial.print("Load_cell output val: ");
      Serial.println(i);
      newDataReady = 0;
      t = millis();

    }
  }

  if (Serial.available() > 0) {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay();
    else if (inByte == 'r') calibrate();
    else if (inByte == 'c') changeSavedCalFactor();
    }

  if (LoadCell.getTareStatus() == true) {
    Serial.println("Tare complete");
  }
  int f = LoadCell.getData();
  for ( e = 1; e <= 3; e++) {

    if ( (f - (k * 280)) <= e * 2100 && (f - (k * 220)) >= e * 1900) {
      j = e;
      lcd.setCursor (0, 0);
      lcd.print("Water: ");
      lcd.print( e );
      delay(5);
    }

    if (f < 1900) {
      j = 0;
      lcd.setCursor (0, 0);
      lcd.print("Water: 0 ");
      delay(5);
    }

  }

  for ( l = 1; l <= 2; l++) {
    if ( (f - (j * 2050)) <= l * 300 && (f - (j * 1950)) >= l * 200) {
      k = l;
      lcd.setCursor (0, 1);
      lcd.print("Mate: ");
      lcd.print( l );
      delay(5);
    }

    if (f <= 150) {
      k = 0;
      lcd.setCursor (0, 1);
      lcd.print("Mate: 0 ");
      delay(5);
    }
  }

}

void calibrate() {
  Serial.println("***");
  Serial.println("Start calibration:");
  Serial.println("Place the load cell an a level stable surface.");
  Serial.println("Remove any load applied to the load cell.");
  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      if (Serial.available() > 0) {
        char inByte = Serial.read();
        if (inByte == 't') LoadCell.tareNoDelay();
      }
    }
    if (LoadCell.getTareStatus() == true) {
      Serial.println("Tare complete");
      _resume = true;
    }
  }

  Serial.println("Now, place your known mass on the loadcell.");
  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");

  float known_mass = 0;
  _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      known_mass = Serial.parseFloat();
      if (known_mass != 0) {
        Serial.print("Known mass is: ");
        Serial.println(known_mass);
        _resume = true;
      }
    }
  }

  LoadCell.refreshDataSet();
  float newCalibrationValue = LoadCell.getNewCalibration(known_mass);

  Serial.print("New calibration value has been set to: ");
  Serial.print(newCalibrationValue);
  Serial.println(", use this as calibration value (calFactor) in your project sketch.");
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");

  _resume = false;
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;

      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

  Serial.println("End calibration");
```

```
 Serial.println("***");
 Serial.println("To re-calibrate, send 'r' from serial monitor.");
 Serial.println("For manual edit of the calibration value, send 'c' from serial monitor.");
 Serial.println("***");
}

void changeSavedCalFactor() {
 float oldCalibrationValue = LoadCell.getCalFactor();
 boolean _resume = false;
 Serial.println("***");
 Serial.print("Current value is: ");
 Serial.println(oldCalibrationValue);
 Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
 float newCalibrationValue;
 while (_resume == false) {
  if (Serial.available() > 0) {
   newCalibrationValue = Serial.parseFloat();
   if (newCalibrationValue != 0) {
    Serial.print("New calibration value is: ");
    Serial.println(newCalibrationValue);
    LoadCell.setCalFactor(newCalibrationValue);
    _resume = true;
   }
  }
 }
 _resume = false;
 Serial.print("Save this value to EEPROM adress ");
 Serial.print(calVal_eepromAdress);
 Serial.println("? y/n");
 while (_resume == false) {
  if (Serial.available() > 0) {
   char inByte = Serial.read();
   if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
    EEPROM.begin(512);
#endif
    EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
    EEPROM.commit();
#endif
    EEPROM.get(calVal_eepromAdress, newCalibrationValue);
    Serial.print("Value ");
    Serial.print(newCalibrationValue);
    Serial.print(" saved to EEPROM address: ");
    Serial.println(calVal_eepromAdress);
    _resume = true;
   }
   else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }
  Serial.println("End change calibration value");
  Serial.println("***");
}
```

Code for 2 weight sensor

```
#include <LiquidCrystal_I2C.h>
#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

//pins:
const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  lcd.begin();
  Serial.begin(57600);
  delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  //LoadCell.setReverseOutput(); //uncomment to turn a negative output value to positive
  unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a few
seconds of stabilizing time
  boolean _tare = true; //set this to false if you don't want tare to be performed in the next step
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag() || LoadCell.getSignalTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
  while (1);
 }
 else {
  LoadCell.setCalFactor(1.0); // user set calibration value (float), initial value 1.0 may be used for this
sketch
  Serial.println("Startup is complete");
 }
 while (!LoadCell.update());
 calibrate(); //start calibration procedure
}

void loop() {

 int e, j, k;
 int l;
 static boolean newDataReady = 0;
 const int serialPrintInterval = 0; //increase value to slow down serial print activity

 // check for new data/start next conversion:
 if (LoadCell.update()) newDataReady = true;

 // get smoothed value from the dataset:
 if (newDataReady) {
  if (millis() > t + serialPrintInterval) {
   float i = LoadCell.getData();
   Serial.print("Load_cell output val: ");
   Serial.println(i);
   newDataReady = 0;
   t = millis();
  }
 }
 // receive command from serial terminal
 if (Serial.available() > 0) {
  char inByte = Serial.read();
  if (inByte == 't') LoadCell.tareNoDelay(); //tare
  else if (inByte == 'r') calibrate(); //calibrate
  else if (inByte == 'c') changeSavedCalFactor(); //edit calibration value manually
 }

 // check if last tare operation is complete
 if (LoadCell.getTareStatus() == true) {
  Serial.println("Tare complete");
 }
 int f = LoadCell.getData();
 for ( e = 1; e <= 3; e++) {

  if ( (f - (k * 280)) <= e * 2100 && (f - (k * 220)) >= e * 1900) {
   j = e;
```

```
    lcd.setCursor (0, 0);
    lcd.print("Water: ");
    lcd.print( e );
    delay(5);
   }

   if (f < 1900) {
    j = 0;
    lcd.setCursor (0, 0);
    lcd.print("Water: 0 ");
    delay(5);
   }

  }

  for ( l = 1; l <= 2; l++) {

   if ( (f - (j * 2050)) <= l * 300 && (f - (j * 1950)) >= l * 200) {
    k = l;
    lcd.setCursor (0, 1);
    lcd.print("Mate: ");
    lcd.print( l );
    delay(5);
   }

   if (f <= 150) {
    k = 0;
    lcd.setCursor (0, 1);
    lcd.print("Mate: 0 ");
    delay(5);
   }
  }
}
void calibrate() {
  Serial.println("***");
  Serial.println("Start calibration:");
  Serial.println("Place the load cell an a level stable surface.");
  Serial.println("Remove any load applied to the load cell.");
  Serial.println("Send 't' from serial monitor to set the tare offset.");

  boolean _resume = false;
  while (_resume == false) {
   LoadCell.update();
   if (Serial.available() > 0) {
    if (Serial.available() > 0) {
     char inByte = Serial.read();
     if (inByte == 't') LoadCell.tareNoDelay();
    }
    }
    if (LoadCell.getTareStatus() == true) {
      Serial.println("Tare complete");
      _resume = true;
    }
  }

  Serial.println("Now, place your known mass on the loadcell.");
  Serial.println("Then send the weight of this mass (i.e. 100.0) from serial monitor.");

  float known_mass = 0;
  _resume = false;
  while (_resume == false) {
    LoadCell.update();
    if (Serial.available() > 0) {
      known_mass = Serial.parseFloat();
      if (known_mass != 0) {
        Serial.print("Known mass is: ");
        Serial.println(known_mass);
        _resume = true;
      }
    }
  }

  LoadCell.refreshDataSet(); //refresh the dataset to be sure that the known mass is measured correct
  float newCalibrationValue = LoadCell.getNewCalibration(known_mass); //get the new calibration value

  Serial.print("New calibration value has been set to: ");
  Serial.print(newCalibrationValue);
  Serial.println(", use this as calibration value (calFactor) in your project sketch.");
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");

  _resume = false;
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.begin(512);
#endif
        EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;

      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

  Serial.println("End calibration");
  Serial.println("***");
  Serial.println("To re-calibrate, send 'r' from serial monitor.");
  Serial.println("For manual edit of the calibration value, send 'c' from serial monitor.");
  Serial.println("***");
}

void changeSavedCalFactor() {
  float oldCalibrationValue = LoadCell.getCalFactor();
  boolean _resume = false;
  Serial.println("***");
  Serial.print("Current value is: ");
  Serial.println(oldCalibrationValue);
  Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
  float newCalibrationValue;
  while (_resume == false) {
    if (Serial.available() > 0) {
      newCalibrationValue = Serial.parseFloat();
      if (newCalibrationValue != 0) {
        Serial.print("New calibration value is: ");
        Serial.println(newCalibrationValue);
        LoadCell.setCalFactor(newCalibrationValue);
        _resume = true;
      }
    }
  }
  _resume = false;
  Serial.print("Save this value to EEPROM adress ");
  Serial.print(calVal_eepromAdress);
  Serial.println("? y/n");
  while (_resume == false) {
    if (Serial.available() > 0) {
      char inByte = Serial.read();
      if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
```

```
       EEPROM.begin(512);
#endif
       EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
       EEPROM.commit();
#endif
       EEPROM.get(calVal_eepromAdress, newCalibrationValue);
       Serial.print("Value ");
       Serial.print(newCalibrationValue);
       Serial.print(" saved to EEPROM address: ");
       Serial.println(calVal_eepromAdress);
       _resume = true;
      }
      else if (inByte == 'n') {
       Serial.println("Value not saved to EEPROM");
       _resume = true;
      }
    }
  }
  Serial.println("End change calibration value");
  Serial.println("***");
}
```

4.4 Product Count Using Full-Bridge Weight Sensor

ABSTRACT

The objective of this experiment is to find the quantity of products found on the stand. We should design and implement a circuit using an Arduino Uno and Full-bridge weight sensor. As a conclusion, screen counter decrements as products taken off the pusher and vice versa.

Keywords: Arduino Uno, Full-bridge weight sensor, Display Screen.

1. INTRODUCTION

The main objective behind this experiment is to find the quantity of a specific product found on the stand. First, we fix the Full-bridge sensor to set an initial value of the weight. As the customer takes products off the stand, the measured weight decreases hence indicating the decrease in products' quantity. On the other hand, and in case the customer put any taken product back on the stand, the measured weight increases indicating the increase in products' quantity. Flowchart is shown in figure 1.

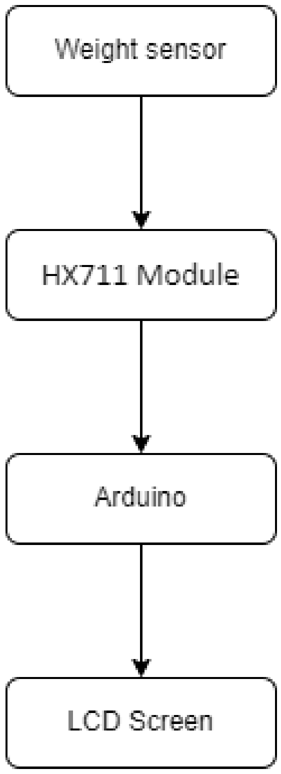


FIGURE 7.4: Block Diagram

2. MATERIALS AND METHODS

1.5 Components

- Full-bridge weight sensor:

  A load cell is a force transducer. It converts an applied force into an electrical signal that can be measured. The electrical signal changes proportionally to the force applied. Multiple kinds of load cells exist: hydraulic, pneumatic, etc. but we are focusing on the most common type – strain gauge load cells.[11]

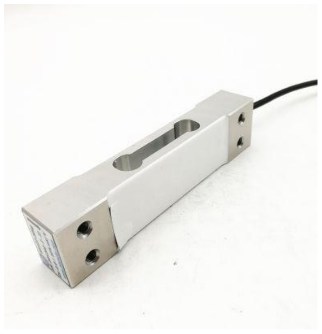


FIGURE 7.5: Full-bridge sensor

- Arduino UNO:

  Arduino Uno as shown in fig.3 is a microcontroller board that has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. [8]

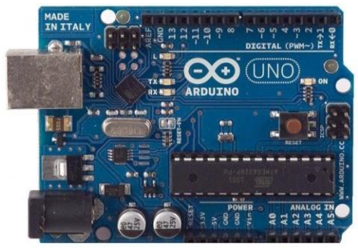


FIGURE 7.3: ARDUINO UNO

- LCD Screen:

  An LCD (Liquid Crystal Display) screen as shown in fig.4 is an electronic display module and has a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix displays is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.[9]

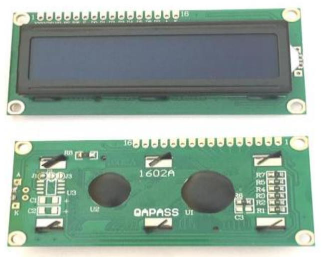


Figure 7.4:LCD Screen

- HX711 Module:

  HX711 as shown in fig.5 is a precision 24-bit analog-to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor.[10]

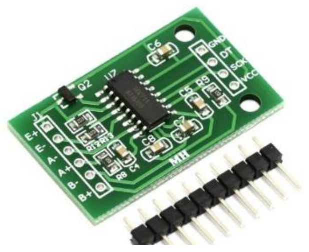


FIGURE 7.5: HX711

2.5 Methodology

It converts an applied force into an electrical signal that can be measured. The electrical signal changes proportionally to the force applied.

The measured weight read by the Full-bridge weight sensor is calculated using the input voltage of the sensor into the Arduino. Moreover, map the input voltage between 0 and 5V into integer values between 0 and 1023 (10 bit).

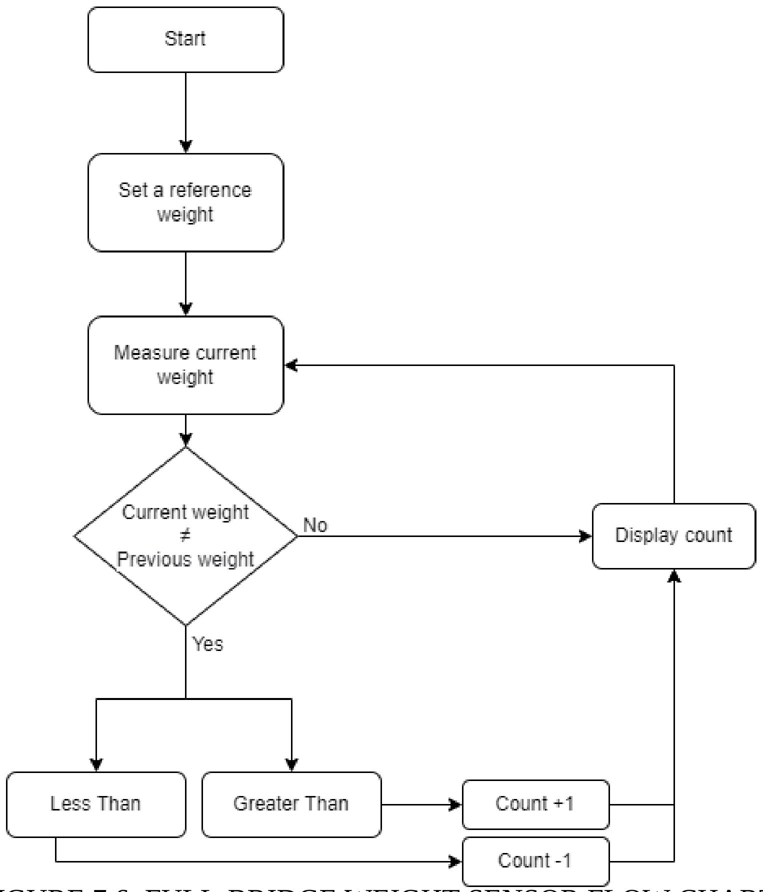


FIGURE 7.6: FULL-BRIDGE WEIGHT SENSOR FLOW CHART

2. RESULTS AND DISCUSSION

We first connect the Full-bridge weight Sensor to the Arduino Uno by the HX711 module, in which we connected the HX711 module VCC to 5V, GND pin to ground, DT to pin 4 and the SCK to pin 5. Moreover, we connected the LCD screen to the Arduino in order to display the measured distance.

Second, we fixed the weight sensor to a plate of PVC hence as the custom purchase any of the products found, the weight measured decreases hence indicating the decrement in the count of products. On the other hand, in case the customer adds a product on the sensor, the weight measured by the sensor increases hence indicating the increment by the count of products.

Third, if the customer adds a product that wasn't initially found on the PVC plate, in which the product added, has different weight from the ones already found. The new weight measured won't be compatible with the previous one hence the LCD screen won't increment the count of product and displays that a new type of products was added to the retail pusher.

Problems faced:

As the customer adds a product with the same or a duplicate weight as the initial ones found on the PVC plate (weight sensor), the count increment by 1 or the weight duplicate. Moreover, the sensor is not accurate enough for products with light weight.

Hence, the problems faced are:

- Items with the same weight cannot be differentiated.
- Items with duplicate weight cannot be differentiated.
- Items with Light weight cannot be differentiated or read accurately.
- It's not precise enough because it differs if we changed the place of the product on the PVC plate.

Proposed Solutions:

In order to better implementation of the experiment and achieve better results, several solutions could be proposed:

- Integrating the weight sensor with another sensor (Ex: camera)
- Using more accurate sensor.
- Stop the counter from adding duplicates.
- Changing the planogram.

The circuit:

Below is the implemented circuit using Arduino UNO, Full-bridge weight sensor, PVC plate, HX711 Module, LCD screen and jumper wires (figure4).

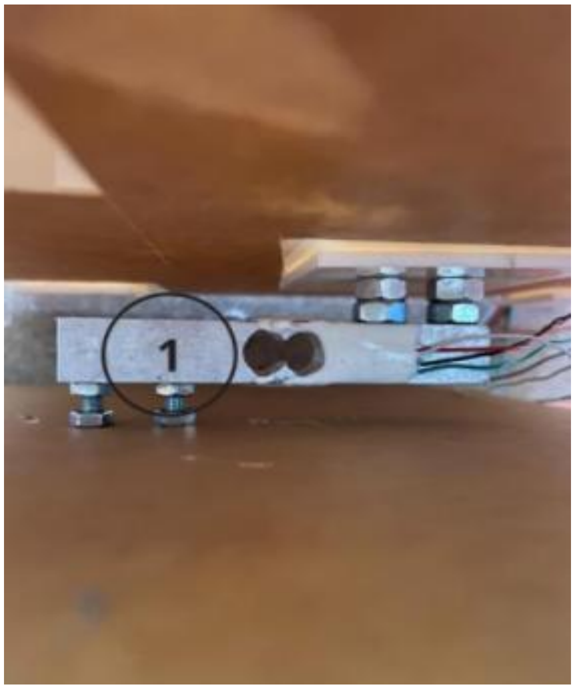


FIGURE 7.7: The Implemented Circuit

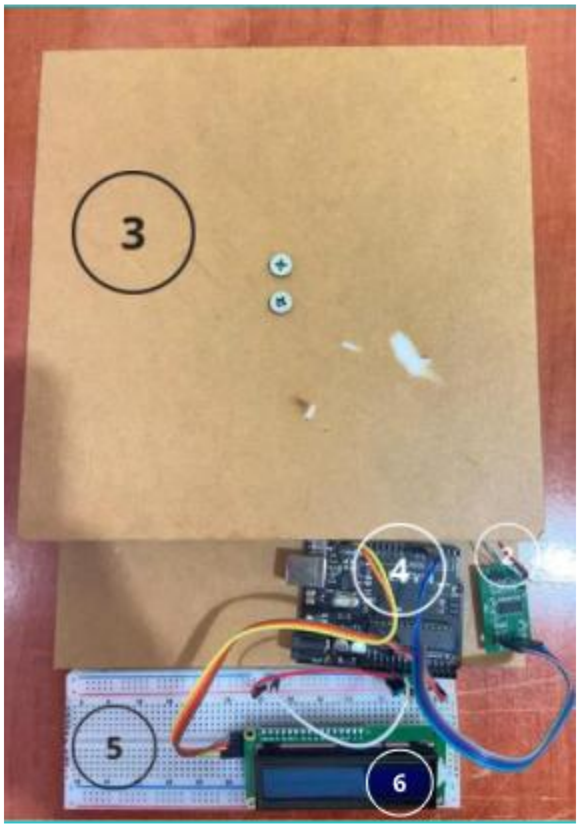


FIGURE 7.8: The Implemented Circuit

COMPONENT'S NUMBER	COMPONENT'S NAME
1	Full-bridge Weight sensor
2	HX711 Module
3	PVC Plate
4	ARDUINO UNO
5	Bread board
6	LCD Screen

TABLE 3: CIRCUIT COMPONENT'S NAME AS SHOWN IN FIGURE 7 and 8

3. CONCLUSION

As stated before, the main objective behind the above experiment is to find the quantity of products placed on the Stand (PVC plate) as a function of weight read by the Full-bridge weight sensor. As customers take-off products, weight sensor reads a weight less than the initial one hence the counter displayed on the screen decrements showing the new number of products found on the pusher. On the other hand, the counter increments and the screen also displays the new number of products found.

4. APPENDIX

```
#include <HX711_ADC.h>
#if defined(ESP8266)|| defined(ESP32) || defined(AVR)
#include <EEPROM.h>
#endif

const int HX711_dout = 4; //mcu > HX711 dout pin
const int HX711_sck = 5; //mcu > HX711 sck pin

//HX711 constructor:
HX711_ADC LoadCell(HX711_dout, HX711_sck);

const int calVal_eepromAdress = 0;
unsigned long t = 0;

void setup() {
  Serial.begin(57600); delay(10);
  Serial.println();
  Serial.println("Starting...");

  LoadCell.begin();
  float calibrationValue;
  calibrationValue = 696.0;
#if defined(ESP8266)|| defined(ESP32)

#endif

  unsigned long stabilizingtime = 2000;
  boolean _tare = true;
  LoadCell.start(stabilizingtime, _tare);
  if (LoadCell.getTareTimeoutFlag()) {
    Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
    while (1);
  }
  else {
    LoadCell.setCalFactor(calibrationValue);
    Serial.println("Startup is complete");
  }
}

void loop() {
  static boolean newDataReady = 0;
  const int serialPrintInterval = 0;


  if (LoadCell.update()) newDataReady = true;
```

```
 if (newDataReady) {
   if (millis() > t + serialPrintInterval) {
     float i = LoadCell.getData();
     Serial.print("Load_cell output val: ");
     Serial.println(i);
     newDataReady = 0;
     t = millis();
   }
 }

if (Serial.available() > 0) {
   char inByte = Serial.read();
   if (inByte == 't') LoadCell.tareNoDelay();
 }

   if (LoadCell.getTareStatus() == true) {
   Serial.println("Tare complete");
 }

}
```

4.5 Product Count Using LDR sensors

ABSTRACT

The objective of this experiment is to find the type and quantity of every product found on the retail shelf. We should design and implement a circuit using an Arduino Uno, photoresistor sensors (LDR). As a conclusion, count of every type of products will be displayed on a screen.

Keywords: Arduino Uno, LDR sensor.

1. INTRODUCTION

The main objective behind this experiment is to find the type and quantity of products found on the retail shelf. Because of the complexity of the wiring, we tried to use the simplest LDR matrix, so we implemented 16 LDR in the form of a (2*8) matrix on a breadboard. When no items covered the LDR's, a (2*8) matrix of zeros will be displayed on the screen (in this case Serial monitor). After putting an item on the LDR's, this item will block out the light, and the LDR covered by this item will give ones instead of zeroes. This matrix will be analyzed using python libraries to know base shape, the type and the count of product present on the shelf. When the item is classified, the count will be calculated based on the number of shapes detected. Block diagram is shown in figure 1.

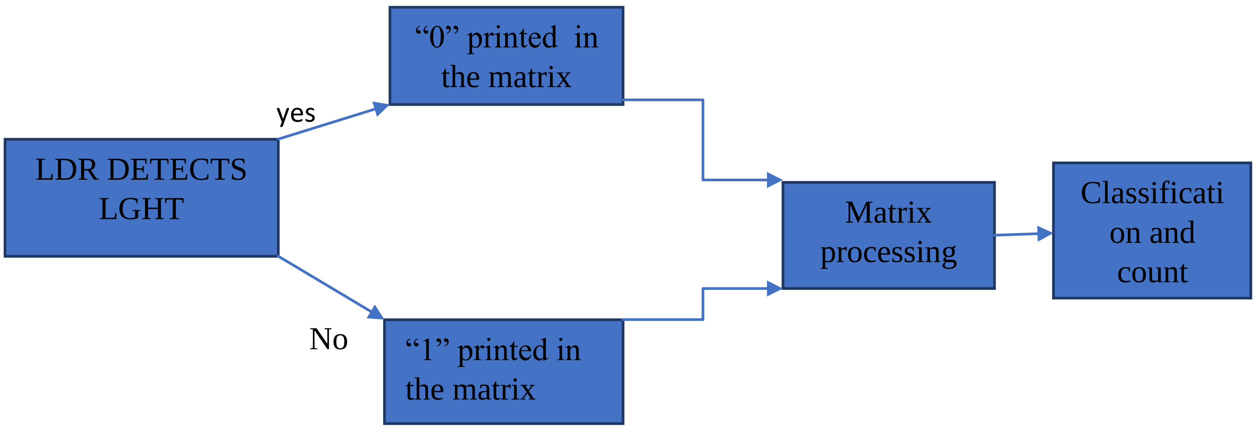


FIGURE *8.1*: LDR SENSOR BLOCK DIAGRAM

\

2. MATERIALS AND METHODS

1.1 Components

- LDR sensor

  LDR or photoresistor is a passive component that decreases resistance with respect to receiving luminosity on the component's sensitive surface. The resistance of a photoresistor decreases with increase in incident light intensity

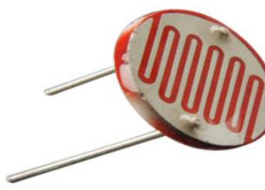


FIGURE 8.6:LDR SENSOR

- 74HC4067 multiplexer:

  The 74HC4067 multiplexer is a single-pole *16*-throw *analog switch* (*SP16T*) suitable for use in *analog* or digital.

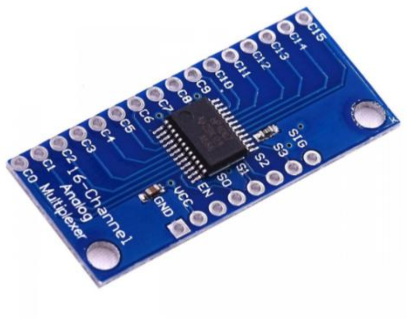


FIGURE 8.3: 16 CHANNELS MULTIPLEXER

- 10 kohms resistors
  The *resistor* is a passive electrical component that creates resistance in the flow of electric current.in this experiment, they will be implemented in series with
  LDR sensor in the voltage divider.

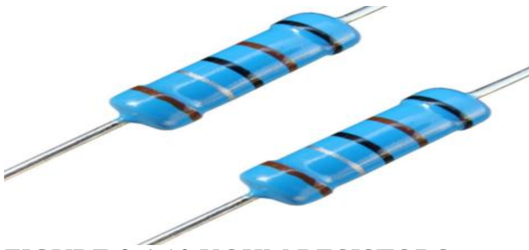


FIGURE 8.4:10 KOHM RESISTORS

1.2 Methodology

The methodology will be divided into three parts:
_Test 1: Using Arduino to read from LDR matrix.
_Test 2: Sending arrays of zeros and ones serially to PyCharm IDE.
_Test 3: Analyzing the matrix to determine the shape and the area of a product on the shelf.

2.21. Test 1: Using Arduino to read from LDR matrix:

3. Pre_Implmentaion:

-Wiring:

LDR INPUTS

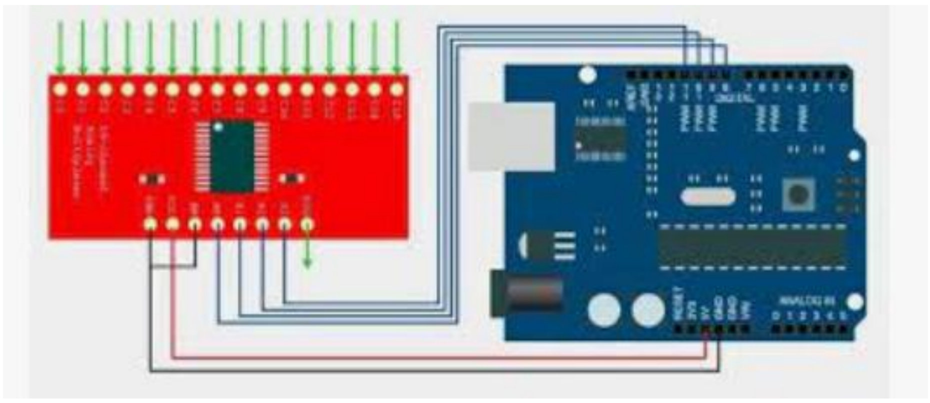


FIGURE 8.5: LDR_MULTIPLEXER_ARDUINO WIRING

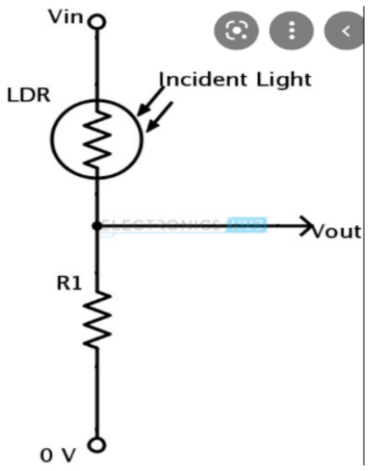


Figure 8.6: LDR implementation in voltage divider

As shown in **figure 6**, every LDR is connected in series with 10 K resistors forming a voltage divider. the Vout of every voltage divider formed by 1 resistor and one LDR is connected to pins(C0_C15)of the multiplexer. the VCC pin of the multiplexer is connected to 5 V of the Arduino, and GND pin to ground. Pins (S0_S3)are connected respectively to pins (8_9_10_11) of the Arduino. The Sig pin of the multiplexer is connected to analog input pin A0 of the Arduino.

-Principle of operation:

Arduino sends simultaneously pulses to the 4 input pins of the multiplexer(S0_S3) by 16 iterations each loop, and each iteration triggers one pin of the 16 pins of the multiplexer(C0_C15), to read analog input from the LDRs through the "SIG" pin of the multiplexer. The analog output of every LDR in a voltage divider has a range (0_1023), and increases when subjected to light. These analog values will be converted to zeros and ones by comparing these analog values to a threshold (400) to differentiate the LDRs in light or in darkness.

-Design Drawing:

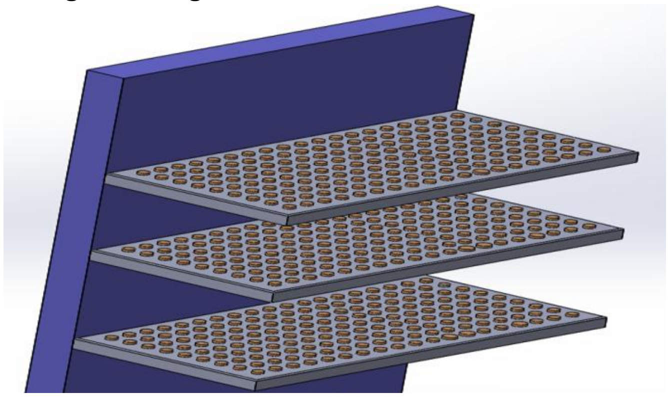


FIGURE 8.7: CAD OF LDRs IMPLENTED ON THE SHELF

LDR's will be implemented on the top of every shelf in the form of a matrix. The size of this matrix and the area between 2 LDR will be know to obtain the shape and the area of the item on the top of the shelf.

<u>**2.212-Result and Discussion:**</u>

After implementing 16 LDR on a breadboard, we run an Arduino code to get readings from the LDR and classify the LDR in light or in darkness. when the LDR detect light, The serial monitor will print "0", when it is in darkness, the serial monitor will print "1". these ones and zeros will be arranged in an array to be sent serially to PyCharm workspace in test 2.
So, when an object covered a number of LDR, same number of ones will appear in the array.

Problems faced:

- Transparent Items cannot be detected.
-Items with same base shape will give same array of zeros and ones.
-Items very close to each other cannot be differentiated and treated as one bigger item.

Proposed Solutions:

- Integrate with another sensor (weight Sensor or pressure Sensor)

The circuit:
Below is the implemented circuit using Arduino UNO, 16 LDR, breadboard,16 channel analog multiplexer,10 kohms resistors .(figure 8)

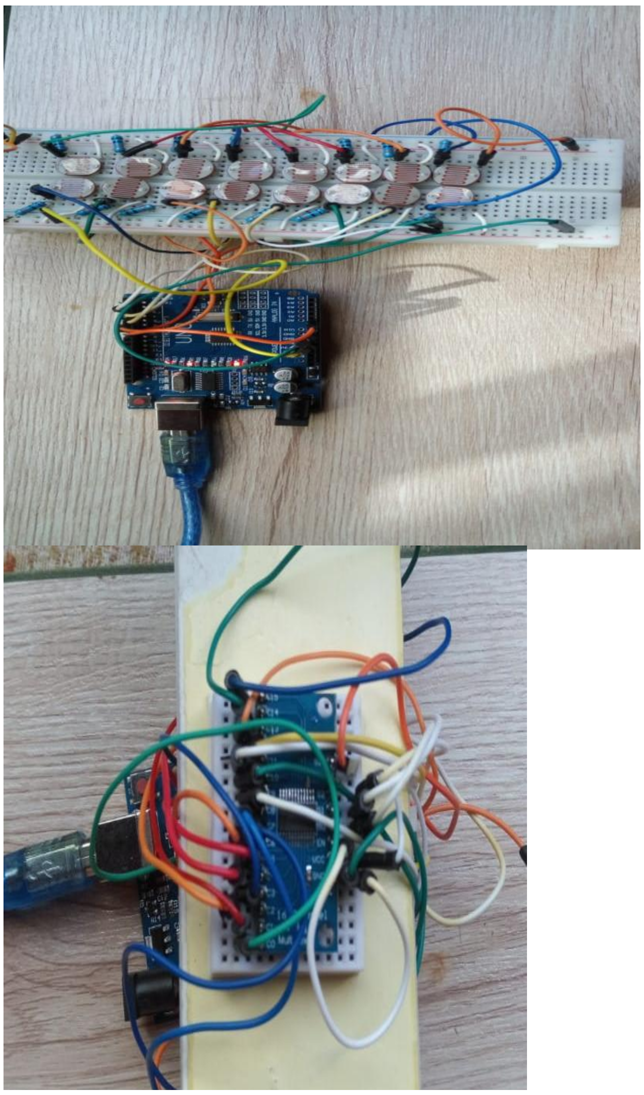


FIGURE 8.8: THE IMPLEMENTED CIRCUIT

4. CONCLUSION:

At the end of this test, we were able to get values from the LDR sensors, and arrange them in an array to be sent serially to PyCharm workspace, but some problems are faced that might be solved by integration with another sensor like weight or pressure sensors.

5. Appendix 1:

```
#define in A0

#define S0 2

#define S1 3

#define S2 4

#define S3 5


int arr[16][4] = {{0, 0, 0, 0},

  {0, 0, 0, 1},

  {0, 0, 1, 0},

  {0, 0, 1, 1},

  {0, 1, 0, 0},

  {0, 1, 0, 1},

  {0, 1, 1, 0},

  {0, 1, 1, 1},

  {1, 0, 0, 0},

  {1, 0, 0, 1},

  {1, 0, 1, 0},

  {1, 0, 1, 1},

  {1, 1, 0, 0},

  {1, 1, 0, 1},

  {1, 1, 1, 0},

  {1, 1, 1, 1}

};
```

```
int ldr_values_arr[8][2];

int ldr_value = 0;

int ldr_analog = 0;

//int j = 7;

void setup() {
  // put your setup code here, to run once:
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);

  pinMode(A0, INPUT);

  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
 int  j = 7;



  for (int i = 0; i <= 15; i++) {
    digitalWrite(S0, arr[i][3]);
    digitalWrite(S1, arr[i][2]);
    digitalWrite(S2, arr[i][1]);
    digitalWrite(S3, arr[i][0]);
    ldr_analog = analogRead(in);
    if(ldr_analog > 450){
     ldr_value = 0;
    }
    else{
     ldr_value = 1;
    }
    if (i > 7) {
     ldr_values_arr[i-8][1] = ldr_value;
     //Serial.println(j);
     //Serial.println(ldr_value);
     //delay(10);
    }
    else {
     ldr_values_arr[i][0] = ldr_value;
    }
    //Serial.println(i);
    //Serial.println(ldr_value);
  }

  for (int k = 0; k <= 7; k++) {
    for (int l = 0; l <= 1; l++) {
     Serial.print(ldr_values_arr[k][l]);
     //Serial.print("/");
    }
    Serial.print(",");
  }
  Serial.print("\n");
```

delay(100);

2.22_Test 2: Sending arrays of zeros and ones serially to PyCharm workspace:

After facing some difficulties to arrange the zeroes and ones in a matrix, we find that using python code is more useful to deal with matrices.so the array of zeroes and ones obtained in **test1** will be sent serially to PyCharm workspace and then this array will be arranged in a matrix with the same size of the same shape of the real LDR matrix on the breadboard.
Actually this test is a software implementation of a python code to receive the array of zeroes and one from the Arduino. So we use python libraries: infix notation, serial,numpy,and re.

Results and conclusion:

At the end of this test, we were able to send the array of zeroes and ones obtained in test 1 ,and arranged in a matrix with the same size of the LDR matrix on the breadboard.

Appendix 2:

```
-   from pyparsing import infix_notation
    import serial
    import numpy as np
    import re

    ldr_matrix = np.ones((8,2))

    s = serial.Serial("COM4", 115200)

    income = s.readline()

    str_income = str(income)
    str_income = str_income.strip("b''")

    indices = [i.start() for i in re.finditer(",", str_income)]
    print(indices)
    print(str_income[indices[0]+1:indices[1]])
    for i in range(len(indices)):
        #print(i)
        if(i==0):
            nums = str_income[0 : indices[i]]
    else:
        nums = str_income[indices[i-1]+1:indices[i]]

    print(i)
    print(nums[0])
    print(type(nums[0]))
    ldr_matrix[i][0] = float(nums[0])
    ldr_matrix[i][1] = float(nums[1])
```

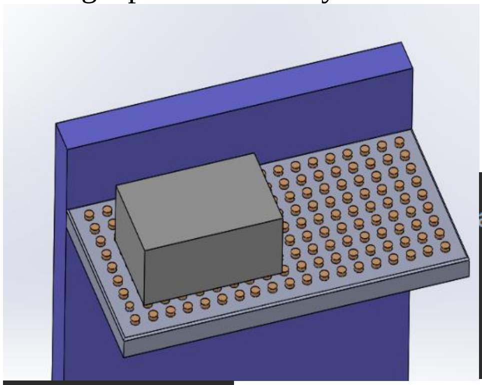
2.23_TEST3: Analyzing the matrix to determine the shape and the area of item on the Shelf :

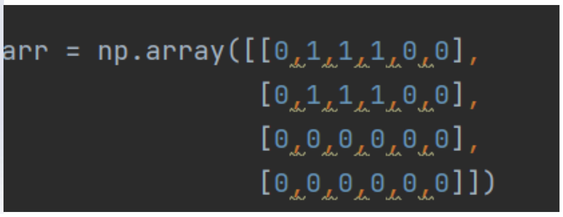
After arranging the reading obtained from LDR 's in a matrix, it's time to analyze this matrix of zeroes and one to obtain the shape of the and area of an item covering a number of LDR's on the shelf.
In this part, for simplicity, we began by putting some matrices, and we tried to use python code to obtain the shape of ones between the zeroes. First, the size of the real LDR matrix, and the distance between the LDR should be Known .So if the shape formed by the ones is rectangle ,then the object on the LDRs is a rectangle and the area is (x*y)where x and y are the dimensions of the rectangle. Here,The distance between the LDRs is assumed to be equal to one unit.

6. Results:

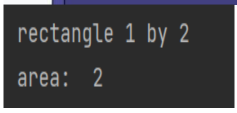
Trial 1:

-We started by the simplest shape which is rectangle.we represented one rectangle putted vertically on the shelf.





```
arr = np.array([[0,1,1,1,0,0],
                [0,1,1,1,0,0],
                [0,0,0,0,0,0],
                [0,0,0,0,0,0]])
```

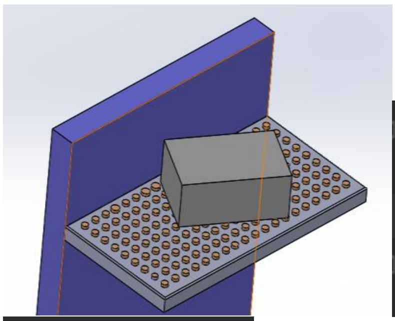


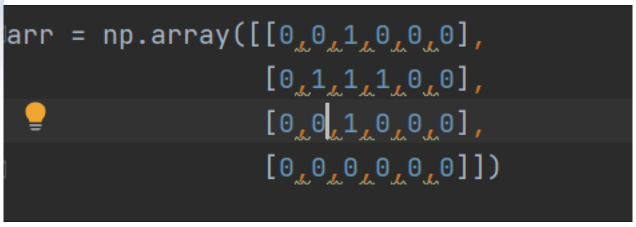
```
rectangle 1 by 2
area:  2
```

FIGURE 8.9: SHEME OF AN OBJECT ON THE SHELF WITH ITS EQUIVALENT MATRIX AND THE AREA AND SHAPE OBTAINED FROM THE CODE

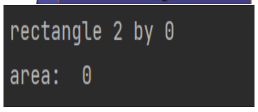
Trial 2:

-Then we represented a rectangle object but rotated some angle neither horizontally nor vertically as shown in figure 10 .and we found that is impossible to detect the shape and area of this object using python code.





```
arr = np.array([[0,0,1,0,0,0],
                [0,1,1,1,0,0],
                [0,0,1,0,0,0],
                [0,0,0,0,0,0]])
```

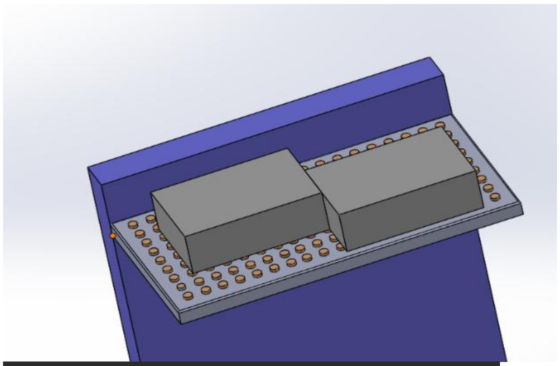


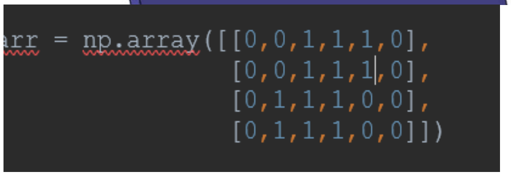
```
rectangle 2 by 0
area:  0
```

FIGURE 8.91: SHEME OF A RECTANGLE ITEM ROTATED ON THE TOP OF THE SHELF WITH ITS EQYUIVALENT MATRIX AND THE RESULT OF THE PYTHON CODE

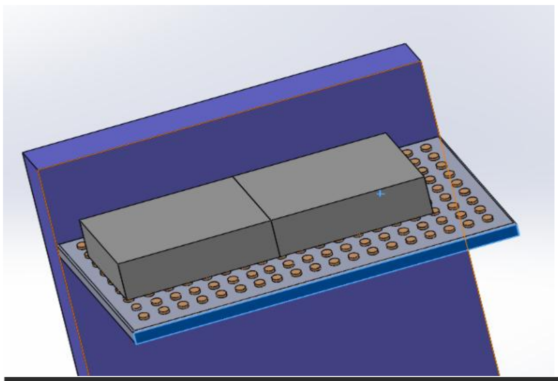
Trial 3:

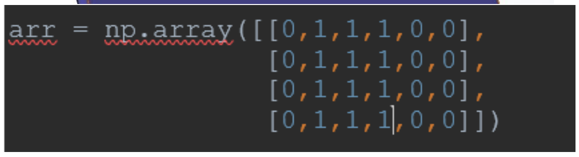
In this trial ,we represented 2 rectangle items on the top of the shelf, very close to each other ,and we find that it is impossible to differentiate between them, and they will be treated as a single object .





```
arr = np.array([[0,0,1,1,1,0],
                [0,0,1,1,1,0],
                [0,1,1,1,0,0],
                [0,1,1,1,0,0]])
```





```
arr = np.array([[0,1,1,1,0,0],
                [0,1,1,1,0,0],
                [0,1,1,1,0,0],
                [0,1,1,1,0,0]])
```

FIGURE 8.92: SHEME OF 2 RECTANGLE ITEMS VERY CLOSE TO EACH OTHER ON THE TOP OF THE SHELF WITH ITS EQUIVALENT MATRICES.

Problems faced:

- the shape and area of a product cannot be detected From the LDRs matrix using simple python codes.

Proposed solution:

- usage of machine learning to analyze pressure sensor array data as 3 dimensions x/y and z for pressure readings.

Appendix 3:

```
import numpy as np

arr = np.array([[0,0,1,1,1,0],
                [0,0,1,1,1,0],
                [0,1,1,1,0,0],
                [0,1,1,1,0,0]])


arr2 = np.sort(arr)

list = []
ones = np.where(arr2==1)
for elements in ones:
    list.append(elements[0])
    list.append(elements[-1])

x = (list[1] - list[0]) + 1
y = (list[3] - list[2]) + 1

area = x * y

print(x)
print(y)

print("area: ", area)
```

3.CONCLUSION

The implementation of light dependent resistors (LDR) in a matrix on the shelf has a lot of limitations in both hardware and software part and cannot be used to detect and count items on the shelf.

Rafik Hariri University
جـامـعـة رفـيـق الـحـريـري

APPENDICES

A. ADDRESSING STUDENT OUTCOMES' KPIS

	How was it addressed in your SLP?	***Where was it addressed in your SLP?***
1. An ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics		
1.1 An ability to apply knowledge of mathematics	***Numerical Analysis methods were used to find the weight and optical sensors equations***	***Chapter 4 sec. 4.1*** ***sec. 4.3***
1.2 An ability to apply knowledge of Science	***Ex: Optics Analysis used for Fisheye Camera and reflection of IR sensor.***	***Chapter 4*** ***sec. 4.1*** ***sec. 4.2***
1.3 An ability to apply knowledge of Engineering	***MECA 540:*** ***State applied knowledge in the area Computer Aided Design (CAD) used to design a retail pusher holder*** ***MECA 443:*** ***State applied knowledge in the area of Mechatronics system design strategies, advanced modeling, hardware and software integration***	***Chapter 4***
2. An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors		

2.1 Design a system/component of a system or a process to meet specific needs while respecting safety, health and welfare of the public and adhering to cultural, social, environmental and economic factors.	***Several designs were proposed in order to achieve the project's aim and the optimum design was selected to satisfy all listed technical criteria.***	***Chapter 3*** ***Sec. 1*** ***Sec. 2*** ***Sec. 3*** ***Sec. 4***
2.2 Modify a system/component of a system or a process to meet specific needs while respecting safety, health and welfare of the public and adhering to cultural, social, environmental and economic factors.	***The first design was by implementing the camera on another shelf, but this is against the retail laws, hence this design was improved by integrating the fish-eye camera on the shelf***	***Chapter 3*** ***Page 19*** ***Page 20***
3. An ability to communicate effectively with a range of audiences		
3.1 Ability to write a well-structured formal report/technical document that addresses an audience with diverse educational-background	***Submitting several technical reports and meetings' minutes to adviser(s) and jury.***	***SLPI report (February 21, 2022) + SLPII Progress Report (April 28, 2022) + SLPII final report (May 4, 2022)***
3.2 Ability to deliver a well-structured formal presentation that addresses an audience with diverse educational-background[1]	***Delivering multiple presentations and demonstrating different phases of the project to adviser(s) and jury members***	***SLPI presentation (February 21, 2022) + SLPII final presentation (April 29, 2022)***
4. An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts		
4.1 Identify global, economic, environmental, and societal impact of implementing engineering solutions using applicable engineering code of ethics to differentiate between ethical/unethical behaviors	***First canon of ASME: Engineers shall hold paramount the safety, health …/ By building a stand that has the safety required.*** ***Fifth canon of ASME: Engineers shall build their***	***Interviews And Brainstorming Session*** ***+ Appendix B (Minutes Of Meeting 2 (30***

	professional ... / When using a camera to collect data about the user/customer we must tell him that.	***minutes))***
4.2 Identify global, economic, environmental, and societal impact of implementing engineering solutions using applicable engineering standards and codes to differentiate between professional/unprofessional behaviors	***Fourth canon of ASME:*** ***Engineers shall act in professional matters for each employer or client as faithful agents or*** ***trustees. / When using a camera to collect data about the user/customer we must tell him that.***	***Appendix B (Minutes of Meeting 2 (30 Minutes))*** + ***Interviews And Brainstorming Session***
5. An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives		
5.1 Ability to plan and organize team tasks collectively to meet established goals	***The team agreed on having each working on a specific task with a collaborative supervision throughout the whole process. Actions are properly documented in minutes of meeting and managed by Gantt chart***	***Appendix B (Minutes of Meeting 4 (30 minutes)) + <u>Gantt chart</u> in sec. 5***
5.2 Ability to carry out tasks assigned by the team to attain set objectives.	***The team was divided to perform four main tasks, work on three different sensors and a fish-eye camera. In addition, the whole team was performing daily meetings to abide by the agreed upon time plan to ensure efficient accomplishment of all tasks in due time.***	***Progress of the project as illustrated in Gantt chart….*** + ***Appendix B (Minutes of Meeting 5 (30 minutes))***
6. An ability to develop and conduct appropriate experimentation, analyze and interpret data, and		

use engineering judgment to draw conclusions		
6.1 An ability to design experiments.	***Different experiments were designed to test the sensors, CADs and object detection using fish-eye camera.***	***Chapter 3***
6.2 An ability to conduct experiments.	***Different experiments were conducted to test the sensors and object detection using fish-eye camera***	***Chapter 4***
6.3 An ability to draw apt evidence-based conclusions by analyzing and interpreting data	***Data collected from experiments to test the sensors and the fish-eye camera are presented, analyzed and figures are results using several trials.***	***Chapter 4*** ***Sec. 4.1*** ***Sec. 4.2*** ***Sec. 4.3*** ***Sec. 4.4***
7. An ability to acquire and apply new knowledge as needed, using appropriate learning strategies.		
7.1 Identify necessary skills and tools of contemporary engineering practice to solve a problem at hand.	***Using SolidWorks, we developed a model for the retail pusher in order to implement on the stand and running fatigue analysis.*** ***Moreover, using Pycharm in order to train different brands and products hence achieving product detection.***	***Chapter 3*** ***Page 12*** ***Chapter 4*** ***Sec. 4.1*** ***Sec 4.2***
7.2 Apply self-learned skills and tools of contemporary engineering practice to solve a problem at hand.	***Using distance and weight sensors, we can precisely collect the count of products, take off count and out of stock data.*** ***Moreover, using the fish-eye camera, we can detect the product***	***Chapter 4*** ***Sec 4.1*** ***Sec 4.2*** ***Sec 4.3***

	brands found on shelves and data analysis of customer's demographics. ***Simulation results achieved by Pycharm, Jetson and Arduino reflected the behavior of the actual prototype.***	

Rafik Hariri University
جامعة رفيق الحريري

B. MINUTES OF MECH 595A MEETING

MINUTES OF MECH 595A MEETING (1)

COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON October 25th, 2021, AT 10:00 AM

Present: Hassan Hariri (advisor), Nadim Inaty (CEO Flexpy), Eyad Shayya (student), Abdulhakim Hujayri (student).

Absent: Hadi Dergham (student), Manwel Shdeed (student).

The meeting came to order at 10:00 am.

1. Discussions and Updates

Discussed the product's competitors and the interviews' questions.

2. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- Likes and Dislikes
- Questionnaire's formulation
- Safety measures
- Project management

3. Expected Deliverables for Next Meeting

Assigned to prepare a gant chart and updated questionnaires.

4. Assessment

The meeting was adjourned at 11:00 am.

Minutes taken by: Eyad Shayya

Rafik Hariri University
جـامـعـة رفـيـق الـحـريـري

MINUTES OF MECH 595A MEETING (2)

COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON October 27th, 2021, AT 7:00 PM

Present: Hassan Hariri (advisor), Eyad Shayya (student), Abdulhakim Hujayri (student), Hadi Dergham (student), Manwel Shdeed (student).

Absent: None.

The meeting came to order at 7 pm.

5. Discussions and Updates

Discussed the updates on product’s competitors and the interviews’ questions.

6. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- Likes and Dislikes
- Questionnaire’s formulation
- Safety measures
- Project management

7. Expected Deliverables for Next Meeting

Assigned to prepare a gant chart.

8. Assessment

The meeting was adjourned at 8:30 pm.

Minutes taken by: Hadi Dergham

Rafik Hariri University
جـامـعـة رفـيـق الـحـريـري

MINUTES OF MECH 595B MEETING (3)

COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON February 16th, 2021, AT 8:00 PM

Present: Hassan Hariri (advisor), Eyad Shayya (student), Abdulhakim Hujayri (student), Hadi Dergham (student), Manwel Shdeed (student).

Absent: None.

The meeting came to order at 8 pm.

9. Discussions and Updates

Discussed the updates of the hands on implementations of the sensors.

10. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- Time of finishing the implementation of the sensors
- BE2 deadline
- PowerPoint Update
- Results of experiments

11. Expected Deliverables for Next Meeting

Photos and results of our experiments

12. Assessment

The meeting was adjourned at 8:30 pm.

Minutes taken by: Hadi Dergham

Rafik Hariri University
جامعة رفيق الحريري

MINUTES OF MECH 595B MEETING (4)
COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON February 21th, 2021, AT 1:00 PM

Present: Hassan Hariri (advisor), Eyad Shayya (student), Abdulhakim Hujayri (student), Hadi Dergham (student), Manwel Shdeed (student).

Absent: None.

The meeting came to order at 1 pm.

13. Discussions and Updates

Discussed the updates of the hands-on implementations of the sensors.

14. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- Time of finishing the implementation of the sensors
- Report for every sensor
- Planogram Products
- Results of experiments

15. Expected Deliverables for Next Meeting

Results for our experiments and discussion.

16. Assessment

The meeting was adjourned at 1:30 pm.

Minutes taken by: Hadi Dergham

Rafik Hariri University
جامعة رفيق الحريري

MINUTES OF MECH 595B MEETING (5)

COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON March 7th, 2021, AT 1:00 PM

Present:	Hassan Hariri (advisor), Eyad Shayya (student), Abdulhakim Hujayri (student), Hadi Dergham (student), Manwel Shdeed (student).
Absent:	None.

The meeting came to order at 1 pm.

17. Discussions and Updates

Discussed the updates of the hands-on implementations of the sensors.

18. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- Generate reports for the sensors
- Results of experiments

19. Expected Deliverables for Next Meeting

Updates of the sensors experiments.

20. Assessment

The meeting was adjourned at 1:30 pm.

Minutes taken by: Hadi Dergham

Rafik Hariri University
جامعة رفيق الحريري

MINUTES OF MECH 595B MEETING (6)

COLLEGE OF ENGINEERING – MME Department – RHU

Group I

ON March 14th, 2021, AT 1:00 PM

Present: Hassan Hariri (advisor), Eyad Shayya (student), Hadi Dergham (student).

Absent: None.

The meeting came to order at 1 pm.

21. Discussions and Updates

Discussed the updates of the hands-on implementations of the sensors.

22. Advisor Comments and Recommendations

Several ideas were discussed about reformulating the questions. The main points are:

- To put a section for the problems and limitations of each method and sensor.
- To precise final BE tasks.

23. Expected Deliverables for Next Meeting

Final BE tasks.

24. Assessment

The meeting was adjourned at 1:30 pm.

Minutes taken by: Hadi Dergham

C. GANT CHART

Act	Project	ES	EF	LS	LF	Slack	Predecessor	Optimistic	Most Likely	Pessimistic	Expected	Var	std
a	Primary Research	0	3	0	3	0	-	1	3	5	3	0.444444444444	0.66666666667
b	Brain Storming Session	3	4	3	4	0	a	1	1	1	1	0	0
c	Early prototyping of individual Functionalities	4	9	4	9	0	a, b	3	4	11	5	1.777777778	1.33333333
d	Secondary Research	9	12	9	12	0	c	1	2	9	3	1.777777778	1.33333333
e	Assembly of Interdepedendant Sub-systems (Prototypes)	12	16	12	16	0	b,c,d	2	3	10	4	1.777777778	1.33333333
f	Assembling Functional Prototype	16	20	16	20	0	e	1	3	11	4	2.777777778	1.66666667

D. REFERENCES

[1] https://shopperception.com/

[2] https://www.awm.tech/

[3]https://www.thingsquare.com/blog/articles/retail-wireless-shelf-monitoring/

[4]https://www.digiteum.com/internet-of-things-retail-industry/

[5]https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a41sk_e.pdf

[6]https://www.hubert.com/categories/Shelving-Racks-Floor-Fixtures-453022/Shelf-Management-453149/Shelf-Management-Pushers-453801

[7] https://www.elecdesignworks.com/shop/prod0236/

[8] https://store.arduino.cc/usa/arduino-uno-rev3

[9] https://www.thingbits.in/products/standard-lcd-16x2-display

[10] https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[11] https://www.monodaq.com/applications/docs/u-x/front-end/load-cells/